Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

ENERGY AND PERFORMANCE-OPTIMIZED SCHEDULING OF TASKS IN DISTRIBUTED CLOUD AND EDGE COMPUTING SYSTEMS

by
Haitao Yuan

Infrastructure resources in distributed cloud data centers (CDCs) are shared by heterogeneous applications in a high-performance and cost-effective way. Edge computing has emerged as a new paradigm to provide access to computing capacities in end devices. Yet it suffers from such problems as load imbalance, long scheduling time, and limited power of its edge nodes. Therefore, intelligent task scheduling in CDCs and edge nodes is critically important to construct energy-efficient cloud and edge computing systems. Current approaches cannot smartly minimize the total cost of CDCs, maximize their profit and improve quality of service (QoS) of tasks because of aperiodic arrival and heterogeneity of tasks. This dissertation proposes a class of energy and performance-optimized scheduling algorithms built on top of several intelligent optimization algorithms. This dissertation includes two parts, including background work, *i.e.*, Chapters 3–6, and new contributions, *i.e.*, Chapters 7–11.

1) Background work of this dissertation.

Chapter 3 proposes a spatial task scheduling and resource optimization method to minimize the total cost of CDCs where bandwidth prices of Internet service providers, power grid prices, and renewable energy all vary with locations. Chapter 4 presents a geography-aware task scheduling approach by considering spatial variations in CDCs to maximize the profit of their providers by intelligently scheduling tasks. Chapter 5 presents a spatio-temporal task scheduling algorithm to minimize energy cost by scheduling heterogeneous tasks among CDCs while meeting their delay constraints. Chapter 6 gives a temporal scheduling algorithm considering temporal variations of revenue, electricity prices, green energy and prices of public clouds.

2) Contributions of this dissertation.

Chapter 7 proposes a multi-objective optimization method for CDCs to maximize their profit, and minimize the average loss possibility of tasks by determining task allocation among Internet service providers, and task service rates of each CDC. A simulated annealing-based bi-objective differential evolution algorithm is proposed to obtain an approximate Pareto optimal set. A knee solution is selected to schedule tasks in a high-profit and high-quality-of-service way. **Chapter 8** formulates a bi-objective constrained optimization problem, and designs a novel optimization method to cope with energy cost reduction and QoS improvement. It jointly minimizes both energy cost of CDCs, and average response time of all tasks by intelligently allocating tasks among CDCs and changing task service rate of each CDC. **Chapter 9** formulates a constrained bi-objective optimization problem for joint optimization of revenue and energy cost of CDCs. It is solved with an improved multi-objective evolutionary algorithm based on decomposition. It determines a high-quality trade-off between revenue maximization and energy cost minimization by considering CDCs' spatial differences in energy cost while meeting tasks' delay constraints. **Chapter 10** proposes a simulated annealing-based bees algorithm to find a close-to-optimal solution. Then, a fine-grained spatial task scheduling algorithm is designed to minimize energy cost of CDCs by allocating tasks among multiple green clouds, and specifies running speeds of their servers. **Chapter 11** proposes a profit-maximized collaborative computation offloading and resource allocation algorithm to maximize the profit of systems and guarantee that response time limits of tasks are met in cloud-edge computing systems. A single-objective constrained optimization problem is solved by a proposed simulated annealing-based migrating birds optimization. This dissertation evaluates these algorithms, models and software with real-life data and proves that they improve scheduling precision and cost-effectiveness of distributed cloud and edge computing systems.

ENERGY AND PERFORMANCE-OPTIMIZED SCHEDULING OF TASKS IN DISTRIBUTED CLOUD AND EDGE COMPUTING SYSTEMS

by
Haitao Yuan

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

August 2020

APPROVAL PAGE

ENERGY AND PERFORMANCE-OPTIMIZED SCHEDULING OF TASKS IN DISTRIBUTED CLOUD AND EDGE COMPUTING SYSTEMS

Haitao Yuan

MengChu Zhou, Dissertation Co-Advisor — Date
Distinguished Professor, Department of Electrical and Computer Engineering, NJIT

Qing Liu, Dissertation Co-Advisor — Date
Assistant Professor, Department of Electrical and Computer Engineering, NJIT

Nirwan Ansari, Committee Member — Date
Distinguished Professor, Department of Electrical and Computer Engineering, NJIT

Abdallah Khreishah, Committee Member — Date
Associate Professor, Department of Electrical and Computer Engineering, NJIT

Xinyue Ye, Committee Member — Date
Associate Professor, Department of Informatics, NJIT

BIOGRAPHICAL SKETCH

Author: Haitao Yuan

Degree: Doctor of Philosophy

Date: August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Doctor of Philosophy in Modeling Simulation Theory and Technology,
  Beihang University, Beijing, China, 2016
- Master of Science in Software Engineering,
  Northeastern University, Shenyang, China, 2012
- Bachelor of Science in Software Engineering,
  Northeastern University, Shenyang, China, 2010

Major: Computer Engineering

Presentations and Publications:

Journal articles:

Haitao Yuan, Jing Bi and MengChu Zhou, "Geography-Aware Task Scheduling for Profit Maximization in Distributed Green Data Centers," *IEEE Transactions on Cloud Computing*, pp. 1–11, June 1, 2020, DOI: 10.1109/TCC.2020.3001051.

Haitao Yuan, MengChu Zhou, Qing Liu and Abdullah Abusorrah, "Fine-grained and Arbitrary Task Scheduling for Heterogeneous Applications in Distributed Green Clouds," *IEEE/CAA Journal of Automatica Sinica*, pp. 1–13, DOI: 10.1109/JAS.2020.1003177, Online, May 2020.

Haitao Yuan, Jing Bi, MengChu Zhou, Qing Liu and Ahmed Ammari, "Bi-objective Task Scheduling for Distributed Green Data Centers," *IEEE Transactions on Automation Science and Engineering*, DOI: 10.1109/TASE.2019.2958979, Online, Jan. 2020.

Haitao Yuan, Heng Liu, Jing Bi and MengChu Zhou, "Revenue and Energy Cost-Optimized Bi-objective Task Scheduling for Green Cloud Data Centers," *IEEE Transactions on Automation Science and Engineering*, DOI: 10.1109/TASE.2020.2971512, Online, Jan. 2020.

Haitao Yuan and MengChu Zhou, "Profit-maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, June 4, 2020, Accepted.

Haitao Yuan, Jing Bi and MengChu Zhou, "Energy-consumption and Performance-optimized Task Scheduling in Distributed Data Centers," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, pp. 1–9, March 17, 2020, Submitted.

Haitao Yuan and MengChu Zhou, "Energy-efficient and QoS-optimized Adaptive Task Scheduling and Management in Cloud Data Centers," *IEEE Transactions on Automation Science and Engineering*, pp. 1–10, May 7, 2020, Submitted.

Jing Bi, **Haitao Yuan**, MengChu Zhou, Qing Liu and Jia Zhang, "ARIMA-Based Multi-Application Workload Prediction with Wavelet Decomposition and Savitzky-Golay Filter in Clouds," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, pp. 1–10, September 14, 2019, Major Revision.

Jing Bi, Shuang Li, **Haitao Yuan** and MengChu Zhou, "BG-LSTM: Integrated Deep Learning Method for Workload and Resource Prediction in Cloud Computing Systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–9, June 11, 2020, Minor Revision.

Xiao Song, Kai Chen, Xiaoxiang Ren and **Haitao Yuan**, "Pedestrian Trajectory Prediction in Heterogeneous Traffic using Facial Keypoints- based Convolutional Encoder-decoder Network," *ACM Transactions on Internet Technology*, pp. 1–13, June 9, 2020, Minor Revision.

Yang Fu, Zeyu Zheng, MengChu Zhou, **Haitao Yuan** and Xiwang Guo, "New Results on Joint Classification of Vertical Structure of Ocean Properties at a Global Scale," *IEEE Access*, pp. 1–8, DOI: 10.1109/ACCESS.2020.2985527, Online, Mar. 2020.

Qingfeng Yao, Zeyu Zheng, Liang Qi, **Haitao Yuan**, Xiwang Guo, Ming Zhao, Zhi Liu, Tianji Yang, "Path Planning Method with Improved Artificial Potential FieldA Reinforcement Learning Perspective," *IEEE Access*, pp. 1–10, May 3, 2020, Submitted.

Conference papers:

Haitao Yuan, Jing Bi and MengChu Zhou, "Profit-Maximized Task Offloading with Simulated-annealing-based Migrating Birds Optimization in Hybrid Cloud-Edge Systems," *Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC 2020)*, Toronto, Ontario, Canada, 2020, pp. 1–6, Oct. 11–14, 2020, Submitted.

Haitao Yuan and Jing Bi, "Fine-grained Task Scheduling in Cloud Data Centers Using Simulated-annealing-based Bees Algorithm," *Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC 2020)*, Toronto, Ontario, Canada, 2020, pp. 1–6, Oct. 11–14, 2020, Submitted.

Haitao Yuan, Jing Bi and MengChu Zhou, "Spatio-Temporal Scheduling of Heterogeneous Delay-Constrained Tasks in Geo-Distributed Green Clouds," *Proceeding of IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, Banff, Canada, May 9-11, 2019, pp. 287–292, DOI: 10.1109/ICNSC.2019.8743294.

Haitao Yuan and MengChu Zhou, "Energy Cost and Performance-Sensitive Bi-objective Task Scheduling in Distributed Data Centers," *Proceeding of IEEE 17th International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, Oct. 2020, pp. 1–6, Accepted.

Shuang Li, Jing Bi, **Haitao Yuan** and MengChu Zhou, "BG-LSTM: Predictive Workload and Resource for Cloud Systems," *Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC 2020)*, Toronto, Ontario, Canada, Oct. 11-14, 2020, pp. 1–6, Submitted.

Heng Liu, Xiaofen Zhang, Jing Bi, **Haitao Yuan** and MengChu Zhou, "Intelligent Task Scheduling for Green Clouds with LSTM-based Prediction and Bi-objective Optimization," *Proceeding of IEEE 17th International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, Oct. 2020, pp. 1–6, Accepted.

Xiaodong Huang, Cheng Lu, Lei Peng, Jing Bi and **Haitao Yuan**, "A Deep Learning Approach to Large-Scale Light Curve Prediction and Real-Time Anomaly Detection with Grubbs Criterion," *Proceeding of IEEE 17th International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, Oct. 2020, pp. 1–6, Accepted.

Yongze Lin, Quanxi Dong, Jing Bi, **Haitao Yuan** and MengChu Zhou, "A Deep Learning Approach to Large-Scale Light Curve Prediction and Real-Time Anomaly Detection with Grubbs Criterion," *Proceeding of IEEE 17th International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, Oct. 2020, pp. 1–6, Accepted.

Jing Bi, Shuang Li, **Haitao Yuan**, Ziyan Zhao and Haoyue Liu, "Deep Neural Networks for Predicting Task Time Series in Cloud Computing Systems,"

Proceeding of IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, Canada, May 9-11, 2019, pp. 86–91, DOI: 10.1109/ICNSC.2019.8743188.

Patents:

Ahmed Chiheb Ammari, **Haitao Yuan**, Jing Bi, MengChu Zhou, Yusuf Al-Turki and Abdullah Abusorrah, "Temporal Task Scheduling in a Hybrid System," *United States Patent*, U.S. Patent No. 10528396, Issued Date: January 7, 2020.

Pending Patents:

Haitao Yuan, Jing Bi, MengChu Zhou, Ahmed Chiheb Ammari, Yusuf Al-Turki, Abdullah Abusorrah and Khaled Sadraoui, "Task Scheduling in a Green Data Center," *United States Patent*, U.S. Patent No. US20180329741A1.

This dissertation is dedicated to my inspiring parents, Shuying Yuan and Xiuqin Xu, for their unconditional love, trust, patience, understanding, support and efforts they gave me during the past years. I am also grateful to them, who let me understand the importance of education and the persistent pursuit of my dreams.

谨以此文献给我的父亲苑术营和母亲许秀芹，感谢他们在过去的岁月里给予我的无条件的爱、信任、耐心、理解、支持和付出。感激他们让我理解教育的重要性和对梦想的执着追求。

ACKNOWLEDGMENT

This has been a great journey. There are many people to whom I feel deeply indebted. First and foremost, my heartfelt thanks go to my wonderful advisor, Prof. MengChu Zhou. I did not know Prof. Zhou when I first came to New Jersey Institute of Technology. Only after several years that I did research under his excellent direction and learned about intelligent optimization, did I realize how fortunate I was to work with one of the most brilliant and world-famous experts in our field. He always accurately captures the nature of the problems, and provides me very insightful and high-level advices about the field. More importantly, Prof. Zhou is an extremely caring, supportive and kind advisor that I could not have requested for more. He gave me persistent encouragement and freedom to explore new research ideas while giving excellent guidance. His persistent patience and support helped me solve many difficult problems throughout my studies at the New Jersey Institute of Technology. This dissertation would not have been possible without his extensive knowledge, continuous support and encouragement. I hope that I will learn from his persistence in my own academic career.

I would further like to thank my co-advisor, Prof. Qing Liu, for his support and helpful comments throughout this Ph.D., especially in the design and implementation of hybrid meta-heuristic algorithms in this dissertation. Prof. Liu has 7-year working experiences in Scientific Data Group, Oak Ridge National Laboratory, and he has excellent research abilities in high-performance computing, big data in data-intensive science, and high-speed networking. He gave me very insightful directions to the research related to my Ph.D. dissertation.

I would like to acknowledge my committee members, Prof. Nirwan Ansari, Prof. Abdallah Khreishah, and Prof. Xinyue Ye for their time and helpful suggestions. In addition, Prof. Ye joined the Department of Landscape Architecture & Urban

Planning, Texas A&M University, College Station, TX, one week after he attended my Ph.D. oral defense. I hope that Prof. Ye will have an excellent academic career there, and I also believe that we will have more interactions and collaborations in my future academic career in US, China or any other countries in the world.

I also would like to express my special gratitude to Prof. Sotirios Ziavras, Prof. Leonid Tsybeskov, Prof. Durgamadhab Misra, Prof. Bipin Rajendran and Prof. Roberto Rojas-Cessa for their helpful guidance throughout my Ph.D. studies. In addition, Prof. Rajendran joined the King's College London, United Kingdom as a Reader in Engineering. I can still remember a challenging yet interesting research problem he asked in my Ph.D. qualifying examination. I am really fortunate to meet Prof. Rajendran in NJIT, and I hope that we will meet each other in the future, and discuss some important scientific international cooperation. Besides, I would like to express my special thanks to Prof. Jia Zhang from Department of Electrical and Computer Engineering, Carnegie Mellon University, Silicon Valley, CA for her strong recommendation for my Ph.D. studies. In addition, I also would like to express my special thanks to Prof. Ying (Gina) Tang from Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ for her help in my Ph.D. studies.

In addition, I am obliged to friends (and forgive me for not being able to list all of them): Jingchu Ji, Xiaoyu Lu, Haoyue Liu, Qi Kang, Jialun Xue, Jiankai Li, Ishani Chatterjee, Fatemeh Mohammadi Shakiba, Ziyan Zhao, Siya Yao, Meiji Cui, Dan You, Li Huang, Liang Qi, Xilong Liu, Shuai Zhang, and many others, who have given me encouragement, friendship and moral support over the recent years.

I also would like to express my thanks to visiting scholars: Prof. Jing Bi, Prof. Yunni Xia, Prof. Lianghua He, Prof. Zizhen Zhang, Prof. Shouguang Wang, Prof. Qinghua Zhu, Prof. Xiwang Guo and Prof. Bo Huang, who have given me valuable advices and suggestions.

I would like to extend my thanks to other faculty and staff members of the Helen and John C. Hartmann Department of Electrical and Computer Engineering for their dedicated assistance throughout my Ph.D. studies.

Lastly, I express my heartfelt gratitude to my parents (Shuying Yuan and Xiuqin Xu) for their concern and unconditional support. They deserve special thanks for their patience, caring, encouragement and understanding. I am very pleased to dedicate this dissertation to them.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Full Name
API	Application Programming Interface
App	Application
BA	Bat Algorithm
BeesA	Bees Algorithm
BAS	Bees Algorithm based on Simulated annealing
CAT	Cumulative Arriving Task
CDC	Cloud Data Center
CPU	Central Processing Unit
CSP	Cloud Service Provider
CST	Cumulative Scheduled Task
DE	Differential Evolution
EA	External Archive
EC2	Elastic Compute Cloud
FA	Firefly Algorithm
FCFS	First-Come-First-Serve
FIFO	First-In-First-Out
5G	The Fifth Generation
FiWi	Fiber-Wireless
FSTS	Fine-grained Spatial Task Scheduling
GA	Genetic Algorithm
GATS	Geography-Aware Task Scheduling
GL-PSO	Genetic Learning Particle Swarm Optimization
GPU	Graphics Processing Unit
GSPSO	Genetic Simulated-annealing-based Particle Swarm Optimization

GTB	Green Task Balancing
IMEAD	Improved Multi-objective Evolutionary Algorithm based on Decomposition
IoT	Internet of Things
ISP	Internet Service Provider
MBO	Migrating Birds Optimization
MBS	Macro Base Station
MDP	Markov Decision Process
MEC	Mobile Edge Computing
MINLP	Mixed Integer NonLinear Program
MIPS	Million Instructions Per Second
MODE	Multi-Objective Differential Evolutionary
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOPSO	Multi-Objective Particle Swarm Optimization
NSGA2	Nondominated Sorting Genetic Algorithm 2
PMA	Profit Maximization Approach
PQTS	Profit and QoS-optimized Task Scheduling
PSO	Particle Swarm Optimization
PUE	Power Usage Effectiveness
QoS	Quality of Service
SA	Simulated Annealing
SADE	Simulated-annealing-based Adaptive Differential Evolution
SBA	Simulated-annealing-based Bat Algorithm
SBDE	Simulated-annealing-based Bi-objective Differential Evolution
SBS	Small-cell Base Station
SLA	Service Level Agreement
SMBO	Simulated-annealing-based Migrating Birds Optimization

SMD	Smart Mobile Device
SOA	Swarm-based Optimization Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2
STLB	Spatio-Temporal Load Balancing
STSRO	Spatial Task Scheduling and Resource Optimization
STTS	Spatio-Temporal Task Scheduling
TTS	Temporal Task Scheduling
VM	Virtual Machine

LIST OF NOTATIONS

Notation	Definition
α	Power usage effectiveness of CDC
α_c	Power usage effectiveness of CDC c
$\beth_{\tau}^{v,n}$	Execution time of tasks of application n in public cloud v in τ
$\beth_{\tau+b}^{v,n}$	Execution time of tasks of application n in public cloud v in $\tau + b$
$\chi_{\tau}^{c,q}$	Clock frequency of server q in CDC c in time slot τ
$\chi_{\tau}^{c,n}$	Clock frequency of each server of application n in CDC c in time slot τ
$\beth_c$	Coefficient of activeness and workload for servers in CDC c
$\beth_c^n$	Coefficient of activeness and workload for servers of application n in CDC c
δ_{τ}	Task loss possibility of applications in time slot τ
$\delta_{\tau}^{c,n}$	Task loss possibility of application n in CDC c in time slot τ
δ_{τ}^{n}	Task loss possibility of application n in time slot τ
$\Delta_{\tau,c,n}^{1}$	Coefficient of maximum variance asymptotic proximity for application n in CDC c in τ
$\hat{\Delta}_{\tau,c,n}^{1}$	Intermediate notation for using perspective function properties of $\Delta_{\tau,c,n}^{1}$
$\Delta_{\tau,c,n}^{2}$	Reciprocal of normalized variance of arrival process for application n in CDC c in time slot τ
$\hat{\Delta}_{\tau,c,n}^{2}$	Intermediate notation for using perspective function properties of $\Delta_{\tau,c,n}^{2}$
$\Delta_{\tau,c}^{3}$	Difference between revenue and energy cost for CDC c in time slot τ
$\Delta_{\tau,c,n}^{3}$	Difference between revenue and energy cost for application n in CDC c in time slot τ
Δ^{4}	Difference between peak and idle power of each server
Δ_{n}^{4}	Difference between peak and idle power of each server for application n
$\Delta_{c,n}^{4}$	Difference between peak and idle power of each server for application n in CDC c
Δ^{5}	Sum of each server's idle power, and one consumed by non-computing facilities
Δ_{n}^{5}	Sum of each server's idle power, and one consumed by non-computing facilities for application n
$\Delta_{c,n}^{5}$	Sum of each server's idle power, and one consumed by non-computing facilities for application n in CDC c

$\Delta^{6}_{c,n}$	Lost rate of tasks for application n in CDC c
$\Delta^{6}_{\tau,c,n}$	Lost rate of tasks for application n in CDC c in time slot τ
$\hat{\Delta}^{6}_{\tau,c,n}$	Intermediate notation for using perspective function properties of $\Delta^{6}_{\tau,c,n}$
$\Delta^{7}_{\tau,c,n}$	Standard normal distribution function for application n in CDC c in time slot τ
$\Delta^{8}_{\tau,c,n}$	Ratio of Gaussian process variance and arrival rate of application n in CDC c in time slot τ
$\Delta^{9}_{\tau,c,n}$	Normalized variance of $\lambda^{c,n}_{\tau}$
Δ^{10}	Intermediate notation for proving convexity of optimization problem
$\Delta^{11}_{\tau,c,n}$	Task queue length of application n in CDC c in time slot τ
$\Delta^{12,c,n}_{\tau}$	Possibility that there are no tasks in queue of application n in CDC c in time slot τ
Δ^{13}_{τ}	Average task loss possibility in time slot τ
$\Delta^{14}_{\tau,c,q}$	Intermediate notation for obtaining response time of task q in CDC c in time slot τ
Δ^{15}	Transformed objective function
$\Delta^{16}_{\tau,c,n}$	Intermediate notation for obtaining response time of application n in CDC c in time slot τ
Δ^{17}_{τ}	Possibility that there are no tasks in queue in time slot τ
Δ^{18}_{τ}	Task queue length in time slot τ
$\Delta^{19}(t)$	Revenue or penalty of a task given its actual response time
ϵ^{n}_{τ}	Penalty cost of each rejected task of application n in time slot τ
$\Gamma_{\tau,c,n}$	Probability function autocovariance for task arriving rate of application n in CDC c in time slot τ
$\overset{0}{\gamma}_{1}$	Positive constant
$\overset{0}{\gamma}_{2}$	Positive constant
$\overset{0}{\gamma}_{3,c}$	Constant for CDC c
$\overset{0}{\gamma}_{4,c}$	Constant for CDC c
$\overset{0}{\gamma}_{5,c}$	Constant for CDC c
$\overset{0}{\gamma}_{6}$	Empirically derived correction constant
$\overset{0}{\gamma}_{7,c,n}$	Constant
$\overset{0}{\gamma}_{8,c,n}$	Constant for each server of application n in CDC c

$\overset{0}{\gamma}_{9,c,n}$	Constant for each server of application n in CDC c
$\beth(\boldsymbol{x})$	Objective function in single-objective optimization
$\hbar_q$	Processing speed of node q
$\hat{\hbar}_1$	Maximum amount of CPU resources in each node
$\hat{\hbar}_2$	Maximum amount of memory resources in each node
$\hat{\hbar}_3$	Maximum amount of bandwidth resources in each node
κ_τ^q	Load level of node q in time slot τ
$\bar{\kappa}_\tau$	Average load level of all nodes in edge computing in time slot τ
$\overset{+}{\lambda}_{\ell,n}$	Accumulated arriving rate of tasks of application n in time slot ℓ
$\overset{+}{\lambda}_{\ell,c,n}$	Accumulated task arriving rate of application n in CDC c in time slot ℓ
λ_τ	Task arriving rate in time slot τ
$\acute{\lambda}_\tau$	Arriving rate of tasks executed in CDC in time slot τ
$\tilde{\lambda}_\tau^c$	Task arriving rate of CDC c in time slot τ
$\lambda_\tau^{c,n}$	Task arriving rate of application n in CDC c in time slot τ
$\lambda_\tau^{c,q}$	Task arriving rate at server q in CDC c in time slot τ
λ_τ^n	Task arriving rate of application n in time slot τ
$\lambda_\tau^{\dagger,n}$	Arriving rate of remaining tasks of application n in time slot τ
$\lambda_\tau^{k,c,n}$	Task arriving rate of application n delivered to CDC c through ISP k in time slot τ
$\lambda_{\tau+b}^n$	Task arriving rate of application n in time slot τ+b
$\lambda_{\tau+b}^{c,n}$	Task arriving rate of application n in CDC c in time slot τ+b
$\lambda_{\tau+b}^{\dagger,c,n}$	Arriving rate of remaining tasks of application n in CDC c in time slot τ+b
$\overset{+}{\lambda}_{\tau,n}$	Accumulated arriving rate of tasks of application n in time slot τ
$\mho$	Penalty of all constraints
μ_τ	Task service rate of CDC servers in time slot τ
μ_τ^n	Task service rate of application n in time slot τ

$\mu_{\tau}^{c,n}$	Task service rate of application n in CDC c in time slot τ
$\mu_{\tau+b}^{c,n}$	Task service rate of application n in CDC c in time slot $\tau+b$
∇_n	Revenue of each task of application n
∇_{τ}^{n}	Revenue of each task of application n in time slot τ
$\nabla_{\tau+b}^{n}$	Revenue of each task of application n in time slot $\tau+b$
$\nu_{\tau}^{v,n}$	Price of VMs for application n in public cloud v in time slot τ
$\nu_{\tau+b}^{v,n}$	Price of VMs for application n in public cloud v in time slot $\tau+b$
ω_c	Loading capacitance in CDC c
ω_c^n	Loading capacitance of each server of application n in CDC c
$\breve{\Omega}$	External archive (EA)
$\lvert\breve{\Omega}\rvert$	Number of individuals in $\breve{\Omega}$
$\Omega_{\tau}^{c,n}$	Variation coefficient of $t_{\tau}^{c,n}$
$\text{ß}_{\tau}^{c,q}$	Variation coefficient of $\mathbb{t}_{\tau}^{c,q}$
$\hat{\partial}$	Upper limit of execution cost of each task executed in edge computing
$\check{\Phi}$	Idle power of each server in CDC
$\hat{\Phi}$	Peak power of each server in CDC
$\hat{\Phi}_n$	Peak power of each server for application n
$\check{\Phi}_n$	Idle power of each server for application n
$\check{\Phi}_n^c$	Idle power of each server of application n in CDC c
$\hat{\Phi}_n^c$	Peak power of each server of application n in CDC c
ϕ_1	Wind-to-electricity conversion rate of CDC
ϕ_{1c}	Wind-to-electricity conversion rate of CDC c
ϕ_2	On-site air density in CDC
ϕ_{2c}	On-site air density in CDC c
ϕ_3	Rotor area of wind turbines in CDC
ϕ_{3c}	Rotor area of wind turbines in CDC c

$\phi_{\tau,4c}$	Wind speed of CDC c in time slot τ
$\phi_{\tau+b,4c}$	Wind speed of CDC c in time slot $\tau+b$
$\check{\Phi}_q^c$	Power consumed by each idle server q in CDC c
ψ_1	Solar-irradiance-to-electricity conversion rate of CDC
ψ_{1c}	Solar-irradiance-to-electricity conversion rate of CDC c
ψ_{2c}	Active irradiance area of solar panels in CDC c
$\psi_{\tau,3c}$	Solar irradiance of CDC c in time slot τ
$\psi_{\tau+b,3c}$	Solar irradiance of CDC c in time slot $\tau+b$
ψ_2	Active irradiance area of solar panels in CDC
Ψ_c	Coefficient of converting clock frequency to supply voltage for servers in CDC c
Ψ_c^n	Coefficient of converting clock frequency to supply voltage for servers of application n in CDC c
$\rho_\tau^{c,n}$	Utilization of each server of application n in CDC c in time slot τ
$\rho_\tau^{c,q}$	Utilization of server q in CDC c in time slot τ
$\rho_\tau^{\ddagger}$	Utilization of each server in CDC in time slot τ
$\breve{\sigma}_{c,n}$	Variance of r_c^n
$\sigma_{\tau,c,n}$	Variance of Gaussian process for tasks of application n in CDC c
$\breve{\sigma}_{c,q}$	Variance of $\mathbb{r}_c^q$
$\hat{\sigma}_{\tau,c,n}$	Variance of $\eth_\tau^{c,n}$
$\tilde{\sigma}_{\tau,c,n}$	Variance of $t_\tau^{c,n}$
$\hat{\sigma}_{\tau,c,q}$	Variance of $\eth_\tau^{c,q}$
$\tilde{\sigma}_{\tau,c,q}$	Variance of $\mathbb{t}_\tau^{c,q}$
Θ	Intermediate notation for proving convexity of optimization problem
θ_1^i	Velocity of individual i
$\theta_1^{i,g}$	Velocity of individual i in iteration g
θ_2^0	Initial temperature
θ_2^g	Current temperature in iteration g

θ_3	Temperature cooling rate
θ_4	Number of grids in any dimension of the space
$\check{\boldsymbol{\theta}}_5$	Minimum value vector of elements
$\check{\boldsymbol{\theta}}_5^d$	Minimum value of the d-th element in $\check{\boldsymbol{\theta}}_5$
$\hat{\boldsymbol{\theta}}_5$	Maximum value vector of elements
$\hat{\boldsymbol{\theta}}_5^d$	Maximum value of the d-th element in $\hat{\boldsymbol{\theta}}_5$
$\hat{\theta}_6^\iota$	Maximum value of objective function ι
$\check{\theta}_6^\iota$	Minimum value of objective function ι
θ_7	Crossover possibility
$\hat{\theta}_7$	Maximum crossover possibility
$\check{\theta}_7$	Minimum crossover possibility
θ_8	Mutation possibility
$\hat{\theta}_8$	Maximum mutation possibility
$\check{\theta}_8$	Minimum mutation possibility
$\theta_{1,i}^A$	Entropy of candidate i in EA in SADE
$\theta_{2,i}^A$	Sparsity degree of candidate i in EA in SADE
$\theta_{3,i}^A$	Distance of candidate i to its lower adjacent one in EA in SADE
$\theta_{4,i,\iota}^A$	Lower adjacent one of candidate i in EA along objective ι in SADE
θ_5^A	Diversity of the Pareto-optimal front in SADE
$\theta_{1,i}^B$	Frequency of bat i in SBA
$\check{\theta}_1^B$	Minimum frequency in SBA
$\hat{\theta}_1^B$	Maximum frequency in SBA
$\theta_{2,i}^B$	Pulse rate of bat i in SBA
$\theta_{2,i,g}^B$	Pulse rate of bat i at iteration g in SBA
$\theta_{2,i}^B$	Initial pulse rate of bat i in SBA
$\check{\theta}_2^B$	Minimum pulse rate in SBA

$\hat{\theta}_2^B$	Maximum pulse rate in SBA
θ_3^B	Increase rate of pulse rate in SBA
$\theta_{4,i}^B$	Loudness of bat i in SBA
$\theta_{4,i,g}^B$	Loudness of bat i at iteration g in SBA
$\check{\theta}_4^B$	Minimum loudness in SBA
$\bar{\theta}_{4,g}^B$	Average loudness of all bats at iteration g in SBA
$\hat{\theta}_4^B$	Maximum loudness in SBA
θ_5^B	Loudness reduction rate in SBA
$\theta_{6,i}^B$	Adaptation value of bat i in SBA
θ_1^D	Scaling factor in SBDE
θ_2^D	Positive constant in SBDE
$\theta_{3,\iota}^D$	Range of objective function ι in SBDE
$\check{\theta}_{4,\iota}^D$	Lower bound of grid region for objective function ι in SBDE
$\hat{\theta}_{4,\iota}^D$	Higher bound of grid region for objective function ι in SBDE
$\theta_{1,i}^M$	Neighboring area of individual i in IMEAD
$\theta_{2,i}^M$	Intermediate notation for selecting a new individual from $\theta_{1,i}^M$ in IMEAD
$\boldsymbol{\theta}_3^M$	Variance vector in IMEAD
$\boldsymbol{\theta}_4^M$	Binary vector in IMEAD
θ_5^M	Intermediate notation for producing a new solution around a new individual in IMEAD
$\boldsymbol{\theta}_6^M$	Vector of the best objective function values obtained in IMEAD
$\boldsymbol{\theta}_{6,\iota}^M$	The best value of objective function ι obtained in IMEAD
$\boldsymbol{\theta}_{7,i}^M$	Weight vector of objective functions of individual i in IMEAD
$\boldsymbol{\theta}_{7,i,\iota}^M$	Weight of objective function ι of individual i in IMEAD
$\theta_{8,i}^M$	Crowded distance of individual i in IMEAD
$\acute{\theta}_{9,i}^M$	Next individual neighboring to individual i in IMEAD
$\grave{\theta}_{9,i}^M$	Previous individual neighboring to individual i in IMEAD

$\check{\theta}_1^P$	Coefficient of individual acceleration reflecting influence of $\check{\mathbf{x}}_i$ in GSPSO
$\hat{\theta}_1^P$	Coefficient of social acceleration reflecting influence of $\hat{\mathbf{x}}$ in GSPSO
θ_2^P	Superior acceleration coefficient in GSPSO
θ_3^P	Maximum velocity of particles in GSPSO
θ_4^P	Percentage of particles with the same fitness values in GSPSO
$\hat{\theta}_4^P$	Specified maximum percentage in GSPSO
θ_5^P	Inertia weight in GSPSO
$\check{\theta}_5^P$	Lower bound of inertia weight w in GSPSO
$\hat{\theta}_5^P$	Upper bound of inertia weight w in GSPSO
θ_1^S	Neighborhood radius of $\overset{o}{N}_{\tau,c,n}$ in BAS
$\acute{\theta}_{1,c,n}^S$	Neighborhood radius of $\lambda_\tau^{c,n}$ in BAS
$\tilde{\theta}_{1,c,n}^S$	Neighborhood radius of $\varrho_\tau^{c,n}$ in BAS
θ_2^S	Reduction rate of neighborhood radius in BAS
$\triangle_\tau^n$	Energy cost of each task of application n in time slot τ
υ_τ	Total transmission time of input/output data to/from CDC through MBS
Υ	Υ=$max_{n\in\{1,2,\cdots,N\}}(B_n)$
$\hat{\varepsilon}$	Specified maximum load balance level
ε_τ	Load balance level of all nodes in edge computing in time slot τ
$\varkappa_1^0$	Weight constant
$\varkappa_2^0$	Weight constant
$\varkappa_3^0$	Weight constant
$\varphi_\tau^{1,q}$	Utilization of CPU resources of node q in time slot τ
$\varphi_\tau^{2,q}$	Utilization of memory resources of node q in time slot τ
$\varphi_\tau^{3,q}$	Utilization of bandwidth resources of node q in time slot τ
$\hat{\varrho}_c$	Maximum running speed of each server in CDC c
$\hat{\varrho}_c^n$	Maximum running speed of each server of application n in CDC c

$\varrho_{\tau}^{c,n}$	Running speed of each server of application n in CDC c in time slot τ
$\varrho_{\tau}^{c,q}$	Running speed of server q in CDC c in time slot τ
ς_1^s	Number of CPU resources of task s
ς_2^s	Number of memory resources of task s
ς_3^s	Number of bandwidth resources of task s
ϑ_c	Activity factor in CDC c
ϑ_c^n	Activity factor of each server of application n in CDC c
Λ_i^g	Difference between $\mathbf{x}_i^g$ and $\grave{\mathbf{x}}_i^g$
ζ_n	Size of each task of application n
B_n	Delay bound of application n
b_{τ}^k	Unit bandwidth price of ISP k in time slot τ
$\hat{B}_k$	Bandwidth limit of ISP k
$\hat{\mathbb{E}}$	Maximum amount of energy in CDC
E_{τ}	Total energy consumed in time slot τ
$\hat{\mathbb{E}}_c$	Maximum amount of energy in CDC c
$\widetilde{E}_{\tau}$	Wind energy produced in time slot τ
$\overset{\circ}{E}_{\tau}$	Solar energy produced in time slot τ
E_{τ}^c	Total energy consumed by CDC c in time slot τ
$\widetilde{E}_{\tau,c}$	Wind energy consumed by CDC c in time slot τ
$\overset{\circ}{E}_{\tau,c}$	Solar energy produced by CDC c in time slot τ
$\overset{+}{E}_{\tau,c}$	Total green energy consumed by CDC c in time slot τ
$E_{\tau}^{c,n}$	Total energy consumed by application n in CDC c in time slot τ
$E_{\tau+b}$	Total energy consumed in time slot $\tau+b$
$\overset{\circ}{E}_{\tau+b}$	Solar energy produced in time slot $\tau+b$
$\widetilde{E}_{\tau+b}$	Wind energy produced in time slot $\tau+b$
$E_{\tau+b}^c$	Total energy consumed by CDC c in time slot $\tau+b$

$\overset{\circ}{E}_{\tau+b,c}$	Solar energy produced by CDC c in time slot $\tau+b$
$\widetilde{E}_{\tau+b,c}$	Wind energy produced by CDC c in time slot $\tau+b$
f_1	Total revenue of tasks
$f_1^{c,*}$	Revenue of tasks of all applications in CDC c
$f_1^{*,n}$	Revenue of tasks of application n in all CDCs
$f_{1,\tau}^{*,n}$	Revenue of tasks of application n in all CDCs in time slot τ
$f_{1,\tau+b}^{*,n}$	Revenue of tasks of application n in all CDCs in time slot $\tau+b$
f_2	Total cost of CDC provider
$\widetilde{f}_2$	Augmented objective function of f_2
f_{21}	ISP bandwidth cost of data transmission among users and CDCs
f_{22}	Energy cost of tasks of all applications
$\widetilde{f_{22}}$	Augmented objective function of f_{22}
$f_{22}^{c,*}$	Energy cost of CDC c
f_{23}	Execution cost of all tasks in edge computing
$f_{23}^{s,q}$	Execution cost of task s executed in node q in edge computing
$\widetilde{f}_\iota$	Augmented objective function ι ($\iota \in \{1, 2, \ldots, \overset{o}{M}\}$)
$\bar{\mathcal{F}}$	Average fitness value for all individuals
$\mathcal{F}_i$	Fitness of individual i
$\tilde{\mathcal{F}}_i$	Absolute value of difference between $\mathcal{F}^i$ and $\bar{\mathcal{F}}$
$\acute{\mathcal{F}}_i$	New fitness for individual i
F_1	Profit of CDC provider
$\widetilde{F_1}$	Augmented objective function of F_1
F_2	Task loss possibility of CDC
$\widetilde{F_2}$	Augmented objective function of F_2
g	Current iteration number
$\hat{g}$	Total number of iterations

$g_l(\boldsymbol{x})$	Inequality constraint z $(1 \leq l \leq \mathcal{N}^{\neq})$
$h_m(\boldsymbol{x})$	Equality constraint y $(1 \leq m \leq \mathbb{N}^{=})$
$\overset{\bullet}{i}$	Elitist individual with the minimum Euclidean distance from individual i
$\overset{\longleftrightarrow}{i,j}$	Euclidean distance between individual i and elitist individual j in $\hat{\Omega}$
L	Length of each time slot
$\acute{M}$	Number of fittest sites in BSA
$\bar{M}$	Number of recruited bees for each non-elite site in BSA
$\check{M}$	Number of recruited bees for each elite site in BSA
$\grave{M}$	Number of elite sites in BSA
$\tilde{M}$	Number of neighbor solutions for each solution
$\overset{o}{M}$	Number of objective functions
M^0	Number of unused neighbor solutions of current solution in SMBO
$\hat{N}$	Maximum number of servers in CDC
$\overset{\bullet}{\mathbb{N}}$	Number of tasks processed by each switched-on server per time in the CDC layer
$\mathcal{N}^A$	Number of applications
$\hat{N}_c$	Number of heterogeneous servers in CDC c
$\mathcal{N}^C$	Number of CDCs
$\hat{N}_{c,n}$	Maximum number of servers for application n in CDC c
$\overset{\bullet}{\mathbb{N}}_{c,n}$	Number of tasks executed by each switched-on server for application n in each minute in CDC c
$\mathbb{N}^D$	Number of decision variables
$\hat{\mathbb{N}}^{\ddagger}$	Maximum number of tasks executed by CDC
$\overset{\#}{N}_{\ell,v,n}$	Number of tasks of application n scheduled to public cloud v in time slot ℓ
$\mathbb{N}^{=}$	Number of equality constraints
$\overset{\infty}{\mathcal{N}}$	Large positive constant
$\mathcal{N}^K$	Number of ISPs
$\hat{N}_n$	Number of servers for application n

$\overset{\bullet}{\mathbb{N}}_n$	Number of tasks executed by each switched-on server for application n in each minute
$\mathbb{N}_n^-$	Maximum number of tasks dropped at task queue of application n
$\mathcal{N}^{\neq}$	Number of inequality constraints
$\mathcal{N}_s$	Size of task s (MIPS/second)
$\mathcal{N}^*$	Total number of time slots
$\overset{o}{N}_{\tau,c}$	Number of switched-on servers in CDC c in time slot τ
$\overset{o}{N}_{\tau,c,n}$	Number of switched-on servers for application n in CDC c in time slot τ
$\overset{\diamond}{\mathbb{N}}_{\tau,n}$	Number of tasks of application n executed by time slot τ
$\overset{o}{N}_{\tau,n}$	Number of switched-on servers of application n in time slot τ
$\overset{\diamond}{\mathbb{N}}_{\tau,c,n}$	Accumulated number of executed tasks of application n in CDC c by time slot τ
$\overset{+}{\mathbb{N}}_{\tau,c,n}$	Accumulated number of arriving tasks of application n in CDC c by time slot τ
$\overset{o}{N}_{\tau+b,c,n}$	Number of switched-on servers of application n in CDC c in time slot τ+b
$N_\tau^\dagger$	Number of tasks scheduled to nodes in edge computing in time slot τ
$\overset{+}{\mathbb{N}}_{\tau,n}$	Accumulated number of arriving tasks of application n in all CDCs by time slot τ
$\overset{o}{N}_{\tau+b,n}$	Number of switched-on servers of application n in time slot τ+b
N_τ^q	Maximum number of tasks that can be executed by node q in edge computing in time slot τ
$\overset{\#}{N}_{\tau,v,n}$	Number of tasks of application n scheduled to public cloud v in time slot τ
$\mathcal{N}^V$	Number of public clouds
N^0	Number of heterogeneous nodes in edge computing
$\mathbb{p}_\tau$	Price of power grid in time slot τ
$\mathbb{p}_{\tau+b}$	Price of power grid in time slot τ+b
$\tilde{P}_\tau$	Total power consumption in time slot τ
p_τ^c	Price of power grid of CDC c in time slot τ
P_τ^c	Total power consumption of CDC c in time slot τ
$\overset{\circ}{P}_{\tau,c}$	Solar power consumed by CDC c in time slot τ
$P_\tau^{c,n}$	Power consumed by each server of application n in CDC c in time slot τ

$\mathbb{P}_{\tau}^{c,q}$	Power consumed by server q in CDC c in time slot τ
$p_{\tau+b}^{c}$	Price of power grid in CDC c in time slot τ+b
$P_{\tau+b}$	Total power consumption in time slot τ+b
$\widetilde{P}_{\tau,c}$	Wind power consumed by CDC c in time slot τ
$\hat{\mathbb{Q}}_{n}$	Capacity limit of task queue of application n
$\hat{Q}_{n}^{c}$	Capacity limit of task queue of application n in CDC c
r_{c}^{n}	Size of each task of application n in CDC c
$\bar{r}_{c}^{n}$	Mean of r_{c}^{n}
$\bar{r}_{c}^{q}$	Mean of $\mathbb{r}_{c}^{q}$
$\mathbb{r}_{c}^{q}$	Size of each task scheduled to server q in CDC c
t	Response time of each task
$t_{\tau}^{c,n}$	Running time of each task of application n in CDC c in time slot τ
$\mathbb{t}_{\tau}^{c,q}$	Running time of each task on server q in CDC c in time slot τ
$\bar{t}_{\tau}^{c,n}$	Mean of $t_{\tau}^{c,n}$
$\bar{\mathbb{t}}_{\tau}^{c,q}$	Mean of $\mathbb{t}_{\tau}^{c,q}$
$\hat{T}^{\dagger}$	Response time limit of tasks executed in edge computing
$\hat{T}^{\ddagger}$	Response time limit of tasks executed in CDC
$\hat{T}_{n}$	Response time limit of application n
T_{τ}	Response time of each task in time slot τ
$\widetilde{T}_{\tau}$	Augmented objective function of T_{τ}
T_{τ}^{c}	Response time of tasks in CDC c
$\mathcal{T}_{\tau}^{c,n}$	Interarrival time for each server of application n in CDC c in time slot τ
$\bar{\mathcal{T}}_{\tau}^{c,n}$	Mean of $\mathcal{T}_{\tau}^{c,n}$
$T_{\tau}^{c,n}$	Response time of tasks of application n in CDC c in time slot τ
$\mathbb{T}_{\tau}^{c,q}$	Response time of tasks in server q in CDC c in time slot τ
$\mathbb{T}_{\tau}^{c,q}$	Interarrival time for server q in CDC c in time slot τ

$\bar{\mathbb{T}}_{\tau}^{c,q}$	Mean of $\mathbb{T}_{\tau}^{c,q}$
$\mathbb{T}_{\tau}^{\dagger}$	Maximum response time of tasks executed in edge computing in time slot τ
$T_{\tau}^{\ddagger}$	Response time of each task executed in CDC in time slot τ
$T_{\tau}^{s,q}$	Execution time of task s on node q in edge computing
u	Server CPU utilization
u_{τ}^{n}	CPU utilization in time slot τ
$u_{\tau}^{c,n}$	CPU utilization of servers of application n in CDC c in time slot τ
$u_{\tau+b}^{c,n}$	CPU utilization of servers of application n in CDC c in time slot τ+b
$\mathbb{U}_{\tau}^{c,q}$	Supply voltage of server q in CDC c
$U_{\tau}^{c,n}$	Supply voltage of each server of application n in CDC c in time slot τ
w_1–w_{24}	A random number or vector
$W_{\tau}^{c,n}$	Waiting time of tasks of application n in each server in CDC c
$\mathbb{W}_{\tau}^{c,q}$	Waiting time of tasks in server q in CDC c
$\boldsymbol{x}$	Vector of decision variables
$\overset{*}{\mathbf{x}}$	Optimal position/solution finally produced
$\hat{\mathbf{x}}$	Globally optimal position/solution of current population
$\tilde{\mathbf{x}}^{B}$	Alternative optimal position/solution in SBA
$\mathbf{x}_i^{B}$	Position/solution from previous positions/solutions of individual i in SBA
$\overset{\bullet}{\mathbf{x}}_g$	Position/solution of the optimal individual in iteration g
$\mathbf{x}_i$	Position/solution of each individual i
$\check{\mathbf{x}}_i$	Locally optimal position/solution of individual i
$\overset{0}{\mathbf{x}}_{i,d}$	Initial value of decision variable d of individual i
$\mathbf{x}_{i,d}$	Value of the d-th element of individual i
$\mathbf{x}_{i,d}^{g}$	Value of the d-th element of $\mathbf{x}_i^{g}$
$\grave{\mathbf{x}}_{i,d}^{g}$	Value of the d-th element of $\grave{\mathbf{x}}_{i,d}^{g}$
$\acute{\mathbf{x}}_{i,d}^{g}$	Value of the d-th element of $\acute{\mathbf{x}}_i^{g}$

$\acute{\mathbf{x}}_i^g$	Position/solution of a new candidate for individual i in iteration g
$\mathbf{x}_i^g$	Position/solution of individual i in iteration g
$\grave{\mathbf{x}}_i^g$	Position/solution of a new mutant individual for $\mathbf{x}_i^g$
$\mathbf{x}_i^1$	Position/solution of a new individual for individual i after crossover operation
$\acute{\mathbf{x}}_i^P$	Position/solution of a superior particle corresponding to particle i in GSPSO
$\breve{\mathbf{x}}_i^P$	Position/solution of an offspring of particle i in GSPSO
$\mathbb{X}$	Population
$\|\mathbb{X}\|$	Size of population
$\overset{*}{\mathbb{X}}$	Set of the best individuals obtained
$y_\tau^{s,q}$	Binary variable denoting whether task s is scheduled to node q in time slot τ
$z_\tau^{v,n}$	Binary variable denoting whether tasks of application n are scheduled to public cloud v in time slot τ

CHAPTER 1

INTRODUCTION

Cloud computing and edge computing are emerging as two primary computing paradiams that are researched, developed and deployed by enterprises, governments, and academic institutes in recent years [1, 2, 3, 4]. By supporting a pay-as-you-go service model, cloud computing removes initial capital, maintenance and software licensing cost. In addition, it has greatly changed the way information technology infrastructure is provided to satisfy various business needs by enabling on-demand infrastructure provisioning [5, 6]. It is implemented in green cloud data centers (CDCs) that manage a great number of large-scale infrastructures [7] typically including millions of servers and cooling facilities [8]. The infrastructure resources in CDCs are shared to concurrently support multiple applications that flexibly deliver services to users around the world.

An increasing number of users deploy their delay-constrained applications, *e.g.*, search engine, scientific computing, distributed file systems, big data processing, deep learning [9, 10] and high-performance computing in data centers. This significantly increases the amount of energy consumed by large-scale data centers. Recent data shows that the energy consumed by data centers in U.S. is roughly 78 billion KWH in 2017, and it accounts for about 2.9% of the total energy consumed in U.S. [11]. In the U.S., CDCs are expected to consume 101.3 billion KWH annually by 2020, equivalent to the output of 50 large power plants. In 2 or 3 years, about 95% of urban data centers would experience total or partial outages that incur annual cost of roughly 2 million U.S$ per infrastructure [12]. Thus, the energy optimization has become a major concern in their server provisioning and cooling systems. To provide high availability and low latency, each application is usually deployed in multiple CDCs

located in different geographical locations [13]. In addition, considering performance and cost, each CDC is connected to multiple Internet service providers (ISPs) that transmit gigantic data among distributed CDCs and millions of users [14]. It is shown that ISP bandwidth and energy cost account for a majority of a CDC provider's operational expense [15].

1.1 Cloud Computing

There have been studies from both industries and academia on energy optimization problems [16]. Users' tasks must first traverse through the wide-area network including multiple ISPs before they can reach back-end CDCs. For example, Google's wide-area network delivers multiple applications, *e.g.*, video, search and mail, to global users [17]. CDCs also need to pay money to ISPs due to users' tasks and response data transmitted among users and CDCs [18]. Currently, typical CDCs transmit more than a petabyte data each day, and therefore, they suffer from huge ISP bandwidth cost [19]. In addition, the bandwidth price of each ISP is determined by a service-level agreement (SLA) [20] specified between users and a CDC provider, and therefore, bandwidth prices of some ISPs are much more expensive than others. However, existing studies do not consider the diversity in bandwidth prices of ISPs, and therefore, may lead to high cost. In addition, CDCs are usually located in different sites where the prices of power grid also exhibit spatial diversity [21]. The wind speed, solar irradiance, on-site air density, the maximum available energy, and the number of servers in each CDC all vary with geographical locations [22]–[24]. Thus, it becomes a big challenge to minimize the total cost of a CDC provider in a market where ISP bandwidth prices, power grid prices, and availability of renewable green energy all show their spatial diversity.

Similar to the work [25], this dissertation work investigates task scheduling of multiple long-delay applications whose delay bound constraints are relatively long,

e.g., high-performance simulation and large-scale data analysis. During the delay bound constraints of tasks, such factors as prices of power grid, solar irradiance and wind speed change with time [25]. In addition, each application is deployed in distributed green CDCs [26] to improve availability and robustness. Thus, tasks of each application can be independently executed within each CDC. Similarly, prices of power grid and availability of renewable energy in CDCs located in different geographical sites have spatial variations, *i.e.*, they vary with locations. CDCs always aim to schedule tasks in the most cost-effective way while meeting their delay bound constraints [27]. However, spatial and temporal variations make it highly challenging and essential to minimize energy cost of CDC providers by smartly scheduling tasks of heterogeneous applications among multiple CDCs while satisfying delay bound constraints of all tasks.

In addition, in 2017, over 80% of energy in U.S. was produced by burning nonrenewable fossil fuel, *e.g.*, coal, petroleum and natural gas. This brings the irreversible harm and pollution to the global environment. In addition, the pressure from governments worldwide is also increasing for reducing carbon footprints that significantly affect the global climate change. For example, Japan establishes the data center council to reduce the soaring energy consumption of data centers [28]. Therefore, a growing number of CDCs, *e.g.*, Microsoft, Google, Amazon, Alibaba and Apple [29], install renewable energy facilities to reduce the environmental pollution caused by the usage of fossil fuel, and migrate to greener CDCs [30]. Current CDCs are mainly powered by three energy sources, *i.e.*, power grid, solar energy and wind energy, and they aim to reduce the brown energy consumption by using renewable devices [31]. The aperiodic arrival of tasks makes it difficult to accurately predict task arriving rate of each application [32]. Thus, it is impossible to execute all tasks of each application with limited resources of each CDC at peak time. For example, when Apple's new iPhones are released, more than two million pre-orders are sent to its

data centers in the first 24 hours, and the Apple Store in some areas is unresponsive [33]. The response time of the CDC may be long and application crash may happen at peak time. Thus, a hybrid cloud scheme is increasingly deployed by CDC to handle peak tasks by outsourcing some tasks to public clouds. Virtual machines (VMs) are created to support applications in public clouds and the operation cost of CDC can be reduced. Currently, more than 82% of companies choose hybrid CDC to deploy their applications [33]. In hybrid CDC, a private CDC provider (called the CDC for short) needs to pay the execution cost of VMs due to the tasks scheduled to public clouds. Within delay bound constraints of long-delay applications, several factors in the CDC and public clouds show temporal variations [21]. Specifically, revenue, prices of power grid, solar irradiance, and wind speed in the CDC all vary with time. Besides, prices of VMs in public clouds also vary with time. Thus, the temporal variations in the factors make it challenging to schedule tasks of each application among the CDC and public clouds in a cost-effective way while meeting their delay bound constraints.

The dramatic growth in the number of arriving tasks significantly brings the energy cost of billions of dollars to their providers [28]. Besides the economic concern, the carbon footprint is significantly increasing and it is expected to exceed the airline industry emissions by 2020. According to the work in [34], CDCs consumed about 2.2% of total U.S. electricity consumption, and originated more than 43 million tons of CO_2 annually. Each large-scale green cloud usually needs as much energy as 25,000 households on average. Therefore, a growing number of enterprises install green energy infrastructure and build CDCs [35] to improve their energy efficiency. Several methods, *e.g.*, disk management, and dynamic voltage and frequency scaling [36], have been proposed to realize them. There are two types of ways to decrease the energy cost: turning off computing servers or decreasing tasks' performance. It is beneficial to reduce the energy consumption for CDCs. Yet it may not maximize the profit of providers and often deteriorates quality of service (QoS) requirements of

tasks. The reason is that applications and their data are increasing so quickly that the running of higher-performance servers requires more energy to efficiently execute users' tasks. According to the items in SLAs [37] signed between users and providers, the execution of tasks of each application contributes revenue to CDC providers. Therefore, the reduction of energy consumption might deteriorate QoS of tasks of users. Any QoS violations can significantly bring the penalty to a CDC provider because users have strict performance requirements of their applications, and then increase its total cost or decrease its profit. Consequently, the profit of providers, and QoS of tasks need to be jointly considered and optimized by intelligently scheduling tasks and allocating infrastructure resources.

1.2 Edge Computing

With the fast development of information and communication technologies, smart mobile devices (SMDs) have been gaining enormous attention with current mobile technologies, *e.g.*, smart phones, wearable devices , tablets and IoTs [38, 39, 40] as they enable convenient communications almost anywhere and anytime. SMDs accelerate the emergence of Internet of Things (IoTs) and drastically trigger a big revolution and rapid development of computationally intensive mobile applications [41, 42], *e.g.*, autonomous driving, interactive online games, speech/face recognition, augmented/assisted/virtual reality, social networking, traffic management, path planning and health monitoring. The imbalance and gap between resource-constrained devices and complex compute-intensive applications is increasingly becoming a bottleneck for enabling user-specified quality of experience, and therefore, may hinder the development of mobile applications [43]. Supported by embedded sensors and on-device cameras, mobile applications evolve with several new features, *e.g.*, face recognition, interactive online gaming and navigation, are increasingly developed. In addition, the era of IoTs is predicted to produce an dramatic amount of raw data

from IoT devices. The data contains important and valuable insights through big data analysis. However, its analysis needs extremely intensive data computation that exceeds the processing, storage as well as battery energy capacities of IoT devices [44, 45], and it leads to significant delay if the computation is pushed to the CDCs.

In traditional remote cloud computing systems, public clouds, *e.g.*, Google Cloud Platform, Microsoft Azure, and Amazon Web Services [46] are adopted, and therefore, incur long latency because of data exchange among them and SMDs with constrained resources in wide area networks. Consequently, it is highly needed to extend services of cloud computing to locations where data is produced, *i.e.*, the network edge [47]. To solve this problem, mobile edge computing (MEC) provides cloud computing functions and enables a promising computation paradigm. It migrates the computation, storage, and servicing capabilities to the network edge, and pulls SMDs out of heavy computation tasks by deploying applications locally with short distance in a distributed manner. [48]. Figure 1.1 illustrates a typical multi-layer edge computing architecture. It has several benefits including high bandwidth, location awareness, real time radio network information, *etc.* Therefore, it is able to significantly decrease latency, reduce congestion of data transmission and prolong the lifetime of battery of SMDs by intelligently and cost-effectively offloading part or all of compute-intensive latency-critical tasks from ubiquitous IoT devices/SMDs to physically close MEC servers. It is viewed as one of key technologies for the fifth generation (5G) networks by the European 5G Infrastructure Public Private Partnership. Therefore, a growing number of recent studies have been proposed from both academia [49] and industries [50] to take advantage of MEC features to decrease traffic overhead of real-life various industrial applications and execute users' intensive workload through offloading.

On the other hand, though computation offloading can effectively utilize powerful computational resources at cloud servers, it is still difficult to guarantee

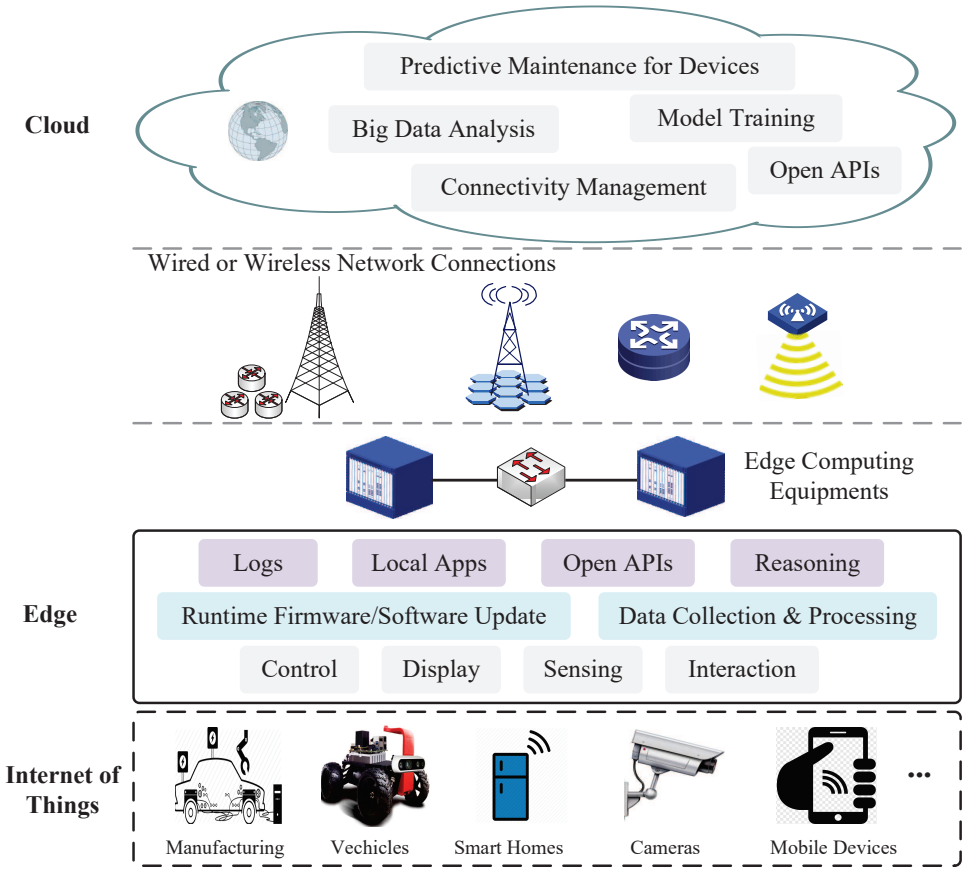


Figure 1.1 Multi-layer edge computing architecture.

the computation performance for traditional battery-powered devices due to limited battery energy for task offloading. For example, mobile applications have to be terminated and the battery energy of SMDs runs out. It might possibly be solved by regularly recharging the batteries or adopting larger batteries. However, hardware cost is increased if larger batteries are installed at SMDs, and it is not desirable. In addition, frequently recharging batteries might be impossible in some application scenarios, *e.g.*, in wireless sensor networks and IoTs for surveillance because the devices are usually difficult-to-reach. Thus, the increasing amount of energy consumption of IoTs brings a strong need for novel computation offloading methods dedicated for distributed cloud and edge computing systems with battery-powered devices. It is challenging to execute them in SMDs that own limited computation

resources and battery capacities. It is shown that CDCs, *e.g.*, Amazon Elastic Compute Cloud (Amazon EC2), and Microsoft Azure, provide new distributed computing to efficiently tackle the limitation of battery and processing capabilities by offloading part or all computation-intensive tasks to CDCs for execution [51]. In 2019, over 70% of calculations have been completed in CDCs. The applications send their tasks to remote CDCs that typically have rich computation resources, high security and huge storage [52, 53]. However, CDCs are usually far away from SMDs, and this leads to unacceptable transmission delay and economic cost for utilizing resources in CDCs [54], and affects real-time performance of latency-sensitive applications.

To tackle the shortcomings of CDCs, edge computing is an emerging architecture for the network edge, and it provides agile and pervasive resources to IoT applications with strict latency need at anytime and anywhere [55]. The need for real-time and scalable data analysis in IoT devices is a major driving power for edge computing where data is generated and processed in the network edge in many applications of smart city, smart home, surveillance networks, connected vehicles, industrial IoT, *etc.* Compared with powerful computing platforms in CDCs, edge computing enables higher computing agility and lower latency. It is shown that about 40% of IoT-produced data is stored and processed in the edge of a network [56]. Local computing in the edge greatly reduces the response time because waiting or communication delay between the edge and CDCs is avoided. Yet, the limits of energy, computation, and storage resources of nodes in IoT devices restrict the number of tasks of resource-hungry applications locally computed in the edge. Therefore, it is unlikely to execute all tasks in nodes in edge computing (*i.e.*, local computing), and some of them have to be offloaded to CDCs to avoid energy depletion and the overall system performance [57]. Thus, it is critically important to rationally schedule all tasks between CDC and edge computing layers, and maximize the profit of distributed cloud and edge

computing systems while ensuring that user-specific response time limits of tasks are well met.

1.3 Organization of This Dissertation

This dissertation work focuses on these challenging problems, and aims to propose several intelligent optimization methods to solve them. It is worth noting that Chapters 3–6 are the background work of this dissertation. The major contributions of this dissertation include Chapters 7–11.

1) Chapter 3 in the background work provides a Spatial Task Scheduling and Resource Optimization (STSRO) method to minimize the total cost of a CDC provider by exploiting the spatial diversity of ISP bandwidth and energy cost while strictly meeting delay bound constraints of tasks of all applications. This dissertation work incorporates the spatial diversity in above factors into a constrained optimization problem, and solves it with the proposed simulated-annealing-based bat algorithm to offer a real-time near-optimal solution. It jointly specifies the optimal allocation of all arriving tasks among multiple ISPs, and determines the optimal setting of each server in each CDC. It considers the variations of many factors including power grid price, wind speed, solar irradiance and on-site air density, and can intelligently schedule all arriving tasks to CDCs within their delay bound constraints.

2) Chapter 4 in the background work provides a geography-aware task scheduling (GATS) approach to achieve the profit maximization for CDC providers. Specifically, Chapter 4 adopts an accurate queue model, *i.e.*, $G/D/1$ to analyze the performance of each CDC. The arriving process of tasks is modeled by a random process with the general probability distribution. GATS jointly investigates and utilizes spatial differences of several factors including ISP bandwidth prices, the prices of grid, solar irradiance, active irradiance area of solar panels, wind speed, on-site air density, rotor area of wind turbines, and the maximum available number of servers in each CDC. This Chapter incorporates spatial differences into a constrained optimization problem that is proven to be a typical convex optimization one. Then, it is directly solved with the interior point method to offer the optimal task scheduling strategy that maximizes the profit of CDC providers by jointly and optimally determining the allocation of tasks of all applications among multiple ISPs and task service rates of servers in each CDC.

3) Chapter 5 in the background work minimizes energy cost for a CDC provider by jointly investigating spatial and temporal variations in prices of power grid and availability of renewable energy while meeting delay bound constraints

of all tasks strictly. In Chapter 5, an energy cost minimization problem is formulated and solved with a hybrid meta-heuristic algorithm named Genetic Simulated-annealing-based Particle Swarm Optimization (GSPSO). GSPSO combines genetic operations of genetic algorithm [58], Metropolis acceptance rule of simulated annealing [59], and social learning of particle swarm optimization [60]. According to the solution obtained by GSPSO, the proposed spatio-temporal task scheduling (STTS) method can jointly determine the optimal split of all tasks among CDCs, and also specify the optimal setting of servers in each CDC in each time slot within tasks' delay bound constraints.

4) Chapter 6 in the background work proposes a temporal task scheduling (TTS) algorithm in the hybrid CDC to consider the temporal variations in prices of power grid, revenue and green energy in the CDC, and prices of VMs in public clouds within tasks' delay bound constraints. Specifically, the profit maximization problem of the hybrid CDC is formulated and solved with GSPSO that is proposed to increase both efficiency and global search accuracy. All tasks of each application are smartly executed in the CDC and public clouds such that tasks' delay bound constraints are strictly met.

5) Chapter 7 in this dissertation proposes a simulated-annealing-based bi-objective differential evolution algorithm to obtain an approximate Pareto optimal set for the formulated bi-objective optimization problem. Then, the minimal Manhattan distance is adopted to determine a knee solution that specifies Pareto optimal task service rates and task split among ISPs for CDCs in each time slot. In this way, a Profit and QoS-optimized Task Scheduling (PQTS) method is proposed to maximize the profit of a CDC provider, and minimize the average task loss possibility of all applications in CDCs by properly splitting users' tasks among multiple ISPs, and adjusting task service rates of each CDC.

6) Chapter 8 in this dissertation formulates the joint optimization of energy cost and QoS as a bi-objective constrained optimization problem where a $G/G/1$ queuing model is used to analyze the performance of each switched-on server. The problem is solved by a Simulated-annealing-based Adaptive Differential Evolution (SADE) algorithm to jointly minimize both energy cost and tasks' response time. The minimal Manhattan distance method is adopted to obtain a knee from a close-to-Pareto-optimal set for good tradeoff between energy cost minimization and QoS maximization.

7) Chapter 9 in this dissertation proposes an Improved Multi-objective Evolutionary Algorithm based on Decomposition (IMEAD) to achieve energy cost minimization and revenue maximization of CDCs. It cost-effectively executes all tasks while meeting their delay constraints. In each time slot, a constrained bi-objective optimization problem for the tradeoff between maximizing the revenue and minimizing CDCs' energy cost is formulated and solved by IMEAD to achieve a high-quality balance of these two objectives.

8) Chapter 10 in this dissertation adopts a $G/G/1$ queuing system, which is the most general model, to analyze the performance of servers in CDCs. The execution time and interarrival time of each task have arbitrary probability distributions. Based on it, the spatial variations are integrated into a single-objective constrained optimization problem, which is further solved by a proposed Bat Algorithm based on Simulated annealing to find a real-time close-to-optimal solution. In this way, a Fine-grained Spatial Task Scheduling (FSTS) algorithm is proposed to minimize the energy cost of a CDC provider by optimally allocating tasks of heterogeneous applications among multiple CDCs, and specifying the running speed of each server and the number of switched-on servers in each green cloud while strictly meeting response time limits of tasks of all applications.

9) Chapter 11 in this dissertation focuses on a 3-layer architecture. It consists of terminal, edge computing, and CDC layers. This chapter aims to maximize the profit of the system provider by smartly scheduling arriving tasks between CDC and edge computing layers while strictly guaranteeing their response time limits. More specifically, given tasks scheduled to execute in a CDC layer, this chapter explicitly specifies the task service rate of a CDC server in each time slot by using a proposed Simulated-annealing-based Migrating Birds Optimization (SMBO). In addition, given tasks scheduled to an edge computing layer, this chapter explicitly specifies the selected node for each task in each time slot. In this way, this chapter proposes a more fine-grained mechanism to obtain the optimal scheduling strategy for arriving tasks. Specifically, it jointly considers CPU, memory, and bandwidth resource limits, load balance requirements of all nodes, and different processing capacities of heterogeneous nodes in the edge computing layer. In addition, it jointly considers the maximum amount of energy, maximum number of available servers, and task queue stability of servers in the CDC layer. By jointly considering the above-mentioned factors, this chapter proposes a profit-maximized collaborative computation offloading and resource allocation algorithm. The system profit is maximized while response time limits of tasks are strictly met.

Extensive experiments with real-life data, *e.g.*, prices of power grid, solar irradiance, wind speed, and tasks in Google cluster, are conducted to evaluate STSRO, GATS, STTS, TTS, PQTS, SADE, IMEAD, FSTS and SMBO algorithms. Experimental results demonstrate that they outperform several typical benchmark scheduling algorithms.

The rest of this dissertation is organized as follows. Chapter 2 discusses the related studies, and the contributions of this dissertation are comparatively pointed out. Chapters 3–6 show the background work of this dissertation. Chapters 7–

11 present the major contributions of this dissertation. Chapter 12 concludes this dissertation, and points out the future work.

CHAPTER 2

RELATED WORK

This chapter discusses related studies, and the contributions of this dissertation are comparatively pointed out.

2.1 Task Scheduling

A growing number of studies focus on the task scheduling problem in CDCs in recent years [61]–[73]. Shah-Mansour *et al.* [61] formulate a utility maximization problem that considers delay, prices of cloud services and the energy consumption, and produces the optimal scheduling for mobile users' tasks. In addition, the optimal pricing strategy is determined for the cloud provider. It effectively balances the tradeoff between delay and the energy consumption. On the contrary, the scheduling method in this study is coarse-grained because it cannot determine the optimal setting of each server in each CDC. Maqsood *et al.* [62] present multiple algorithms to jointly realize data allocation and task scheduling in a unified way. Besides, a feasible system model for Network-on-Chip architectures is proposed to effectively capture the energy consumption by considering caches, processing cores, and the Network-on-Chip subsystem. Nevertheless, it only considers the optimization of energy consumption in data centers. Nir *et al.* [63] propose an energy and cost-aware task scheduling model that allows mobile devices to schedule some tasks to cloud resources. Its optimal solution can significantly reduce the total cost of the cloud provider but it does not consider the energy produced by renewable sources. Chen and Chang [64] propose a cloud framework to realize user-oriented energy optimization for retail electricity providers. A linear programming model is designed to minimize the multiperiod global cost and stabilize the renewable energy consumption for enhanced integration.

However, it does not consider the spatial diversity of renewable energy sources during delay bound constraints of tasks of multiple applications.

Canali *et al.* [65] propose an allocation model for virtual elements and aim to minimize the energy consumption of a software-defined cloud data center. Besides, the energy consumption is modeled by incorporating virtual elements' computing costs on physical servers, migrating costs across servers, and data transferring costs between virtual elements. On the other hand, the proposed method can be only applied to a single data center. Chen *et al.* [66] present a streaming workflow scheduling algorithm that considers characteristics of streaming workflow and the price diversity of geo-distributed data centers. It aims to minimize the total cost for streaming big data processing provided that the latency requirement is strictly met. Nevertheless, it ignores the spatial diversity of ISP bandwidth prices, power grid prices, and availability of renewable green energy. The work [67] designs a novel task scheduling algorithm to reduce network cost and completion time of big data processing tasks across geo-distributed data centers. It is implemented in Apache Spark, and both indices are reduced by a substantial margin. The work [68] investigates a joint energy management problem for electric vehicles of employees and geo-distributed data centers. A total cost minimization problem of a data center provider is formulated as a large-scale convex optimization one, and solved by a distributed algorithm. The data center workload scheduling and electric vehicle charging under the specified power constraints are jointly realized.

In addition, Wu *et al.* [69] design a method to schedule multicast-oriented tasks in data center networks with different topologies. Lyapunov optimization is adopted to propose a distributed online method to maximize the time-average profit with local information. In addition, a destination grouping mechanism is designed to solve the scalability problem and significantly reduce the number of queues in data center networks. Ismail and Materwala [70] classify and evaluate 13 task scheduling

algorithms with the unified environment and setup to achieve their objective comparison in terms of the energy consumed by the cloud infrastructure. The workload is carefully chosen to typify cloud IoT applications, *e.g.*, wide area power grid measurement systems, connected vehicles, and smart meter infrastructures. Varshney and Simmhan [71] design AutoBoT, which is a collection of task scheduling methods for bag of tasks with strict deadlines on virtual machines, to reduce the monetary cost. It uniquely decreases costs by using preemptible spot-priced virtual machines, which are cheaper but unreliable and time-variant. Timely completion is guaranteed by considering many factors including checkpointing, migration, pricing, and the number of virtual machines. Kumar *et al.* [72] present a green energy-aware task scheduling and classification method by using the container technology for the sustainability of data centers. It transfers arriving tasks from multiple devices to the data center with enough amount of green available energy. Then, a green energy-based container consolidation and host specification method is further proposed. Hsieh *et al.* [73] design a job allocation scheduler to balance utilization of resources. It classifies different jobs and assigns them to CPU-bound and I/O-bound queues. It increases the actual performance of Hadoop and the usage of nodes in heterogeneous computing infrastructure. The inaccurate slot settings are detected by adding two parameters, and a dynamic allocation method is further designed for jobs.

Different from above studies, Chapter 3 in this dissertation aims to minimize the total cost of a CDC provider by jointly exploiting the spatial diversity of ISP bandwidth prices, power grid prices, and availability of renewable green energy while strictly meeting delay bound constraints of tasks of all applications. It jointly determines the optimal allocation of all tasks among multiple available ISPs, and specifies the optimal setting of each server in each CDC in each time slot. Chapter 9 in this dissertation designs an IMEAD algorithm to achieve the energy cost minimization and the revenue maximization of CDCs. All tasks of three applications

are cost-effectively and smartly scheduled to execute in different CDCs within their delay bound limits.

2.2 Green Data Center

Recently, an increasing number of existing studies have been proposed to adopt green renewable energy in large-scale data centers in recent years [74]–[86]. Nguyen and Cheriet [74] design an environment-aware method for virtual slices to cope with the intermittence of renewable energy. A virtual slice allocation problem that considers renewable energy availability, VM locations, and network capacity is designed and solved to effectively reduce the environmental footprint. Qiu *et al.* [75] propose a genetic-based algorithm for chip multiprocessors with phase-change memory in green clouds. It realizes the tradeoff of the memory usage efficiency and the total execution time by scheduling tasks to cores. However, it aims to reduce the energy consumption by only optimizing the memory usage, and it ignores the adoption of green energy. Deng *et al.* [76] propose an online algorithm to achieve the eco-aware energy optimization and load scheduling for distributed data centers. A stochastic optimization problem is obtained and tackled with the Lyapunov optimization theory. Then, an online control algorithm is proposed to achieve the minimization of the eco-aware energy cost of data centers while meeting tasks' performance requirement. It focuses on the long-term time average of the eco-aware power cost. The work [77] proposes a green energy management method that explicitly and implicitly integrates renewable energy in clouds. The concept of green energy virtualization is introduced to address the uncertainty issue in availability of green energy. Two threshold parameters are introduced in SLAs and a greenSLA algorithm is proposed to establish green SLA without causing higher cost.

The work [78] proposes two energy-efficient and computation-efficient embedding algorithms to embed a virtual data center in a green and robust way. Different

network topologies and scales are adopted to evaluate the proposed algorithms with respect to the acceptance ratio, long-term revenue, and energy consumption of a cloud service provider. Khosravi *et al.* [79] design the total energy cost as a relation between the energy consumption and the overhead energy. Then, several efficient virtual machine placement methods are proposed to evaluate their actual performance and determine the parameters that have the greatest impact on the brown and green energy consumption, cost and carbon footprint. Tripathi *et al.* [80] investigate several key parameters about carbon footprint and energy cost, and propose multiple virtual machine placement methods to increase the usage of renewable energy. Total energy cost is formulated as a function of the energy required by servers and overhead energy. It helps providers decrease the reliance on the power grid energy that is typically produced from the burning of fossil fuels. In addition, Anastasopoulos *et al.* [81] develop a service provisioning method according to stochastic linear programming. They adopt dimensionality reduction methods including Lagrangian relaxation and sample average approximation to tackle the computational complexity. The provisioning method realizes fast convergence and decreases the CO_2 emissions by about 60% for different tasks.

In addition, Cheng *et al.* [82] propose a dynamic power-sensitive resource provisioning method for heterogeneous workloads in data centers totally powered by green energy. It aims to maximize the system throughput and reduce the system energy consumption in terms of renewable power supply. It is realized to efficiently search the optimal resource allocation strategy with a simulated annealing algorithm combined with fuzzy performance modeling. Rahmani *et al.* [83] propose a model to obtain high-quality configurations for microgrid in enterprise-scale green data centers. An optimization model is presented for considering greenhouse gas emissions and costs of all components of a microgrid system. The costs of capital, operational, and degradation are obtained according to the 20-year running of the system. A real data

center with a specified load demand is used to demonstrate that the model produces high-quality microgrid configurations for tradeoffs of sustainability and cost. Wu *et al.* [84] design two solution algorithms by using multi-objective optimization, to realize a good tradeoff between cost and brown energy consumption in cloud networks under both data center placement and data center addition scenarios. They give *via* simulations how to select the optimal number of data centers and their sites over two cases based on USNET and NSFNET topologies. Yang *et al.* [85] design a green cloud data center framework that uses the artificial intelligence techniques. A scheduling control engine and an smart refrigerating one are put forward to decrease energy consumption. A green cloud data center platform is built to achieve the scheduling control engine and evaluate the framework feasibility. Experimental results show that it achieves a high-energy-efficiency and low-power-consumption data center operation. Zhang *et al.* [86] consider a green energy-aware inter-data-center virtual machine migration problem in elastic infrastructure, and formulate it as an integer linear program. It aims to minimize the brown energy cost consumed by data centers. CVX and Gurobi are adopted to address the challenging problem for small-scale networks, and several heuristic algorithms are proposed to obtain the optimal solution of large-scale networks.

Different from above studies, Chapter 3 in this dissertation jointly considers the spatial diversity of CDCs' many factors including power grid prices, wind speed, solar irradiance, on-site air density, the maximum available energy and the number of servers in each CDC. Then, it smartly schedules all tasks of each application to CDCs located in multiple geographical locations within their delay bound constraints. In addition, above studies ignore the temporal variations of green energy within tasks' delay bound constraints. Chapter 6 in this dissertation considers applications whose delay bound constraints are relatively long, *e.g.*, 15–30 minutes, and aims to achieve the profit maximization or energy cost minimization of a CDC provider. Chapter

9 in this dissertation jointly optimizes providers' revenue and their energy cost. It formulates a constrained bi-objective optimization problem, and solves it by IMEAD for the optimal tradeoff between these two objectives. It jointly takes advantages of the spatial differences in CDCs provided that delay constraints of tasks of applications are met.

2.3 Profit Maximization

The cloud computing community has proposed several methods to achieve the profit maximization for a CDC provider [87]–[97]. Mei *et al.* [87] design a cloud broker and specify prices of its virtual machines while maximizing its profit and reducing costs for users. They formulate the virtual machine pricing and multiserver configuration as a constrained profit maximization problem. A heuristic method based on bisection search and partial derivative is proposed to solve it. The close-to-optimal solutions are obtained to greatly reduce the user cost. Ma *et al.* [88] investigate dynamic admissions of delay-sensitive tasks with function chain constraints in a distributed cloud, and aim to maximize the profit of a service provider. A dynamic profit maximization problem is formulated and an online heuristic algorithm is designed to solve it. In addition, the offline version of this problem is formulated as NP-hard and a solution of integer linear programming is given. Zhang *et al.* [89] propose polynomial time and truthful auctions to maximize social welfare and/or the profit of a cloud provider. The Fenchel duality is adopted in their primal-dual framework and gives more structures for convex optimization problems than Lagrangian duality. Based on it, a framework of online primal-dual optimization for allocation of virtual machines is proposed to maximize the social welfare. Wan *et al.* [90] formulate a problem of dynamic server pricing in which a data center provider specifies prices of servers according to the resource demand. Then, a reactive pricing algorithm is proposed to dynamically adjust prices of servers in response to changes of states. The close-to-optimal profit is achieved by

considering green energy, battery levels and spot power prices. Theoretical analysis is provided and real-world trace-driven simulations prove it is robust compared with exogenous environment variations.

In addition, Wang *et al.* [91] investigate an interaction system of a smart power grid with a cloud computing system and distributed photovoltaic power generation. They jointly optimize the dispatch and routing of service requests in the cloud with the power flow analysis of power grid. The near-optimal strategies of two players, *i.e.*, a power grid controller and a cloud controller, in Stackelberg game are designed by using simulated annealing and convex optimization techniques. Ren *et al.* [92] consider a wireless cloud system where a service provider manages a data center and provides its cloud services to users at dynamic prices. Then, scheduling of delay-tolerant batch services and pricing strategies are jointly optimized for the long-term profit maximization of the service provider. A provably-efficient algorithm is proposed to realize dynamic scheduling and pricing in an arbitrarily random environment. A close-to-optimal average profit is produced while the length of a job queue is bounded. Patel *et al.* [93] present a generalized framework for network flow-based resource allocation that jointly minimizes energy and maximizes profit. This framework models the profit maximization under three different SLAs of clients. Based on it, optimal resource allocations are derived by considering resource heterogeneity and varying server utilization while SLAs are met. Hammoud *et al.* [94] propose a maximin game theoretical model to assist a broker that is responsible for managing and creating cloud federations. The obtained solution achieves the detection maximization of malicious providers, and improves the profit of cloud providers and quality of service of the cloud federations.

Besides, Deng *et al.* [95] design a system model for a cloud provider to dynamically increase the scale of distributed data centers. It can deploy more VMs of cloud users and effectively decrease the bandwidth cost. An optimization problem

is formulated for the cloud provider to achieve the profit maximization, and further solved in different conditions. Their simulation results prove that the proposed model and algorithms effectively increase both the total revenue and users' satisfaction, and decrease requests' average latency. Wu *et al.* [96] investigate the problem of cloud service continuity and they consider a profit of maintaining the continuity of a service. An optimization problem is first formulated to achieve the total profit maximization subject to energy constraints. To solve it, two approximation algorithms are designed for two practical cases including one with enough servers and another with limited servers. The two algorithms are further combined to realize both good average performance and worst-case performance. Benbrahim *et al.* [97] maximize net profits and minimize penalties of cloud services with placement of VMs. The placement optimization of virtual machines is formulated as a mixed integer non-linear program, which is NP-hard, and a heuristic method is proposed to optimize their net profits and overall penalties.

Different from these studies, Chapter 9 in this dissertation aims to simultaneously maximize the revenue of CDC providers, and minimize its energy cost. The revenue and cost tradeoff solution aims to achieve a high-quality balance between the revenue of CDC providers and its cost. A novel bi-objective optimization algorithm named IMEAD is proposed to realize it, and intelligently schedules all tasks to different CDCs within their delay bound limits.

2.4 Hybrid Cloud

A growing number of emerging studies investigate task scheduling in a hybrid cloud in recent years [98]–[109]. Mao *et al.* [98] propose an algorithm to schedule MapReduce tasks, and meet the deadline and cost constraints in a hybrid cloud. It maximizes the resource efficiency of a private cloud and minimizes the execution cost of public clouds such that job execution time is within its constraint. Zuo *et al.* [99] present

a resource allocation method where some tasks are outsourced to public clouds when resources of an IaaS provider are not sufficient to meet tasks' performance demand. The profit maximization problem of the IaaS provider is formulated as an integer program and solved by a self-adaptive method. Different from above studies, this dissertation work aims to maximize profit by investigating the temporal variations of many factors including VM prices of public clouds and the revenue of the hybrid CDC. Vasile *et al.* [100] propose a resource-aware scheduling approach for workflows and batch jobs. Resources are divided into groups with hierarchical clustering. A scheduling algorithm is proposed for each group to dynamically provision resources for heterogeneous computing-intensive and I/O-intensive applications. Genez *et al.* [101] aim to evaluate the effect of imprecise bandwidth information in inter-cloud links on the scheduling of workflows. They propose a method to tackle imprecise bandwidth information and its impact on the estimates of cost and makespan. The proposed method chooses a deflating factor on the available bandwidth as the input to the workflow scheduler. Simulation results demonstrate that it increases the number of solutions whose makespans are shorter than the specified deadline, and decreases the underestimations of cost and makespan.

In addition, Zhu *et al.* [102] consider a workflow scheduling problem in a hybrid cloud system where tasks are stochastic, compute-intensive, dependent, deadline-constrained and scheduled on distributed and elastic cloud resources. They propose an iterated heuristic framework to execute jobs, and explore three optimization objectives including usage time, utilization and number of VMs. Then, they propose two job collecting mechanisms and develop two timetabling approaches. Prasad *et al.* [103] design an algorithm to solve the procurement problem of multiple resources in hybrid clouds. Users in cloud submit their needs, and vendors send bids including QoS, price and their resource sets. The algorithm is scalable given that there are a great number of more continually increasing cloud vendors. Hwang *et al.* [104] propose a model

to solve the migration challenges that convert one resource into the same or another one in hybrid clouds. The problem is formulated as a constraint satisfaction one. The server components are iteratively decomposed and servers are consolidated. A mapping algorithm is proposed to match computing resources with network ones for ensuring network affinity. Lu *et al.* [105] aim to maximize an profit of a multimedia cloud service provider by scheduling multimedia services to distributed users in a hybrid cloud. A service provisioning model is designed to provision resources in the hybrid cloud. A Lyapunov optimization method is adopted to maximize the profit of the cloud provider, and an online algorithm is proposed to provision the hybrid cloud in a distributed way. An ϵ-persistent method is applied to bound the worst-case latency of provisioned requests.

Besides, Charrada *et al.* [106] investigate the deployment of applications by considering the placement selection of their components between a private cloud and public clouds. An efficient algorithm is proposed to deploy service-based applications that can be architecture-based and behavior-based compositions of services in a hybrid cloud. Qiu *et al.* [107] employ a Lyapunov optimization method, and a dynamic control algorithm is proposed to smartly place contents and schedule requests in a hybrid cloud distributed in multiple data centers. It minimizes the overall operational cost while meeting request response time limits. Rigorous analysis is provided to prove that it well bounds response times within a specified QoS target, and guarantees that the cost is also bounded within a constant gap. Li *et al* [108] aim to optimize the operational cost for a hybrid cloud provider by theoretically analyzing the optimization problem by using a framework of Lyapunov optimization. A dynamic and online provisioning algorithm is proposed, and it tackles real-life challenges without priori renting price information of public clouds and the probability distribution of requests. A set of real-life data is used to prove that it effectively decreases the operational cost. Zhang *et al.* [109] design a two-stage task scheduling

framework and propose efficient algorithms to enhance the service quality of clouds. According to the historical information of task scheduling, an optimal number of VMs with different resource attributes are created in advance. Then, according to the complexity of tasks, the optimal set of VMs are selected from the created ones to execute tasks. Based on the two-stage strategy, efficient task scheduling algorithms are further presented.

Different from above studies, Chapter 6 in this dissertation specifies the number of tasks executed in public clouds and task service rates of the CDC within tasks' delay bound constraints. Then, all tasks are smartly executed in the CDC and public clouds while tasks' delay bound constraints are strictly met.

2.5 Application Behavior Analysis

Recently, an increasing number of studies analyze the application behavior analysis and model the performance of applications in CDCs to achieve the QoS modeling [23, 24, 76], [110]–[125]. The work [23] adopts a $G/D/1$ queue to characterize the distribution of workload among geographically distributed green data centers. Then, an optimization problem is formulated as a convex optimization one, and solved to maximize the profit of data center providers by optimally distributing workload among multiple data centers. The work [24] proposes a convex optimization-based strategy to maximize the profit of green data centers. It uses a $G/D/1$ queueing model to calculate the probability that the waiting time of a task is larger than its specified deadline. It considers SLAs between users and data centers, availability of renewable energy generation at each data center and stochastic characteristics of workload. In [110], servers in each rack are modeled as an $M/M/m$ queuing system and their actual average latency is calculated accordingly. Then, an adaptive power control algorithm is proposed to investigate the correlation between computer room air conditioners and the power consumption of servers. The work [111] adopts an $M/M/n$ queuing

model to calculate the average delay of each active server for data centers. Then, a mixed-integer nonlinear programming problem is solved to realize the optimal energy cost management and load balancing for data centers while meeting users' SLAs. In the work in [112], the average response time at each server in a cloud is calculated according to an $M/M/c$ queuing system, and then the average response time of all users is obtained. A profit optimization problem is next formulated accordingly for the cloud provider.

In addition, Cao *et al.* [113] regard a multiserver system as an $M/M/m$ queueing model based on which an optimal multiserver configuration problem is formulated and solved analytically to realize profit maximization in a cloud environment. They consider two power consumption and server speed models, and derive a probability density function of waiting time for each new task. Bi *et al.* [114] establish a hybrid queueing model based on the combination of $M/M/1$ and $M/M/m$ queueing models. Then, a non-linear constrained optimization problem is formulated, and solved with a hybrid meta-heuristic algorithm to maximize the profit of cloud providers provided that clients' performance requirements are met. Furthermore, the number of VMs for each tier of multi-tier web applications is determined to improve the performance of applications and reduce the energy cost of resources. Mei *et al.* [115] treat a service system as an $M/M/m$ queueing model based on which performance indicators that affect the profit of the proposed double resource renting scheme are designed and analyzed. The double renting scheme can greatly reduce the waste of resources while meeting QoS of all tasks in clouds. Their results demonstrate that it guarantees the service quality of all tasks, and achieves more profit.

Nevertheless, the above studies can only ensure that the average delay of all arriving tasks is within their delay bound constraints. Yet the long tail effect exists in the delay distribution of arriving tasks in realistic data centers [116]. It indicates that the actual delay of some arriving tasks might not meet their delay bound constraints.

Chapter 5 in this dissertation distinguishes from these studies in that STTS-scheduled tasks strictly meet delay bound constraints of tasks. In addition, these above studies handle QoS of tasks by using constraints in their formulated optimization problems. However, it is more advantageous to consider the QoS optimization as an objective than a constraint as this can give more choices to decision makers. Hence, different from above-mentioned studies, Chapter 7 in this dissertation aims to jointly maximize the profit of CDC providers, and QoS of tasks of all applications.

In addition, Bi *et al.* [117] compute task arriving rates according to internal and external workload for multiple resource-intensive applications. They develop a probabilistic queueing model to cope with non-steady states in a smart controller. Then, computing and storage resource consumption in a virtualized CDC is minimized. El Kafhali and Salah [118] propose a stochastic model according to queuing theory to analyze the performance. Data center platforms are modeled as an open queuing system for their QoS analysis. Then, the number of needed instances of VMs is estimated and used to meet specified QoS requirements. Satpathy *et al.* [119] propose a queueing model to schedule and manage a set of VM requests. This queueing model makes it easy to realize, analyze and validate complex systems like clouds. Its queueing structure is designed as a single-queue-single-service facility by using an $M/M/1$ queue. VM requests are executed with a First-Come-First-Serve (FCFS) manner and forwarded to data centers for placement. Then, a multi-objective VM placement method is designed to decrease the resource and power consumption at data centers. Ponraj [120] adds tasks into a priority-based probability queuing model, and schedules them into a suitable VM. Specifically, an $M/G/1$ queuing model is adopted to derive the waiting time of tasks. Then, a VM placement algorithm is proposed to reduce completion time and processing cost by considering resources, QoS metrics and VM status. It provides a solution to minimize the completion time of overall jobs in both dynamic and static workloads.

Besides, Darzanos *et al.* [121] model virtualized infrastructure of each cloud service provider (CSP) as an $M/M/1$ queueing system. Based on it, revenue and cost functions for each CSP are derived. Then, task forwarding-based and capacity sharing-based federation methods are designed, and the joint business and the reward-driven modes are proposed to benefit both CSPs and customers. Li *et al.* [122] propose a parallel virtual queue model to buffer tasks of the same type into a separate queue. Three parallel policies including buffering, offloading and resource allocation are designed to improve task completion ratio, resource allocation balance and throughput. Fang *et al.* [123] treat a public CSP as an $M/M/1$ queueing system and study the pricing effect of CSP on the equilibrium behaviors of independent cloud users in a monopoly cloud market. Two pricing mechanisms are proposed to maximize both social welfare and revenue. Besides, a cloud broker is also modeled as an $M/M/$l queueing system with infinite capacity, and a price-based resource access control method is designed. They also analyze how their pricing mechanism affects equilibrium behaviors of cloud users, and the social-optimal and revenue-optimal pricing strategies in view of this CSP. Santhi *et al.* [124] adopt two connected queues including $M^{[X]}/M/1$ and $M/M/1$ models for a cloud architecture in a healthcare system. Then, the FCFS discipline is adopted, and the waiting time and the number of patients of different classes in both queues are obtained. Fang *et al.* [125] analyze the performance of each rack and servers on it by modeling it as an $M/M/n$ queueing system. Then, a constrained nonlinear optimal control problem is formulated to minimize energy consumed by a cloud system while QoS and thermal constraints of each device are met. The simulation results demonstrate that it achieves significant energy saving while guaranteeing throughput of a data center.

Unlike these methods, Chapters 8 and 10 in this dissertation adopt the most general model, *i.e.*, a $G/G/1$ queue, to analyze the performance of each switched-on server. In the model, both execution time and interarrival time of each task follow

arbitrary probability distributions. Different from them, Chapter 8 formulates a bi-objective constrained optimization problem and solves it by the proposed SADE to obtain a good tradeoff between energy cost minimization and QoS maximization. Based on the $G/G/1$ queue model, FSTS is proposed in Chapter 10 to specify a running speed of each server and the number of switched-on servers in each data center at different locations. In addition, these existing studies design performance needs as constraints in their optimization problems. Therefore, performance or QoS needs are only satisfied to the minimum extent in their methods.

2.6 Energy Management in CDCs

Recently, more and more studies are conducted to realize efficient energy management in CDCs [126]–[141]. Yu *et al.* [126] investigate an energy management problem for multiple microgrids of data centers. They aim to achieve the long-term operational cost minimization by considering uncertainties in prices of power grid, renewable energy and arriving workloads. They design a real-time and distributed algorithm to solve their proposed problem. Fang *et al.* [127] propose a two-time-scale approach to minimize the energy consumed by high-performance-computing CDCs through dynamic processor frequency scaling, cooling supplement and task assignment. Then, the energy minimization problem is solved in a two-time-scale manner. The processor frequency and task assignment are optimized in a steady thermal environment, and the cooling supplement is optimized in a dynamic one. Rong *et al.* [128] propose a comprehensive set of mechanisms to minimize the environmental impact and maximize the efficiency of data centers by considering cost reduction, energy consumption and environment protection. They also show the future energy-saving trends for data centers. Vasudevan *et al.* [129] formulate the assignment of applications to VMs as a problem of profile-driven optimization under different constraints, and solve it by a genetic algorithm. They improve a penalty-based genetic

algorithm by applying the longest cloudlet, the fastest processor and a procedure for repairing infeasible solutions. Finally, they develop a scalable method for application assignment to realize the tradeoff between resource utilization and energy efficiency.

In addition, Hu *et al.* [130] consider several geo-distributed data centers that are powered by both energy storage and fuel cells. An online algorithm is designed to minimize the gap between energy demand and supply by joint workload scheduling and energy management. The output of fuel cells is managed and workloads are migrated among data centers. Kaur *et al.* [131] apply software defined data centers to decrease energy utilization levels. A multiobjective optimization problem is formulated to derive the optimal resource allocation for applications, and a suboptimal method based on the first fit decreasing algorithm is proposed to reduce energy consumption with negligible violations of QoS. Hogade *et al.* [134] give several workload management methods to minimize the energy cost for geo-distributed data centers. They consider many factors including co-location interference, data center cooling power, time-of-use electricity pricing, net metering, renewable energy and distribution models of peak demand pricing. Canali *et al.* [135] propose a method to minimize the energy consumption in a software-defined CDC by efficiently allocating virtual elements. The energy consumption is modeled by investigating computing, migrating and data transferring costs. Three different strategies are proposed to save energy without a significant growth in solution complexity.

Besides, Yadav *et al.* [136] propose three adaptive energy-aware algorithms based on robust regression models to minimize SLA violations and energy consumption. Experiments based on real workload traces prove that these algorithms effectively decrease energy consumption while keeping the specified performance levels in a CDC. Hu *et al.* [137] aim to mitigate the limited load by adjusting both energy demand and supply with joint energy management and workload scheduling. They investigate multiple distributed data centers that are powered by both energy storage

and fuel cells. An online algorithm is proposed to the gap minimization between energy demand and supply by simultaneously migrating workloads and optimizing fuel cells output among data centers. Al-Dulaimi *et al.* [138] develop cloud radio access network solutions and algorithms for computing the optimal network mapping solution and backup topology while interfacing requests from inactive or low-flow femtocells are denied. Then, a graph-coloring method is proposed to label new fronthaul femtocell clusters by using power as a performance metric. Simulation results are given to show the efficient performance of obtained solutions in large-scale networks. Chen *et al.* [141] formulate the resource allocation as a robust optimization problem with the objective of minimizing the worst-case net cost. It is further converted into a convex program, which is solved in a distributed manner with a dual decomposition method for constructing sustainable and energy-efficient data centers. It jointly exploits the spatio-temporal variations of workload demand, local temperature, energy prices and the availability of renewable power. It performs better than several state-of-the-art allocation methods with extensive numerical experiments according to real-life data.

Different from the above-mentioned studies, Chapter 8 in this dissertation aims to jointly minimize both the energy cost of a distributed data center provider, and the task response time of all tasks. It intelligently and jointly splits all arriving tasks among data centers, and specifies the number of switched-on servers and the running speed of each server in each data center. In addition, different from these studies, Chapter 10 in this dissertation aims to minimize the energy cost of a data center provider by investigating the spatial variations of several factors, which include CPU utilization of servers, running speed limit of each server, peak and idle power of each server, power usage effectiveness, maximum available energy and servers, price of power grid, and task queue stability in each CDC. Then, it proposes FSTS to smartly consume power grid, wind and solar energy by optimally allocating tasks of

heterogeneous applications among multiple data centers, and specifying the running speed of each server and the number of switched-on servers in each CDC provided that response time limits of tasks of all applications are strictly met.

2.7 Resource Optimization Methods in Clouds

In recent years, resource optimization methods have attracted a growing amount of attention [142]–[153]. Lyu *et al.* [142] develop a semidistributed and heuristic offloading decision algorithm to jointly optimize the offloading, computation and communication resources for system utility maximization. The formulated problem is reduced to a submodular maximization one, and further decomposed into two subproblems. The first one is tackled with convex and quasiconvex optimization, and the second one is tackled with submodular set function optimization. Li *et al.* [143] address the problem of virtual network function placement by investigating service function chain requests of users. It is formulated as an integer linear programming problem for minimizing the number of physical machines, and solved by a two-stage heuristic method, which includes a correlation-based greedy algorithm and an adjustment one for requests of virtual network functions. Li *et al.* [144] jointly optimize resource optimization and congestion control to realize the energy efficiency-guaranteed tradeoff between delay performance and throughput in heterogeneous cloud radio access networks. Their formulated stochastic optimization problem is transformed into three subproblems solved in parallel. Luna *et al.* [145] design a decentralized probabilistic method to optimize performance of cloud services. They consider an Infrastructure-as-a-service framework in which users can configure virtual resources dynamically to meet specific computational needs. It supports performance metrics of the cloud and security metrics by using cryptographic algorithms for data storage.

In addition, Mireslami *et al.* [146] propose a runtime-friendly and cost-effective algorithm to minimize the cost of deployment while guaranteeing QoS performance

needs. It provides an optimal option, from users' point of view, to deploy Web applications in a cloud environment. A multi-objective optimization algorithm is proposed to simultaneously minimize cost and maximize QoS performance. Zhou *et al.* [147] develop a declarative optimization engine to provision resources for scientific workflows in distributed clouds. It allows users to describe their workflow optimization objectives and constraints of specific problems. A probabilistic optimization method is proposed to evaluate their formulated problems to tackle the cloud dynamics. The power of GPUs is leveraged to accelerate the solution search in a timely and fast way. Chou *et al.* [148] propose a dynamic power-saving resource allocation method to improve energy efficiency by using a particle swarm optimization algorithm. It investigates the energy consumed by physical and virtual machines, and improves the energy efficiency ratio of an air conditioner. The least squares regression mechanism is adopted to predict resource utilization for provisioning VMs. Domanal *et al.* [149] propose a hybrid bio-inspired algorithm for resource management and task scheduling in a cloud environment. It efficiently schedules tasks to virtual machines with an improved particle swarm optimization algorithm. Then, demanded resources including CPU and memory are allocated and managed accordingly for improving reliability and decreasing the average response time.

Besides, Alrawahi *et al.* [150] investigate the challenging QoS satisfaction when resources in cloud of things are traded and resource allocation is performed. A QoS model is designed to address the problem by optimizing five QoS objectives including energy consumption, resource cost, fault tolerance, response time and resource coverage. Jiao *et al.* [151] consider a dynamic resource allocation problem for service provisioning in distributed multi-tier clouds, and develop an online algorithm that constructs a series of regularized subproblems, which are solved at corresponding time slots. Two predictive control algorithms are designed to inherit theoretical assurance of the online algorithm, and improve practical performance. Li *et al.*

[152] formulate the resource reservation and allocation with uncertain needs of mobile users as a robust optimization model. Then, a robust joint resource reservation and allocation algorithm in mobile cloud computing is designed to achieve the optimal provisioning of radio resources and VM resources. Qiu *et al.* [153] design a correlated modeling method by using a Bayesian method, Laplace-Stieltjes transform, and semi-Markov models to analyze reliability performance and energy correlations for cloud applications. A recursive method is presented by using a check-pointing fault recovery mechanism, and can jointly optimize energy consumption and service time for running a cloud application. Finally, a derivation method is presented to specify Pareto optimal solutions, which are further evaluated with illustrative examples.

Different from these studies, Chapter 10 in this dissertation formulates an energy minimization problem as a single-objective constrained optimization one, and solves it with a newly proposed Simulated-annealing-based Bat Algorithm (SBA). SBA combines the Metropolis acceptance criterion of SA into BA, and performs the SA-based selection and the disruptive selection to ensure its high convergence speed and solution accuracy.

2.8 Computation Offloading in Edge Computing

The computation offloading is of great importance in edge computing, and has been attracting a growing amount of attention in recent years [57], [154]–[165]. Zhao *et al.* [57] propose a collaborative computation offloading method that offloads application services to automobiles in vehicular networks, and optimizes resource allocation for MEC and cloud computing. A distributed algorithm for computation offloading and resource allocation is designed to obtain the optimal solution. The computation time and system utility for computation-intensive tasks are improved effectively. Liu *et al.* [154] design a price-based distributed approach to schedule users' offloaded computation tasks. They formulate a Stackelberg game to analyze the

interaction between users and the edge cloud. The prices are set by the edge cloud to achieve revenue maximization by considering its finite computation capacity. Then, each user separately makes its own offloading decision to realize the minimization of latency and payment. Based on the network information of the edge cloud, differentiated and uniform pricing algorithms are developed and implemented in a distributed way. Wei *et al.* [155] model computation offloading as a Markov decision process (MDP), and reinforcement learning algorithms are used to obtain the optimal offloading decision. A polynomial value function approximation approach is developed to accelerate the learning process. Then, an after-state reinforcement learning mechanism for MDP is designed to obtain the optimal offloading strategy for real MEC systems. Guo and Liu [157] adopt hybrid fiber-wireless (FiWi) networks to support the coexistence of multiaccess edge computing and a centralized cloud. An architecture is presented to adopt the FiWi access networks. The problem of collaborative computation offloading among cloud and MEC is investigated. Then, approximation game-theoretic collaborative computation offloading strategies are proposed. Their strategies achieve higher performance and availability than existing MEC offloading methods.

In addition, Ning *et al.* [158] formulate a single user computation offloading problem with unlimited MEC resources, and it is solved by a branch and bound algorithm. Then, they formulate a multiuser computation offloading problem as a mixed integer linear programming one by investigating resource competition among mobile users. A heuristic iterative resource allocation algorithm for MEC is designed to dynamically obtain the offloading strategy. Bi and Zhang [159] aim to maximize the weighted sum computation rate of all wireless devices by jointly optimizing the computing mode selection and the allocation of system transmission time. Then, a joint optimization approach based on the alternating direction method of multipliers decomposition mechanism is proposed and it achieves slower growth of computational

complexity as the size of networks increases. Lei *et al.* [160] design a joint multiuser scheduling and computation offloading algorithm in an edge computing system to minimize the average weighted sum of power consumption and delay under stochastic traffic arrival. A dynamic optimization problem is formulated into an infinite-horizon and continuous-time MDP model. An approximate dynamic programming method is proposed to deal with the problem of curse of dimensionality. Yu *et al.* [161] investigate a scenario where mobile users offload their computation tasks to the network edge, and share the computed results among them. They design a fine-grained optimal collaborative offloading strategy with caching-enhancements to achieve the execution delay minimization at the mobile terminal side. An optimal offloading with a caching enhancement scheme is proposed for femto-cloud and MEC scenarios, respectively.

Besides, Chen *et al.* [162] design an online peer offloading framework for small-cell base stations (SBSs) by adopting a Lyapunov method. It aims to maximize the long-term performance of system while keeping energy consumption below a long-term constraint of each SBS. They formulate a peer offloading game among SBSs and analyze its efficiency loss and equilibrium. The performance of edge computing is dramatically improved with decentralized peer offloading among SBSs. Dinh *et al.* [163] investigate a multi-edge-node multi-user computation offloading problem, which is formulated as a non-cooperative exact potential game. In addition, a model-free reinforcement learning offloading method for unknown channel state information is proposed to help mobile users learn their long-term offloading policies to maximize their long-term payoffs. Hong *et al.* [164] investigate communication-routing and computation-offloading problems to minimize energy consumption and computation time of each task. The joint problem is formulated as a potential game where devices in industrial IoTs specify their computation-offloading policies. Then, a cooperative-messaging multi-hop method is proposed and two QoS-aware distributed

algorithms are developed to realize the Nash equilibrium. Their simulation results prove that the proposed algorithms provide high performance gain for IoTs in different scenarios and high scalability when the device size grows. Wei *et al.* [165] consider a multi-user computation offloading problem for mobile cloud computing in a dynamic environment. Mobile users are inactive or active dynamically, and their wireless channels to offload computation change randomly. An offloading decision process is formulated as a stochastic game for mobile users in a dynamic environment because each user selfishly makes its computation offloading decision. A multi-agent stochastic algorithm is proposed to achieve the Nash equilibrium with an analytically derived convergence rate. Although recent studies consider the computation offloading for edge computing, they fail to achieve the joint optimization of computation offloading and resource allocation in CDC.

Different from these studies, Chapter 11 in this dissertation jointly optimizes the computation offloading between CDC and edge computing layers, and resource allocation in the edge computing layer for latency sensitive tasks. The resources include CPU, memory, and bandwidth at the edge computing layer and servers in the CDC layer.

2.9 Performance Optimization in Edge Computing

In edge computing, performance optimization is an important yet challenging topic, which involves evaluating performance and deciding where to execute users' tasks and how to allocate computing resources [166]–[177]. Responsiveness of tasks is important because applications are usually real-time. Tao *et al.* [166] analyze the energy efficiency and performance guarantee of MEC. An energy minimizing optimization problem is formulated and solved by using Karush-Kuhn-Tucker conditions for better performance of tasks and lower energy consumption. A request offloading scheme is designed by specifying bandwidth capacity and energy consumption at each time slot.

Zhu *et al.* [167] design a solution for quality and latency optimized task scheduling in vehicular fog computing. The task allocation across mobile and stationary fog nodes is formulated as a bi-objective optimization problem by considering constraints on quality loss, service latency and fog capacity. An event-triggered dynamic task allocation method is proposed by applying binary particle swarm optimization and linear programming-based optimization to achieve a tradeoff between quality loss and service latency. Ren *et al.* [168] formulate a joint computation and communication resource allocation problem to minimize the average latency of all mobile devices. A closed-form optimal task allocation policy is derived in terms of normalized cloud computation and backhaul communication capacities. In addition, the original problem is further transformed into an equivalent convex optimization one and solved by a convex optimization technique to obtain the optimal computation resource allocation. Han *et al.* [169] consider a two-tier network in which helpers with caching resources are deployed in a coverage area of base stations. The caching optimization problems are formulated and their convexity properties are given. In this way, the optimal caching policy is obtained by a low-complexity algorithm. Simulation results prove the proposed policy achieves a significant performance gain over an existing policy for video on demand services.

In addition, Mehrabi *et al.* [170] design a heuristic-based and low-complexity mechanism with minimum requirement for parameter tuning in multi-access edge computing environments. Its optimized solution is studied against two typical client-based solutions including buffer-based adaptation and rate-based adaptation. It aims to prove the solution efficiency and quantify benefits brought by network-assisted adaptation over client-based approaches. Wang *et al.* [171] propose a framework for a performance-power tradeoff of mobile service providers by jointly scheduling network resources in a cloud radio access network and computation resources in mobile edge cloud computing. A resource scheduling problem is formulated as a stochastic one

and an optimization framework is designed with an extended Lyapunov method for guaranteeing low congestion and high system stability. Hu and Li [172] formulate a transmitting power allocation problem for energy consumption minimization of mobile users. It is solved by a subgradient-based noncooperative game model and a quasiconvex technique. A joint resource scheduling and request offloading problem is modeled as a mixed-integer nonlinear program for the response delay minimization of requests. It is transformed into a double decision-making problem solved by an improved fast and elitist multiobjective genetic algorithm. Ren *et al.* [173] formulate a joint computation and communication resource allocation problem for the weighted-sum latency minimization of all mobile devices. Then, an optimal closed-form task allocation policy is obtained as a function of the cloud computation and the backhaul communication capacities by adopting the convex optimization. Simulation results show that the proposed collaborative scheme achieves much better delay performance than traditional schemes.

Besides, Wu *et al.* [174] aim to improve code offloading on edge devices by focusing on maximizing the capability of CPU computation, and minimizing cross-device I/O and interdomain data transfer cost. The efficient and in-depth analysis of both extra system burden and performance optimization is conducted. Then, a distributed and fast code offloading method for edge devices is implemented to offload user-specified binary codes across heterogeneous instruction set architectures. It improves the system performance, and decreases burdens on data transfer and I/O latency. Ouyang *et al.* [175] investigate a service performance optimization problem with a long-term cost constraint in the mobile edge. To address unpredictable mobility of users, Lyapunov optimization is applied to decompose the optimization problem into several real-time optimization subproblems that do not need priori user mobility information. An approximation algorithm is designed to obtain a close-to-optimal solution based on Markov approximation. Tao *et al.* [176] consider an

energy efficiency optimization problem with a guaranteed performance constraint in MEC. Karush-Kuhn-Tucker conditions are applied to solve it, and a request offloading method is presented for achieving less energy consumption with higher performance of tasks. The offloading method is determined by bandwidth capacity and energy consumption. Chen *et al.* [177] model the optimal computation offloading as a MDP, and aim to maximize the long-term utility performance. The offloading decision is made according to energy and task queue states, and channel qualities between base stations and mobile users. Then, to avoid the high dimensionality curse in a state space, a deep Q-network-based strategic algorithm is proposed to achieve computation offloading without the priori information of network dynamics.

Compared with existing studies, Chapter 11 in this dissertation aims to improve the system performance by proposing a smart offloading algorithm. It considers two offloading destinations (nodes in edge and servers in CDC), and provides a fine-grained model for response time of tasks executed in edge and CDC by jointly splitting tasks among nodes in edge, and specifying the task service rate of servers in CDC.

2.10 Energy Optimization in Edge Computing

Optimizing energy consumption is one of the most challenging problems in edge computing because its nodes are typically equipped with limited battery energy [178]–[189]. Xu *et al.* [178] investigate a multiuser MEC system with a constraint of task latency. Users can reduce energy consumption by partially or completely offloading their tasks to an server for MEC while meeting a latency constraint. They formulate the joint optimization of data compression, computation offloading and resource allocation as a problem with constraints of latency and MEC computation capacity. The problem is then transformed into a convex one solved by convex optimization. Li *et al.* [179] incorporate MEC into virtualized cellular networks with machine-to-machine communications to optimize computing resource allocation

and reduce energy consumption. They formulate a random access process as partially observable MDP to minimize energy consumption and execution time of the system. Zhang *et al.* [180] consider a power consumption problem in a multiuser MEC system with energy harvesting devices. A problem of power consumption minimization with constraints of QoS and battery stability is formulated as a stochastic optimization program. An online algorithm based on Lyapunov optimization is designed to solve the problem. It only needs current states of users. A distributed algorithm is proposed by using an alternating direction approach of multipliers to decrease the computational complexity. Zhang *et al.* [181] present an dynamic and online task scheduling method to consider a tradeoff between execution delay and energy consumption for a MEC system with energy harvesting capability. An energy consumption and execution delay minimization problem with buffer queue stability and battery level constraints is formulated. A Lyapunov optimization approach is adopted to obtain the optimal scheduling of CPU-cycle frequencies and data transmission power for mobile devices.

In addition, Cui *et al.* [182] investigate a tradeoff between the latency and energy consumption to meet user needs of different IoT applications. A constrained multiobjective optimization problem is formalized and solved to find the optimal solutions with the improved fast and elitist nondominated sorting genetic algorithm including problem-specific genetic operators and an encoding scheme. Ji *et al.* [183] investigate a power consumption problem in a multiuser MEC system with energy harvesting devices. The power consumption minimization problem is formulated as a stochastic optimization program. A Lyapunov optimization-based online algorithm is designed to solve it. In addition, a distributed algorithm is proposed to reduce system computational complexity with an alternating direction method of multipliers. Sun *et al.* [184] develop an energy-aware mobility management method to optimize the delay due to both radio access and computation under a long-term energy consumption

constraint of users. It is derived based on multi-armed bandit and Lyapunov optimization theories, and it well tackles the imperfect state information of the system in an online manner. Zhou *et al.* [185] consider a problem of energy-efficient workload offloading and develop a distributed low-complexity method by using a consensus alternating direction method of multipliers. The problem incorporates many local variables for each user equipment, and it is transformed into a general consensus one with separable constraints and objectives. The consensus problem is further decomposed into several subproblems distributed across users' equipments and solved in a simultaneous manner. Simulation results prove that the energy consumption is reduced significantly by their proposed algorithm.

Besides, Pu *et al.* [186] develop a hybrid edge computing framework to augment vehicle resources for large-scale vehicular crowdsensing applications. It adopts cooperative vehicles and a VM pool of an edge cloud controlled by an application manager. A multitask and multivehicle offloading problem is formulated to minimize the energy consumed by network-wide vehicles that serve heterogeneous applications, and reconcile both vehicle incentives and application deadlines. Then, a knapsack-based resource allocation method for a VM pool, and a graph transformation-based workload assignment policy are proposed. Nan *et al.* [187] employ dual energy sources to support fog nodes where solar power provides primary energy and grid power serves as the backup supply. An analytic framework is proposed to incorporate green energy to support the operation of fog computing-based systems. A Lyapunov optimization-based algorithm is proposed to trade off average response time and energy cost in IoTs. Dong *et al.* [188] develop a cooperative fog computing system to offload workload on an entire fog layer by forwarding data. Then, a joint optimization problem is formulated to trade off quality of experience and energy in a fairness-enabled fog computing process. Its convexity is given and a fairness cooperation algorithm is designed to find the optimal fairness cooperation strategy

of all fog nodes. Zhou *et al.* [189] investigate a problem of energy-efficient workload offloading in vehicular networks. To solve it, a low-complexity distributed method is designed by using a consensus alternating direction approach of multipliers. Several local variables are incorporated for each user equipment, and the original problem is converted into an equivalent consensus problem with multiple objectives and different constraints. Then, the consensus problem is decomposed into many subproblems, which are solved simultaneously.

Different from these studies, Chapter 11 in this dissertation provides a higher-accuracy and fine-grained energy model. It jointly considers CPU, memory, and bandwidth resource limits, load balance requirements of all nodes, and different processing capacities of heterogeneous nodes in the edge computing layer. In addition, it jointly considers key parameters of CPU utilization, task service rate of CDC servers, task arriving rate, power usage effectiveness, peak and idle power of each server, price of power grid, maximum amount of energy, maximum number of available servers and task queue stability in the CDC layer.

CHAPTER 3

SPATIAL TASK SCHEDULING FOR COST MINIMIZATION IN DISTRIBUTED GREEN CLOUD DATA CENTERS

This chapter presents the proposed Spatial Task Scheduling and Resource Optimization (STSRO) algorithm, and it is organized as follows. Section 3.1 presents the motivation and system architecture of cloud data centers (CDCs). Based on the architecture of CDCs, Section 3.2 formulates a cost minimization problem for the CDC provider. Section 3.3 describes the proposed Simulated-annealing-based Bat Algorithm (SBA) algorithm to solve the problem and to develop STSRO to minimize the total cost of the CDC provider. Trace-driven experiments with real-life data are conducted to evaluate the proposed STSRO in Section 3.4. Section 3.5 concludes this chapter.

3.1 Motivation and System Architecture

This section presents the system architecture of CDCs illustrated in Figure 3.1. A typical cloud provider manages multiple CDCs in different locations, and provides different types of applications to global users. Each CDC typically hosts a server cluster consisting of a huge number of servers that range from several hundreds to several thousands. In addition, to guarantee the response time, robustness and availability, multiple available ISPs that transfer data among CDCs and users are designed to connect to each CDC. In addition, similar to the work in [18], it is assumed that replicas, *e.g.*, programs and data, for each application have been copied and distributed across all CDCs. Thus, applications and their data are consistent with each other, and therefore, tasks of each application can be independently executed within each CDC. Besides, it is assumed that servers of each application are homogeneous while servers of different applications are heterogeneous in hardware.

In Figure 3.1, users around the world send their various tasks to CDCs through multiple types of electronic devices, *e.g.*, smart phones, computers, servers and

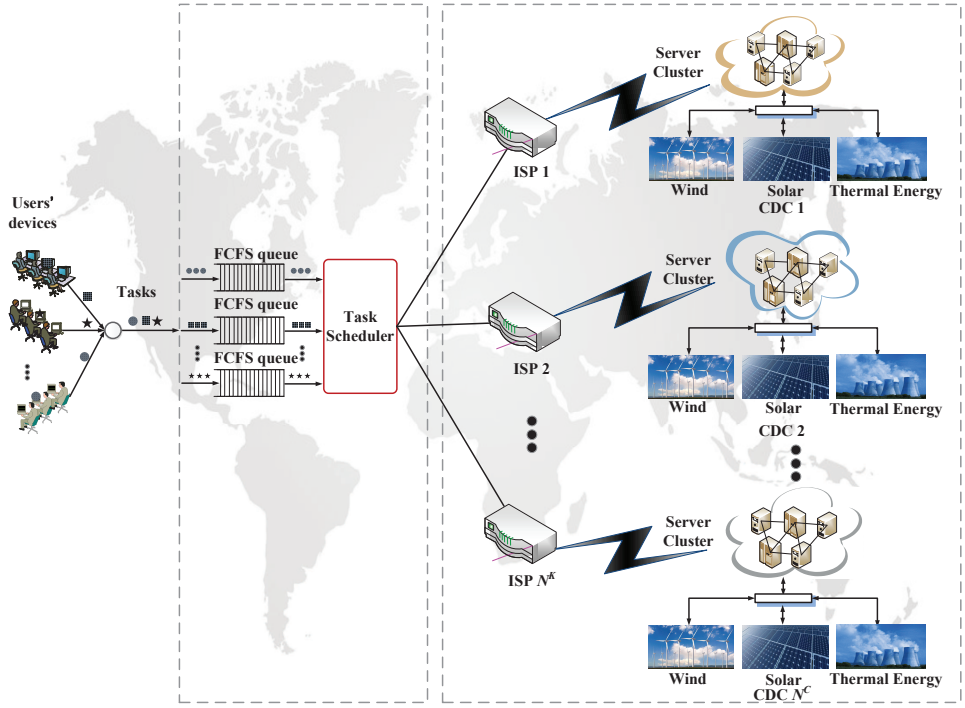


Figure 3.1 System architecture of CDCs.

laptops. CDCs run as follows. In each CDC, users' tasks are executed based on the First-Come-First-Serve (FCFS) policy [190]. Tasks of each application are enqueued into their corresponding FCFS queue. The information about all queues is sent to *Task Scheduler*. Besides, each CDC can obtain electricity from multiple power sources (power grid, solar and wind energy suppliers) and periodically transmit the information to *Task Scheduler*. The information includes prices of power grid, wind speed, solar irradiance, on-site air density, peak (idle) power of each server, *etc.* Based on above information, *Task Scheduler* executes STSRO to jointly specify the optimal allocation of all arriving tasks among multiple available ISPs, and determine the optimal setting of each server in each CDC. Then, the setting information of servers is adopted to configure them in each CDC.

3.2 Problem Formulation

Based on the architecture of CDCs, the cost minimization problem is formulated. The objective is to minimize the total cost of a CDC provider denoted by f_2. f_2 consists

of two parts that are f_{21} and f_{22}, respectively. Here f_{21} denotes ISP bandwidth cost of transmitting data between users and CDCs, and f_{22} denotes CDCs' energy cost brought by the execution of tasks scheduled to CDCs in time slot τ.

$$f_2 = f_{21} + f_{22} \tag{3.1}$$

f_{21} is calculated as:

$$f_{21} = \sum_{k=1}^{\mathcal{N}^K} \left(b_\tau^k \left(\sum_{c=1}^{\mathcal{N}^C} \sum_{n=1}^{\mathcal{N}^A} \left(\lambda_\tau^{k,c,n} \zeta_n L \right) \right) \right) \tag{3.2}$$

In equation (3.2), L denotes the length of each time slot, $\mathcal{N}^K$ denotes the number of available ISPs, $\mathcal{N}^C$ denotes the number of CDCs, and $\mathcal{N}^A$ denotes the number of applications deployed in each CDC. Besides, b_τ^k denotes the unit bandwidth price of ISP k in time slot τ, $\lambda_\tau^{k,c,n}$ denotes the arriving rate of tasks of application n delivered to CDC c through ISP k in time slot τ, and ζ_n denotes the average size of each task of application n.

f_{22} is calculated as:

$$f_{22} = \sum_{c=1}^{\mathcal{N}^C} \left(p_\tau^c \left(\max \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c}, 0 \right) \right) \right) \tag{3.3}$$

In equation (3.3), p_τ^c denotes the price of power grid produced by thermal power generation in CDC c in time slot τ. $\mu_\tau^{c,n}$ denotes the service rate of tasks of application n in CDC c in time slot τ.

There are many existing studies that adopt an $M/M/1$ queueing system to evaluate the performance of each server of each application in existing data centers [110]. Thus, similarly, servers of application n in each CDC are modeled as an $M/M/1/\hat{Q}_n^c/\infty$ queueing system. E_τ^c denotes the total energy consumed by the execution of tasks of all applications in CDC c in time slot τ. Therefore, E_τ^c is calculated by (3.4).

In equation (3.4), $\lambda_{\tau}^{c,n}$ denotes the arriving rate of tasks of application n in CDC c in time slot τ. $\delta_{\tau}^{c,n}$ denotes the loss possibility of tasks of application n in time slot τ. $\overset{\bullet}{\mathbb{N}}_{c,n}$ denotes the number of tasks executed by each switched-on server for application n per minute in CDC c. $\check{\Phi}_{n}^{c}$ and $\hat{\Phi}_{n}^{c}$ denote the idle and peak power of each server for application n in CDC c, respectively. α_c denotes the value of power usage effectiveness of CDC c. $\hat{Q}_{n}^{c}$ denotes the capacity of the task queue of each server for application n in CDC c, and it is the maximum number of tasks that all servers of application n can execute.

$$E_{\tau}^{c}=\sum_{n=1}^{\mathcal{N}^{A}}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau}^{c,n}+\Delta_{c,n}^{4}\lambda_{\tau}^{c,n}\left(1-\delta_{\tau}^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right) \tag{3.4}$$

where

$$\begin{aligned}
\Delta_{c,n}^{5}&=\check{\Phi}_{n}^{c}+\left(\alpha_c-1\right)\hat{\Phi}_{n}^{c}\\
\Delta_{c,n}^{4}&=\hat{\Phi}_{n}^{c}-\check{\Phi}_{n}^{c}\\
\delta_{\tau}^{c,n}&=\frac{1-\rho_{\tau}^{c,n}}{1-(\rho_{\tau}^{c,n})^{\hat{Q}_{n}^{c}+1}}(\rho_{\tau}^{c,n})^{\hat{Q}_{n}^{c}}\\
\rho_{\tau}^{c,n}&=\frac{\lambda_{\tau}^{c,n}}{\mu_{\tau}^{c,n}}
\end{aligned}$$

$\overset{\circ}{E}_{\tau,c}$ denotes the solar energy consumed by tasks of all applications in CDC c in time slot τ. Following the work in [191], $\overset{\circ}{E}_{\tau,c}$ is calculated as:

$$\overset{\circ}{E}_{\tau,c}=\psi_{1c}\psi_{2c}\psi_{\tau,3c}L \tag{3.5}$$

where ψ_{1c} denotes the conversion rate of solar irradiance to electricity in CDC c, ψ_{2c} denotes the active irradiance area of solar panels in CDC c, and $\psi_{\tau,3c}$ denotes the solar irradiance in CDC c in time slot τ.

$\widetilde{E}_{\tau,c}$ denotes the wind energy consumed by the execution of tasks of all applications in CDC c in time slot τ. Following the work in [191], $\widetilde{E}_{\tau,c}$ is calculated

as:

$$\widetilde{E}_{\tau,c}=\frac{1}{2}\phi_{1c}\phi_{2c}\phi_{3c}\left(\phi_{\tau,4c}\right)^3 L \tag{3.6}$$

where ϕ_{1c} denotes the conversion rate of wind to electricity in CDC c, ϕ_{2c} denotes the on-site air density in CDC c, ϕ_{3c} denotes the rotor area of wind turbines in CDC c, and $\phi_{\tau,4c}$ denotes the wind speed in time slot τ in CDC c.

Let $\hat{B}_k$ denote the bandwidth capacity limit of ISP k. The total bandwidth allocated to all tasks that are transmitted through ISP k must be less than or equal to $\hat{B}_k$ in time slot τ, *i.e.*,

$$\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}\left(\lambda_\tau^{k,c,n}\zeta_n\right)\leq\hat{B}_k \tag{3.7}$$

Let $\hat{N}_{c,n}$ denote the number of servers for application n in CDC c. The number of switched-on servers for application n in CDC c is $\frac{\mu_\tau^{c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}}$ in time slot τ. Then,

$$\frac{\mu_\tau^{c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}}\leq\hat{N}_{c,n} \tag{3.8}$$

Let $\hat{\mathbb{E}}_c$ denote the amount of maximum available energy in CDC c. The amount of energy consumed by the execution of tasks of all applications in CDC c must be less than or equal to $\hat{\mathbb{E}}_c$ in time slot τ. Therefore,

$$\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_{c,n}^5\mu_\tau^{c,n}+\Delta_{c,n}^4\lambda_\tau^{c,n}\left(1-\delta_\tau^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right)\leq\hat{\mathbb{E}}_c \tag{3.9}$$

In time slot τ, to guarantee the stability of the task queue of application n in CDC c, $\lambda_\tau^{c,n}$ must be less than or equal to $\mu_\tau^{c,n}$. Therefore,

$$\lambda_\tau^{c,n}=\sum_{k=1}^{\mathcal{N}^K}\lambda_\tau^{k,c,n}<\mu_\tau^{c,n} \tag{3.10}$$

In time slot τ, the sum of $\lambda_{\tau}^{k,c,n}$ must be equal to the arriving rate of tasks of application n, λ_{τ}^{n}. Therefore,

$$\lambda_{\tau}^{n}=\sum_{c=1}^{\mathcal{N}^C}\lambda_{\tau}^{c,n}=\sum_{c=1}^{\mathcal{N}^C}\sum_{k=1}^{\mathcal{N}^K}\lambda_{\tau}^{k,c,n} \tag{3.11}$$

Let $\hat{T}_n$ denote the response time constraint of tasks of application n. In time slot τ, the average response time of tasks of application n in CDC c cannot exceed its constraint $\hat{T}_n$.

$$\frac{\Delta_{\tau,c,n}^{11}}{\mu_{\tau}^{c,n}\left(1-\Delta_{\tau}^{12,c,n}\right)}\leq\hat{T}_n \tag{3.12}$$

where

$$\Delta_{\tau,c,n}^{11}=\frac{\rho_{\tau}^{c,n}}{1-\rho_{\tau}^{c,n}}-\frac{\left(\hat{Q}_n^c+1\right)\left(\rho_{\tau}^{c,n}\right)^{\hat{Q}_n^c+1}}{1-\left(\rho_{\tau}^{c,n}\right)^{\hat{Q}_n^c+1}}$$

$$\Delta_{\tau}^{12,c,n}=\frac{1-\rho_{\tau}^{c,n}}{1-\left(\rho_{\tau}^{c,n}\right)^{\hat{Q}_n^c+1}}$$

$$\rho_{\tau}^{c,n}=\frac{\lambda_{\tau}^{c,n}}{\mu_{\tau}^{c,n}}$$

Based on equations (3.1)–(3.12), the cost minimization problem for CDCs is given as:

$$\textbf{Min}\ \{f_2\}$$

subject to

$$\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}\left(\lambda_{\tau}^{k,c,n}\zeta_n\right)\leq\hat{B}_k \tag{3.13}$$

$$\frac{\mu_{\tau}^{c,n}}{\mathbb{N}_{c,n}^{\bullet}}\leq\hat{N}_{c,n} \tag{3.14}$$

$$\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau}^{c,n}+\Delta_{c,n}^{4}\lambda_{\tau}^{c,n}\left(1-\delta_{\tau}^{c,n}\right)}{\mathbb{N}_{c,n}^{\bullet}}L\right)\leq\hat{\mathbb{E}}_c \tag{3.15}$$

$$\lambda_{\tau}^{c,n}=\sum_{k=1}^{\mathcal{N}^K}\lambda_{\tau}^{k,c,n}<\mu_{\tau}^{c,n} \tag{3.16}$$

$$\lambda_{\tau}^{n}=\sum_{c=1}^{\mathcal{N}^C}\lambda_{\tau}^{c,n}=\sum_{c=1}^{\mathcal{N}^C}\sum_{k=1}^{\mathcal{N}^K}\lambda_{\tau}^{k,c,n} \tag{3.17}$$

$$\frac{\Delta_{\tau,c,n}^{11}}{\mu_{\tau}^{c,n}\left(1-\Delta_{\tau}^{12,c,n}\right)}\leq\hat{T}_n \tag{3.18}$$

$$\lambda_{\tau}^{k,c,n}\geq 0,\mu_{\tau}^{c,n}>0(1\leq k\leq\mathcal{N}^K,1\leq c\leq\mathcal{N}^C,1\leq n\leq\mathcal{N}^A) \tag{3.19}$$

Constraint (3.19) specifies the valid ranges of decision variables including $\lambda_{\tau}^{k,c,n}$ and $\mu_{\tau}^{c,n}$. It is also assumed that time slot-related parameters, *e.g.*, p_{τ}^{c}, b_{τ}^{k}, $\psi_{\tau,3c}$ and $\phi_{\tau,4c}$ are already well predicted with existing prediction algorithms, *e.g.*, stacked autoencoder deep neural network [192]–[196], at the beginning of each time slot τ. The method to solve the constrained optimization problem is described in Section 3.3, and its optimal solution jointly specifies the optimal allocation of all arriving tasks of each application among multiple ISPs, and determines the optimal setting of each server in each CDC. In this way, the cost of a CDC provider is minimized while delay bound constraints of all tasks of each application are strictly met.

3.3 Simulated-annealing-based Bat Algorithm

f_2 in the constrained optimization problem is nonlinear with respect to continuous decision variables. Thus, it is a constrained nonlinear program [197]. This section adopts a penalty function method [198] to transform it into an unconstrained nonlinear program.

$$\underset{\lambda_{\tau}^{k,c,n},\mu_{\tau}^{c,n}}{\mathbf{Min}}\ \left\{\widetilde{f}_2=\overset{\infty}{\mathcal{N}}\mho+f_2\right\} \tag{3.20}$$

where $\widetilde{f}_2$ denotes the augmented objective function, $\overset{\infty}{\mathcal{N}}$ is a large positive constant, and $\mho$ denotes the penalty of the violation of all constraints. Let $\boldsymbol{x}$ denote the vector of decision variables including $\lambda_{\tau}^{k,c,n}$ and $\mu_{\tau}^{c,n}$. Let $\overset{0}{\gamma}_1$ and $\overset{0}{\gamma}_2$ be two positive constants. $\mho$ is obtained as:

$$\mho=\sum_{l=1}^{\mathcal{N}^{\neq}}\left(\max\{0,-g_l(\boldsymbol{x})\}\right)^{\overset{0}{\gamma}_1}+\sum_{m=1}^{\mathbb{N}^{=}}\left|h_m(\boldsymbol{x})\right|^{\overset{0}{\gamma}_2} \tag{3.21}$$

In equation (3.21), each inequality constraint l ($1 \leq l \leq \mathcal{N}^{\neq}$) is transformed into $g_l(\boldsymbol{x}) \geq 0$. If it is not violated, its penalty is 0; otherwise, its penalty is $(\max\{0, -g_l(\boldsymbol{x})\})^{\overset{0}{\gamma_1}}$. Each equality constraint m ($1 \leq m \leq \mathbb{N}^{=}$) is transformed into $h_m(\boldsymbol{x}) = 0$. If it is not violated, its penalty is 0; otherwise, its penalty is $\left|h_m(\boldsymbol{x})\right|^{\overset{0}{\gamma_2}}$. In this way, an unconstrained problem is obtained. There are several typical algorithms, *e.g.*, conjugate gradient method [199] and sequential quadratic programming [200] to solve it. However, they usually depend on first-order or second-order derivatives of $\widetilde{f}_2$, and therefore, they are only suitable for specific optimization problems with these mathematical structures [201]. In addition, their optimization processes are complex and therefore, the quality of their final solutions is not satisfying.

Meta-heuristic algorithms have several advantages, *e.g.*, robustness, wide applicability and easy implementation. Thus, they have been commonly applied to solve different types of complex optimization problems. Each meta-heuristic algorithm has its own pros and cons [202, 203]. As a typical example, SA has been proven to be effective in solving continuous and discrete constrained optimization problems. Its Metropolis acceptance rule allows some moves that worsen the objective function value in order to escape from local optima. It has been demonstrated that SA is able to finally obtain global optima by careful selection of the temperature cooling rate. Its main disadvantage is that its convergence process can be very slow [202]. Besides, bat algorithm (BA) is commonly applied due to its many advantages, *e.g.*, quick convergence. It may easily trap into local optima in its exploration and exploitation processes [202]. Thus, its final solutions are usually low-quality when it is applied to solve large-scale optimization problems with high-dimension search spaces. To improve the efficiency and global search accuracy of BA, this section adopts a hybrid meta-heuristic algorithm named Simulated-annealing-based Bat Algorithm (SBA) by combining SA's Metropolis acceptance rule into BA. In this way, the

diversity of solutions is increased to improve BA's performance, thereby yielding its excellent variant. The pseudo codes of SBA are given in Algorithm 1.

The details of Algorithm 1 are described here. Line 1 initializes positions and velocities of all bats. Let $\mathbf{x}_i^g$ and $\theta_1^{i,g}$ denote the position and velocity of bat i in iteration g. $\mathbf{x}_i^g$ and $\theta_1^{i,g}$ are $\mathbb{N}^D$-dimension vectors that include decision variables. The first $\mathcal{N}^K{*}\mathcal{N}^C{*}\mathcal{N}^A$ elements of each vector store $\lambda_\tau^{k,c,n}$. The next $\mathcal{N}^C{*}\mathcal{N}^A$ elements of each vector store $\mu_\tau^{c,n}$. Therefore, $\mathbb{N}^D{=}\mathcal{N}^C{*}\mathcal{N}^A(\mathcal{N}^K{+}1)$. Line 2 initializes frequency $\theta_{1,i}^B$, pulse rate $\theta_{2,i}^B$ and loudness $\theta_{4,i}^B$ of bat i. Here, $\theta_{1,i}^B{\in}[\check{\theta}_1^B, \hat{\theta}_1^B]$, $\theta_{2,i}^B{\in}[\check{\theta}_2^B, \hat{\theta}_2^B]$, and $\theta_{4,i}^B{\in}[\check{\theta}_4^B, \bar{\theta}_{4,g}^B]$. Line 3 calculates the fitness value of each position, and stores the optimal position in $\overset{*}{\mathbf{x}}$. Line 4 sets the initial temperature θ_2^0. Let $\hat{g}$ denote the maximum number of iterations. Let $|\mathbb{X}|$ denote the number of bats. Let $\theta_{6,i}^B$ denote the adaptation value of position $\mathbf{x}_i$ of bat i. Line 7 calculates $\theta_{6,i}^B$ in current temperature θ_2^g based on (3.22). Let θ_2^g denote the current temperature in iteration t. Line 8 determines an alternative optimal position $\tilde{\mathbf{x}}^B$ selected from positions of all bats with a roulette strategy [204].

$$\mathbf{x}_i{=}\frac{exp^{\frac{-\left(\widetilde{f_2}(\mathbf{x}_i)-\widetilde{f_2}(\overset{*}{\mathbf{x}})\right)}{\theta_2^g}}}{\sum\limits_{i=1}^{|\mathbb{X}|} exp^{\frac{-\left(\widetilde{f_2}(\mathbf{x}_i)-\widetilde{f_2}(\overset{*}{\mathbf{x}})\right)}{\theta_2^g}}} \tag{3.22}$$

Line 8 adjusts frequencies and updates positions and velocities of all bats based on equations (3.23)–(3.25).

$$\theta_{1,i}^B{=}\check{\theta}_1^B + \left(\hat{\theta}_1^B - \check{\theta}_1^B\right) w_1 \tag{3.23}$$

where w_1 denotes a random number drawn from a uniform distribution.

$$\theta_1^{i,g+1}{=}\theta_1^{i,g} + \left(\mathbf{x}_i^g - \tilde{\mathbf{x}}^B\right) \theta_{1,i}^B \tag{3.24}$$

$$\mathbf{x}_i^{g+1}{=}\mathbf{x}_i^g + \theta_1^{i,g+1} \tag{3.25}$$

Line 10 calculates the new fitness value of bat i. Lines 11–14 choose a position ($\mathbf{x}_i^B$) from the best positions for bat i, and produce a new position ($\mathbf{x}_i^1$) around it with random walk if $rand{>}\theta_{2,i}^B$ is met.

$$\mathbf{x}_i^1{=}\mathbf{x}_i^B + w_2\bar{\theta}_{4,g}^B \tag{3.26}$$

where $w_2(w_2{\in}[-1,1])$ is a random number, and $\bar{\theta}_{4,g}^B{=}\frac{1}{|\mathbb{X}|}\sum_{i=1}^{|\mathbb{X}|}\theta_{4,i,g}^B$ is the average loudness of all bats at iteration g.

Lines 15–18 accept the new solution for bat i, increase $\theta_{2,i}^B$ and decrease $\theta_{4,i}^B$ based on equation (3.27) if $rand{<}\theta_{4,i}^B$ and $\widetilde{f}_2(\mathbf{x}_i){<}\widetilde{f}_2(\overset{*}{\mathbf{x}})$.

$$\theta_{4,i,g+1}^B{=}\theta_5^B\theta_{4,i,g}^B, \theta_{2,i,g+1}^B{=}\theta_{2,i}^B[1 - exp^{-\theta_3^B g}] \tag{3.27}$$

Line 19 ranks bats and determines the current optimal position $\overset{*}{\mathbf{x}}$. Finally, the best position of all bats, $\overset{*}{\mathbf{x}}$, is output as the final solution.

3.4 Performance Evaluation

The following experiments evaluate STSRO with real-life data. STSRO is coded and implemented with MATLAB 2017, and it runs on a computer with an Intel Xeon E5-2699AV4 CPU at 2.4 GHz and a 32-GB DDR4 memory.

3.4.1 Parameter Setting

This section adopts realistic tasks of three applications in Google cluster[1] for one day on May 10, 2011. Figure 3.2 shows task arriving rates of three applications (types 1, 2 and 3) that are sampled every 5 minutes. This section adopts realistic power grid prices for one day on May 10, 2011 in capital region of New York, U.S.[2].

Here, $\mathcal{N}^K{=}3$, $\mathcal{N}^C{=}3$ and $\mathcal{N}^A{=}3$. Based on the work in [24], the parameter setting of energy sources including power grid, wind energy and solar energy is shown

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).
[2]http://www.nyiso.com/public/index.jsp (accessed on May 10, 2019).

Algorithm 1 SBA (Simulated-annealing-based Bat Algorithm)

Initialize positions and velocities of all bats
Initialize frequency $\theta_{1,i}^B$, pulse rate $\theta_{2,i}^B$ and loudness $\theta_{4,i}^B$ of bat i
Calculate the fitness value of each position, and store the optimal position in $\overset{*}{\mathbf{x}}$
Set the initial temperature θ_2^0
$g \leftarrow 1$
while $g \leq \hat{g}$ **do**
  Calculate $\theta_{6,i}^B$ of bat i in current temperature θ_2^g based on equation (3.22)
  Determine $\tilde{\mathbf{x}}^B$ from positions of all bats with the roulette strategy
  Adjust frequencies, positions and velocities based on equations (3.23)–(3.25)
  Calculate the new fitness value of bat i
  if $rand > \theta_{2,i}^B$ **then**
    Choose a position from the best positions for bat i
    Produce a new position around it with random walk based on equation (3.26)
  end if
  if $rand < \theta_{4,i}^B$ && $\widetilde{f}_2(\mathbf{x}_i) < \widetilde{f}_2(\overset{*}{\mathbf{x}})$ **then**
    Accept the new solution for bat i with SA's Metropolis acceptance rule
    Increase $\theta_{2,i}^B$ and decrease $\theta_{4,i}^B$ based on equation (3.27)
  end if
  Rank bats and determine currently optimal position x_*
  $\theta_2^{g+1} \leftarrow \theta_2^g \theta_3$
  $g \leftarrow g+1$
end while
Output the best position of all bats, $\overset{*}{\mathbf{x}}$

Table 3.1 Parameter Setting of Energy Sources

	Power grid		Wind energy			Solar energy	
	α_c	$\hat{\mathbb{E}}_c$ (WH)	ϕ_{1c}	ϕ_{3c} (m^2)	ϕ_{2c} (kg/m^3)	ψ_{1c}	ψ_{2c} (m^2)
c=1	1.2	1.7×10^8	0.3	25000	1.225	0.2	15000
c=2	1.4	2.25×10^8	0.375	31250	1.5313	0.25	18750
c=3	1.6	1×10^8	0.45	37500	1.8375	0.3	22500

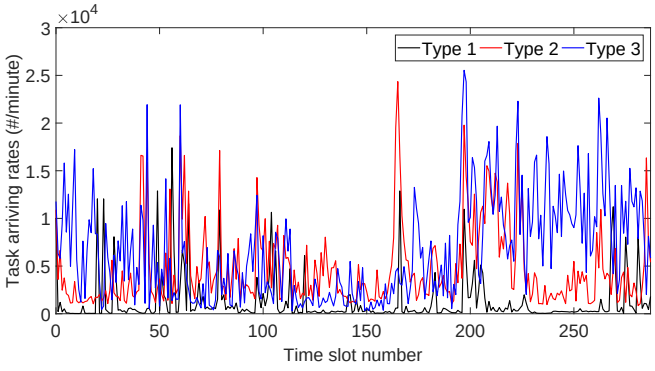


Figure 3.2 Task arriving rates of three applications.

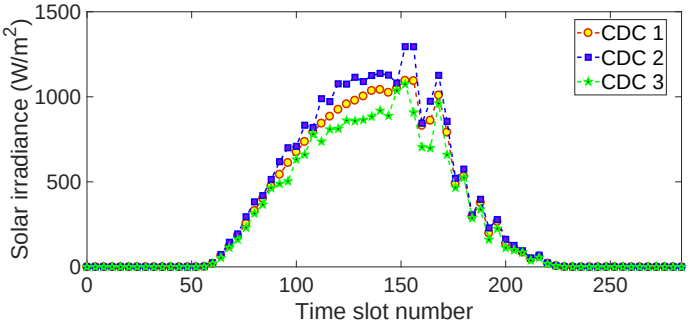


Figure 3.3 Solar irradiance of three CDCs.

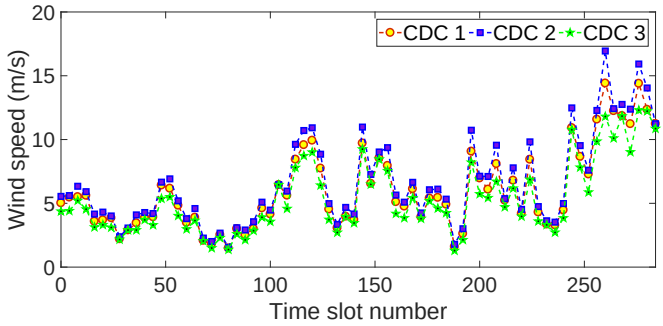


Figure 3.4 Wind speed of three CDCs.

Table 3.2 Parameter Setting of Three CDCs-Part 1.

	$\overset{\bullet}{\mathbb{N}}$ (tasks/second)			$\check{\Phi}_n^c$ (W)			$\hat{\Phi}_n^c$ (W)		
	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
c=1	0.15	0.3	0.6	200	100	50	400	200	100
c=2	0.15	0.3	0.6	250	125	62.5	500	250	125
c=3	0.15	0.3	0.6	300	150	75	600	300	150

Table 3.3 Parameter Setting of Three CDCs-Part 2

	$\hat{Q}_n^c$			$\hat{N}_{c,n}$		
	n=1	n=2	n=3	n=1	n=2	n=3
c=1	50	55	60	1200	1500	1800
c=2	55	60	65	1000	1250	1500
c=3	65	70	75	1200	1500	1800

in Table 3.1. This chapter collects data about solar irradiance[3] and wind speed[4] for one day on May 10, 2011. The solar irradiance and the wind speed in three CDCs are shown in Figures 3.3 and 3.4.

[3]http://www.nrel.gov/midc/srrl_bms/ (accessed on May 10, 2019).
[4]http://www.nrel.gov/midc/nwtc_m2/ (accessed on May 10, 2019).

In addition, $\mathbb{N}$, $\check{\Phi}_n^c$, $\hat{\Phi}_n^c$, $\hat{Q}_n^c$ and $\hat{N}_{c,n}$ are set and given in Tables 3.2 and 3.3. According to the existing study [18], the bandwidth prices of three ISPs are set and shown in Figure 3.5. The power grid prices of three CDCs are shown in Figure 3.6. According to the work in [31], $\hat{B}_1$=4×10^6 (Mbps), $\hat{B}_2$=5×10^6 (Mbps) and $\hat{B}_3$=6×10^6 (Mbps). In addition, ζ_1=8 (Mb), ζ_2=5 (Mb), ζ_3=2 (Mb), $\hat{T}_1$=0.15 (Second), $\hat{T}_2$=0.2 (Second) and $\hat{T}_3$=0.25 (Second).

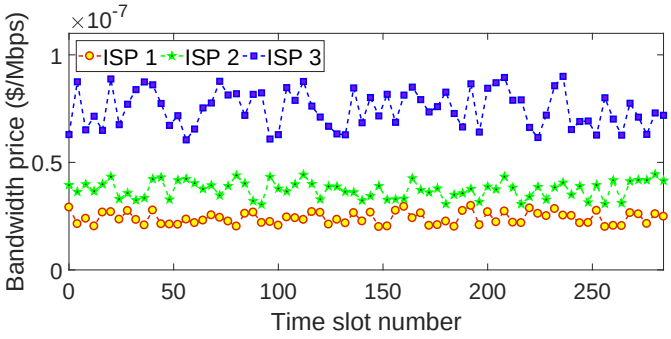


Figure 3.5 Bandwidth prices of three ISPs.

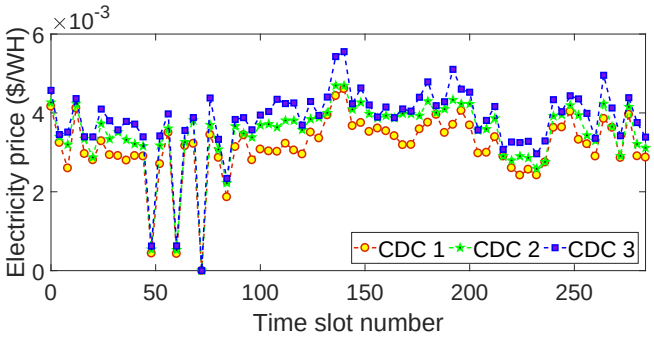


Figure 3.6 Electricity prices of three CDCs.

It is worth noting that many meta-heuristic algorithms are sensitive to the setting of their parameters. Therefore, based on the parameter setting in previous studies [202], the parameter setting of SBA is determined and shown as follows. $\check{\theta}_1^B$=0, $\hat{\theta}_1^B$=100, $\check{\theta}_2^B$=0, $\hat{\theta}_2^B$=1, $\check{\theta}_4^B$=1 and $\hat{\theta}_4^B$=100. In addition, $\hat{g}$=50, $w_1 \in (0,1)$, $\theta_{4,i,g}^B$=θ_3^B=0.9, θ_2^0=10^{12}, $\hat{g}$=10^3 and θ_3=0.975. Besides, $\overset{\infty}{\mathcal{N}}$=10^{20} and $\overset{0}{\gamma}_1$=$\overset{0}{\gamma}_2$=2.

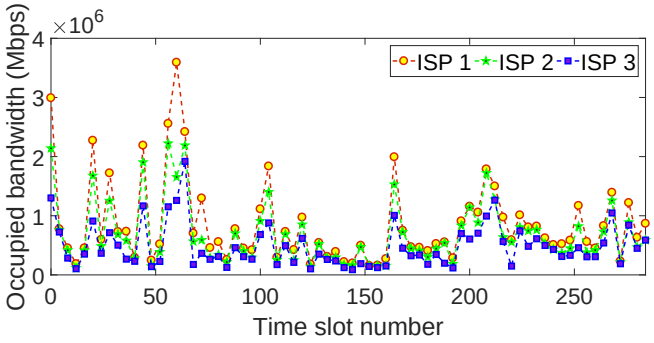


Figure 3.7 Occupied bandwidth of three ISPs.

3.4.2 Experimental Results

The occupied bandwidth of three ISPs connecting to CDCs is considered. As is shown in Figure 3.5, bandwidth prices of three ISPs are different from each other. Figure 3.7 illustrates that the occupied bandwidth of each ISP differs significantly because of the variations in bandwidth prices of ISPs. The reason is that STSRO aims to minimize the total cost of a CDC provider by specifying the optimal allocation of all arriving tasks among multiple ISPs. It is observed that the number of tasks that traverse through ISP 1 is the largest while the number of tasks that traverse ISP 3 is the smallest among three ISPs. The result is consistent with bandwidth prices of three ISPs, *i.e.*, ISP 1's bandwidth price is the smallest while ISP 3's bandwidth price is the largest.

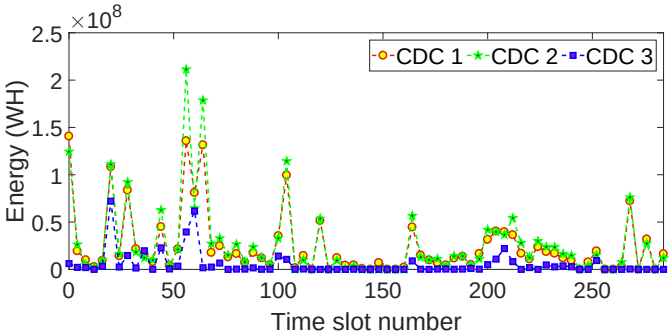


Figure 3.8 Consumption of energy produced by thermal power generation.

The consumption of energy produced by thermal power generation in three CDCs is shown in Figure 3.8. As is shown in Figure 3.6, power grid prices of three

CDCs are also different from each other. Figure 3.8 shows that the consumption of energy produced by thermal power generation in three CDCs varies due to the differences in power grid prices of three CDCs. Similarly, the reason is that STSRO aims to minimize the total cost of a CDC provider by determining the optimal setting of each server in each CDC. It is shown that the consumption of energy produced by thermal power generation in CDC 3 is the largest while that in CDC 1 is the smallest among three CDCs. The result is consistent with power grid prices of three CDCs, *i.e.*, CDC 3's power grid price is the smallest while that of CDC 1 is the largest.

To show the performance of SBA, this chapter compares it with two typical meta-heuristic algorithms including SA and BA. The reasons of choosing them for comparison are described as follows. It is demonstrated that SA can converge to a global optimum in theory by careful design of the cooling rate of temperature because it can smartly escape from a local optimum. Thus, the comparison between SBA and SA can demonstrate the accuracy of SBA's final solution. In addition, it is also shown that BA's convergence speed is quick [202]. Thus, the comparison between SBA and BA can demonstrate SBA's convergence speed.

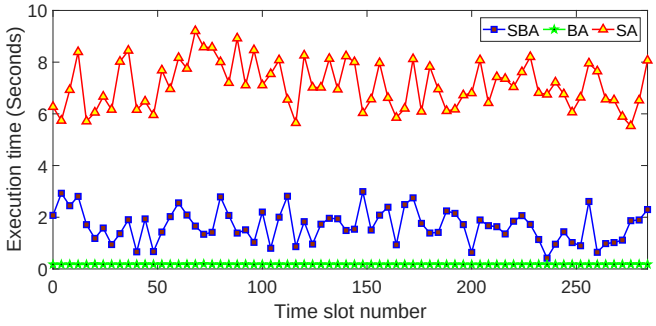


Figure 3.9 Comparison of execution time.

Figure 3.9 shows the comparison of the execution time of SBA, BA and SA. It is shown that the average execution time of SA is 7.13 seconds that is 4.27 times larger than that of SBA, 1.67 seconds, and 39.61 times larger than that of BA, 0.18 seconds. In addition, though BA's execution time is the smallest, this is caused by

its fast trap into a local optimum. Figure 3.10 presents the total cost comparison of each iteration of SBA, BA and SA in time slot 50. Here each iteration in SBA means Lines 7–21 in Algorithm 1. The iterations of BA and SA have similar meaning as that of SBA. Figure 3.11 shows the penalty in each iteration in each time slot, which is calculated based on equation (3.21).

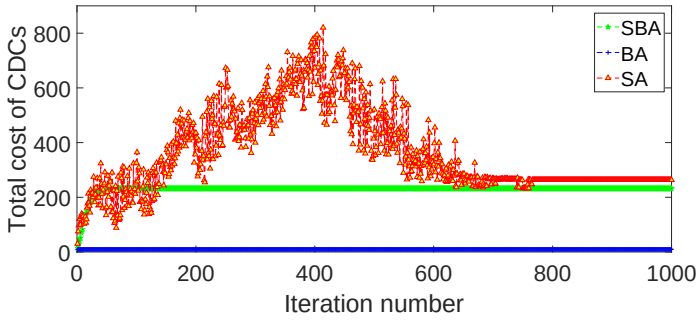


Figure 3.10 Total cost of each iteration in time slot 50.

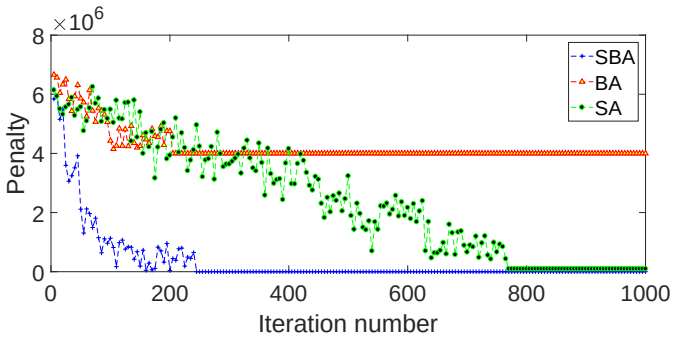


Figure 3.11 Penalty of each iteration.

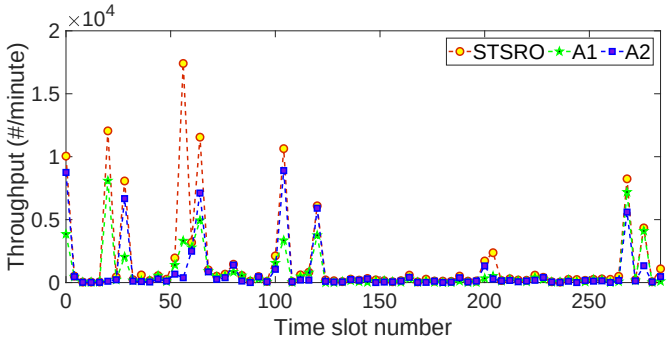
It is shown that BA converges after the least number of iterations compared to SBA and SA. Nevertheless, Figure 3.11 illustrates that the penalty of BA's final solution is extremely large (about 4×10^6). This result shows that its final solution cannot satisfy all the constraints in the formulated optimization problem. BA's final solution is the worst due to its quick trap into a local optimum. SA requires about 766 iterations to converge to its final solution, and the total cost of its final solution is \$263.31. SBA only requires 241 iterations to converge to its final solution, and its corresponding final total cost is \$232.26. SBA decreases the total cost of a CDC

provider by \$31.05 in much fewer iterations than SA. Figure 3.11 presents that the penalty of SBA's final solution is 0. It means that SBA can obtain a high-quality solution meeting all the constraints in the formulated problem. Therefore, Figures 3.9–3.11 show that the adoption of SA's Metropolis acceptance rule in SBA can increase the diversity of solutions, and improve the efficiency and global search accuracy of BA.

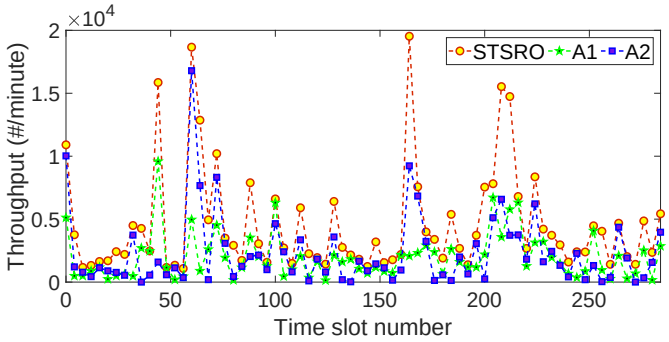
To demonstrate the effectiveness of STSRO, it is compared with two typical intelligent optimization approaches [31, 76] in terms of the total cost and throughput of the CDC provider.

1) Method A1, similar to the cheap-electricity-first scheduling in [76], schedules tasks to CDCs according to the order of their power grid prices. The CDC with the least power grid price executes the largest number of tasks while the one with the highest power grid price executes the least number of tasks.
2) Method A2, similar to renewable energy-first scheduling in [31], schedules tasks to CDCs according to the order of their amount of renewable energy. The CDC with the largest amount of renewable energy executes the largest number of tasks while the one with the least amount executes the least number of tasks.

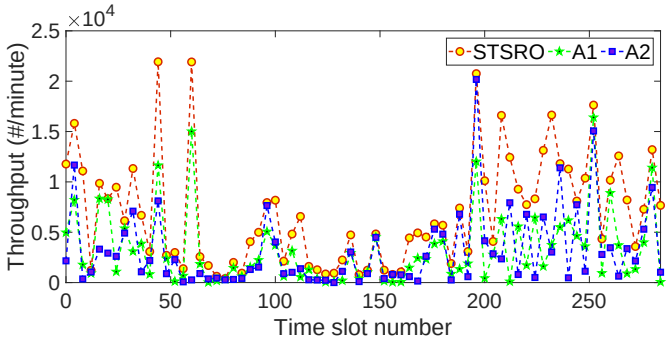
Figure 3.12 compares STSRO with A1 and A2 in terms of the throughput of each application, which is the number of its tasks scheduled in time slot τ. For example, it is observed in Figure 3.12(a) that the throughput of STSRO is larger than those of A1 and A2 in each time slot for application 1. For type 1 application, the throughput of STSRO is larger than those of A1 and A2 by 52.94% and 52.11% on average, respectively. The reason is that the bandwidth capacity of each ISP, the number of available servers for each application and the maximum amount of available energy in each CDC are all limited in each time slot. Therefore, some arriving tasks are refused and not executed to CDCs when using A1 or A2. Thus, Figure 3.12(a) shows that the throughput of CDCs is drastically increased with STSRO. Therefore, Figure 3.8 demonstrates the effectiveness of the proposed STSRO.



(a) Type 1



(b) Type 2



(c) Type 3

Figure 3.12 Throughput of STSRO, A1, and A2.

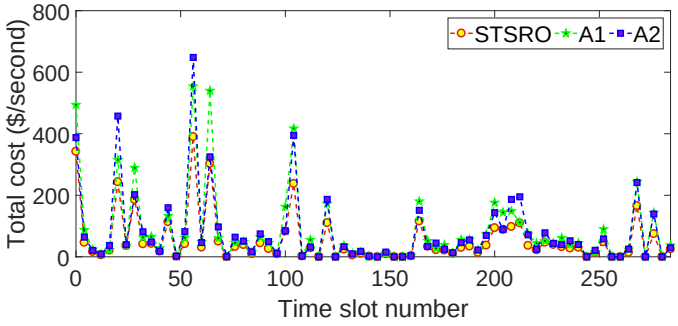


Figure 3.13 Total cost of STSRO, A1, and A2.

What's more, with A1 and A2, an average allocation policy is adopted by ISPs [31, 76]. It means that all tasks are evenly allocated among multiple ISPs. Figure 3.13 illustrates the total cost of STSRO, A1 and A2, respectively. To guarantee the execution performance of tasks, the penalty cost is usually specified in an SLA for task refusal [205, 206]. It is determined after the negotiation between users and a CDC provider, and all refused tasks bring the penalty to a CDC provider. Let ϑ_τ^n denote the penalty paid by their provider if a task of application n is refused in time slot τ. ϑ_τ^n in SLAs is typically larger than the maximum cost caused by the execution of each task of application n among CDCs in time slot τ. It motivates the CDC provider to strictly meet the delay bound constraints of all tasks. The total cost in time slot τ is obtained by calculating the sum of the cost brought by executed tasks in CDCs, and the penalty due to refused tasks in time slot τ. It is shown in Figure 3.13 that compared with A1 and A2, the total cost of STSRO can be reduced by 30.58% and 30.82% on average, respectively. The reason is that STSRO smartly schedules tasks among ISPs and CDCs by jointly considering the spatial diversity in bandwidth prices of ISPs, power grid prices and the availability of renewable green energy in CDCs. Then, the throughput of the CDC provider is drastically increased and its total cost is reduced provided that delay bound constraints of all tasks of each application are strictly met.

3.5 Summary

Cloud data centers (CDCs) need a huge amount of bandwidth and energy to execute multiple applications. Existing studies investigate the energy cost minimization problem in CDCs. The spatial diversity of bandwidth prices of Internet service providers, power grid prices and the availability of renewable green energy brings an opportunity to minimize the total cost of a CDC provider. A nonlinear optimization problem is formulated and solved by the proposed simulated-annealing-based bat algorithm. In this way, this chapter proposes a Spatial Task Scheduling and Resource

Optimization (STSRO) method to minimize the total cost of the CDC provider by exploiting such spatial diversity in CDCs. STSRO can cost-effectively schedule all arriving tasks of heterogeneous applications while strictly meeting their delay bound constraints. Experimental results demonstrate that it drastically increases the throughput and reduces the total cost of the CDC provider in comparison with two recent scheduling methods provided that delay bound constraints of all tasks are strictly met.

CHAPTER 4

GEOGRAPHY-AWARE TASK SCHEDULING FOR PROFIT MAXIMIZATION IN DISTRIBUTED GREEN DATA CENTERS

This chapter presents the details of the proposed Geography-Aware Task Scheduling (GATS) algorithm, and it is organized as follows. A profit maximization problem for cloud data centers (CDCs) is formulated in Section 4.1. Section 4.2 proposes an optimization framework to solve the problem and to realize GATS that maximizes the total profit of a CDC provider. Section 4.3 provides the performance evaluation and results by using real-life data. Section 4.5 concludes this chapter.

4.1 Problem Formulation

Based on the architecture in the system architecture of CDCs illustrated in Figure 3.1, this section gives the formulation of the profit maximization problem for a CDC provider. Section 4.2 presents the proposed optimization framework to solve this problem. In this way, GATS can maximize the profit of the CDC provider in each time slot while guaranteeing arriving tasks of each application to be smartly executed and scheduled within their delay bound constraints.

The profit maximization problem is formulated as follows. For clarity, this section first gives the following several assumptions. Similar to existing studies [21, 76], it is assumed that time-related parameters do not vary within each time slot, and only vary among different time slots. The notations adopted throughout this chapter are explained as follows. Let $\mathcal{N}^A$ denote the number of applications deployed in each CDC. Let L denote the length of each time slot. Let $\lambda_\tau^{c,n}$ and $\mu_\tau^{c,n}$ denote the task arriving rate and service rate of application n in CDC c in time slot τ, respectively. $\delta_\tau^{c,n}$ denotes the loss possibility of tasks of application n in time slot τ in CDC c. Similar to the work in [23, 24], servers of application n in a CDC are modeled as

a $G/D/1$ queueing system. Therefore, tasks of application n in CDC c arrive in a Gaussian process, and $\delta_\tau^{c,n}$ is obtained as:

$$\delta_\tau^{c,n}=\Delta_{\tau,c,n}^1 exp^{-\frac{1}{2}\mathbf{Min}_{\tau\geq 1}\ \Delta_{\tau,c,n}^2} \tag{4.1}$$

where

$$\Delta_{\tau,c,n}^1=\frac{1}{\lambda_\tau^{c,n}\sqrt{2\pi}\sigma_{\tau,c,n}} e^{\frac{(\mu_\tau^{c,n}-\lambda_\tau^{c,n})^2}{2(\sigma_{\tau,c,n})^2}} \int_{\mu_\tau^{c,n}}^{\infty} (t-\mu_\tau^{c,n})\, exp^{-\frac{(t-\lambda_\tau^{c,n})^2}{2(\sigma_{\tau,c,n})^2}} dt \tag{4.2}$$

and for each $\tau\geq 1$,

$$\Delta_{\tau,c,n}^2=\frac{\left(\hat{T}_n\mu_\tau^{c,n}+\tau\left(\mu_\tau^{c,n}-\lambda_\tau^{c,n}\right)\right)^2}{\tau\Gamma_{\tau,c,n}(0)+2\sum_{l=1}^{\tau-1}(\tau-l)\Gamma_{\tau,c,n}(l)} \tag{4.3}$$

Here, $\Gamma_{\tau,c,n}(l)$ denotes the autocovariance of the probability function of the task arriving rate of application n in CDC c, and $\Gamma_{\tau,c,n}(0)=(\sigma_{\tau,c,n})^2$. $\sigma_{\tau,c,n}$ denotes the variance of Gaussian process corresponding to tasks of application n in CDC c. Besides, $\hat{T}_n$ denotes the response time limit of tasks of application n. Let ∇_τ^n denote the payment brought by each task of application n in time slot τ. In addition, to provide the performance assurance for users' tasks, a service level agreement (SLA) is usually signed between a CDC provider and users. In addition, let ϵ_τ^n denote the penalty paid by the provider brought by the refusal of each task of application n in time slot τ. Let $f_1^{c,*}$ denote CDC c's revenue brought by tasks of all applications in time slot τ. Then,

$$f_1^{c,*}=\sum_{n=1}^{\mathcal{N}^A}\left((1-\delta_\tau^{c,n})\,\nabla_\tau^n\lambda_\tau^{c,n}L-\delta_\tau^{c,n}\epsilon_\tau^n\lambda_\tau^{c,n}L\right) \tag{4.4}$$

Let f_1 denote the total revenue of the CDC provider brought by tasks of all applications in time slot τ. Then, f_1 is calculated as:

$$f_1=\sum_{c=1}^{\mathcal{N}^C} f_1^{c,*} \tag{4.5}$$

Let f_2 denote the total cost of the CDC provider in time slot τ. f_2 consists of two major parts that are f_{21} and f_{22}, respectively. f_{21} denotes the Internet service provider (ISP) bandwidth cost of the CDC provider caused by data transmission among CDCs and users in time slot τ. f_{22} denotes the energy cost of the CDC provider brought by all tasks of all applications scheduled to execute in CDCs in time slot τ. Let $\mathcal{N}^K$ denote the number of available ISPs. Then,

$$f_2 = f_{21} + f_{22} \tag{4.6}$$

where

$$f_{21} = \sum_{k=1}^{\mathcal{N}^K} \left(b_\tau^k \left(\sum_{c=1}^{\mathcal{N}^C} \sum_{n=1}^{\mathcal{N}^A} \left(\lambda_\tau^{k,c,n} \zeta_n L \right) \right) \right) \tag{4.7}$$

In equation (4.7), $\mathcal{N}^K$ denotes the number of ISPs connecting to multiple CDCs. Besides, the unit bandwidth price of ISP k in time slot τ is denoted by b_τ^k. The average size of application n's tasks is denoted by ζ_n. In addition, the arriving rate of application n's tasks that are transmitted to CDC c through ISP k in time slot τ is $\lambda_\tau^{k,c,n}$.

f_{22} is obtained as:

$$\begin{aligned} f_{22} &= \sum_{c=1}^{\mathcal{N}^C} \left(p_\tau^c \left(\max \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c}, 0 \right) \right) \right) \\ &= \sum_{c=1}^{\mathcal{N}^C} \left(\max \left(p_\tau^c \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c} \right), 0 \right) \right) \\ &= \sum_{c=1}^{\mathcal{N}^C} (\max(p_\tau^c (\sum_{n=1}^{\mathcal{N}^A} (\frac{\Delta_{c,n}^5 \mu_\tau^{c,n} + \Delta_{c,n}^4 \lambda_\tau^{c,n} (1 - \delta_\tau^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}} L) - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c}), 0)) \end{aligned} \tag{4.8}$$

In equation (4.8), p_τ^c denotes the price of power grid produced by the thermal energy in CDC c in time slot τ. $E_\tau^{c,n}$ denotes the total energy consumed by the execution of tasks of application n in CDC c in time slot τ. E_τ^c denotes the total

energy consumed by the execution of tasks of all applications in CDC c in time slot τ. Then, E_τ^c is obtained as:

$$E_\tau^c = \sum_{n=1}^{\mathcal{N}^A} E_\tau^{c,n} = \sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_{c,n}^5 \mu_\tau^{c,n} + \Delta_{c,n}^4 \lambda_\tau^{c,n} \left(1 - \delta_\tau^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}} L \right) \tag{4.9}$$

where

$$\Delta_{c,n}^5 = \check{\Phi}_n^c + \left(\alpha_c - 1\right) \hat{\Phi}_n^c$$

$$\Delta_{c,n}^4 = \hat{\Phi}_n^c - \check{\Phi}_n^c$$

In equation (4.9), $\overset{\bullet}{\mathbb{N}}_{c,n}$ denotes the number of application n's tasks processed by each switched-on server in one minute in CDC c, and α_c denotes the power usage effectiveness value [24] of CDC c. $\hat{\Phi}_n^c$ and $\check{\Phi}_n^c$ denote the peak and idle power of each server of application n in CDC c, respectively.

Let $\overset{\circ}{E}_{\tau,c}$ and $\widetilde{E}_{\tau,c}$ denote the solar and wind energy consumed by tasks of all applications in CDC c in time slot τ, respectively. Similar to Chapter 3, $\overset{\circ}{E}_{\tau,c}$ and $\widetilde{E}_{\tau,c}$ are calculated with equations (3.5) and (3.6) in the green energy model introduced in Section 3.2, respectively.

Similar to the work in [24], it is assumed that the number of arriving tasks is typically large and the available amount of solar and wind energy is not sufficient to power all servers in CDCs. Then,

$$E_\tau^c \geq \overset{\circ}{E}_{\tau,c} + \widetilde{E}_{\tau,c} \tag{4.10}$$

Therefore, f_{22} is further obtained as:

$$\begin{aligned} f_{22} &= \sum_{c=1}^{\mathcal{N}^C} \left(p_\tau^c \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c} \right) \right) = \sum_{c=1}^{\mathcal{N}^C} \left(p_\tau^c \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c} \right) \right) \\ &= \sum_{c=1}^{\mathcal{N}^C} \left(p_\tau^c \left(\sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_{c,n}^5 \mu_\tau^{c,n} + \Delta_{c,n}^4 \lambda_\tau^{c,n} \left(1 - \delta_\tau^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}} L \right) - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c} \right) \right) \end{aligned} \tag{4.11}$$

Let F_1 denote the profit of the CDC provider in each time slot. F_1 is obtained by calculating the difference between f_1 and f_2, *i.e.*,

$$F_1{=}f_1 - f_2{=}\sum_{c=1}^{\mathcal{N}^C} f_1^{c,*} - f_{21} - f_{22} \tag{4.12}$$

According to (4.11), F_1 is obtained as follows.

$$\begin{aligned} F_1{=}&\sum_{c=1}^{\mathcal{N}^C} f_1^{c,*} - \sum_{c=1}^{\mathcal{N}^C}\left(p_\tau^c\left(E_\tau^c{-}\overset{\circ}{E}_{\tau,c}{-}\widetilde{E}_{\tau,c}\right)\right) - f_{22} \\ {=}&\sum_{c=1}^{\mathcal{N}^C}\left(f_1^{c,*} - p_\tau^c\left(E_\tau^c{-}\overset{\circ}{E}_{\tau,c}{-}\widetilde{E}_{\tau,c}\right)\right) - f_{22} \end{aligned} \tag{4.13}$$

The objective of the problem is to maximize F_1, *i.e.*,

$$\underset{\lambda_\tau^{k,c,n},\ \mu_\tau^{c,n}}{\mathbf{Max}}\left\{\sum_{c=1}^{\mathcal{N}^C}\left(f_1^{c,*} - p_\tau^c\left(E_\tau^c{-}\overset{\circ}{E}_{\tau,c}{-}\widetilde{E}_{\tau,c}\right)\right) - f_{22}\right\}$$

In addition, $\hat{B}_k$ denotes the bandwidth limit of ISP k. Therefore, the total bandwidth occupied by tasks of all applications that are scheduled to transmit through ISP k cannot exceed $\hat{B}_k$ in each time slot τ. Then,

$$\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}\left(\lambda_\tau^{k,c,n}\zeta_n\right) \leq \hat{B}_k \tag{4.14}$$

Let $\hat{N}_{c,n}$ denote the number of available servers for application n in CDC c. Thus, the number of switched-on servers of application n in CDC c in time slot τ must satisfy:

$$\frac{\mu_\tau^{c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}} \leq \hat{N}_{c,n} \tag{4.15}$$

Besides, to provide the stability assurance for a task queue of application n in CDC c in time slot τ, $\lambda_\tau^{c,n}$ is obtained as:

$$\lambda_\tau^{c,n} = \sum_{k=1}^{\mathcal{N}^K}\lambda_\tau^{k,c,n} < \mu_\tau^{c,n} \tag{4.16}$$

Moreover, in each time slot τ, the sum of $\lambda_\tau^{k,c,n}$ should be equal to the task arriving rate of application n, λ_τ^n, *i.e.*,

$$\lambda_\tau^n = \sum_{c=1}^{\mathcal{N}^C} \lambda_\tau^{c,n} = \sum_{c=1}^{\mathcal{N}^C} \sum_{k=1}^{\mathcal{N}^K} \lambda_\tau^{k,c,n} \tag{4.17}$$

Finally, the average number of tasks dropped at each task queue of application n is $\lambda_\tau^{c,n}\delta_\tau^{c,n}$, and it must not exceed its limit, $\mathbb{N}_n^-$, specified in typical SLAs:

$$\lambda_\tau^{c,n}\delta_\tau^{c,n} \leq \mathbb{N}_n^- \tag{4.18}$$

4.2 Optimization Framework

To maximize the profit of the CDC provider, the split of tasks of all applications among multiple ISPs and task service rates of servers in each CDC are jointly optimized and updated within each time slot. Therefore, the following constrained optimization problem is designed and solved at the beginning of each time slot. Based on the constraints (4.10)–(4.18), the profit maximization problem for the CDC provider is formulated as follows.

$$\underset{\lambda_\tau^{k,c,n},\ \mu_\tau^{c,n}}{\mathbf{Max}} \left\{ \sum_{c=1}^{\mathcal{N}^C} \left(f_1^{c,*} - p_\tau^c \left(E_\tau^c - \overset{\circ}{E}_{\tau,c} - \widetilde{E}_{\tau,c} \right) \right) - f_{22} \right\}$$

subject to

$$\sum_{c=1}^{\mathcal{N}^C} \sum_{n=1}^{\mathcal{N}^A} \left(\lambda_\tau^{k,c,n} \zeta_n \right) \leq \hat{B}_k \tag{4.19}$$

$$\frac{\mu_\tau^{c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}} \leq \hat{N}_{c,n} \tag{4.20}$$

$$\sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_{c,n}^5 \mu_\tau^{c,n} + \Delta_{c,n}^4 \lambda_\tau^{c,n} \left(1-\delta_\tau^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}} L \right) \geq \overset{\circ}{E}_{\tau,c} + \widetilde{E}_{\tau,c} \tag{4.21}$$

$$\lambda_\tau^{c,n} = \sum_{k=1}^{\mathcal{N}^K} \lambda_\tau^{k,c,n} < \mu_\tau^{c,n} \tag{4.22}$$

$$\lambda_\tau^n = \sum_{c=1}^{\mathcal{N}^C} \lambda_\tau^{c,n} = \sum_{c=1}^{\mathcal{N}^C} \sum_{k=1}^{\mathcal{N}^K} \lambda_\tau^{k,c,n} \tag{4.23}$$

$$\lambda_\tau^{c,n}\delta_\tau^{c,n} \leq \mathbb{N}_n^- \tag{4.24}$$

$$\lambda_\tau^{k,c,n} \geq 0,\ \mu_\tau^{c,n} > 0 \left(1 \leq k \leq \mathcal{N}^K, 1 \leq c \leq \mathcal{N}^C, 1 \leq n \leq \mathcal{N}^A\right) \tag{4.25}$$

Therefore, the problem is a typical convex optimization one, as is proven in the Appendix. Thus, it can be directly solved by efficient optimization methods, *e.g.*, the interior point method.

Theorem 1: *The formulated optimization problem is a convex optimization problem if*

$$\nabla_{\tau}^{n}+\epsilon_{\tau}^{n}-\frac{p_{\tau}^{c}\Delta_{c,n}^{4}}{\mathbb{N}_{c,n}^{\bullet}}>0,\ \forall c,n \tag{4.26}$$

4.3 Performance Evaluation

This section evaluates GATS by the trace-driven simulation with realistic data. The trace-driven simulation has been commonly and widely used to conduct research in CDCs. Besides, this section adopts the publicly available real-world benchmark workload traces from Google production systems to show the significance of GATS. The workload trace contains data from a cluster of nearly 12.5k machines for a period of 29 days in May 2011. The trace is reliable and widely used in many existing studies in their experiments. Similar to this chapter, many recent studies in top conferences and journals [23, 24, 31, 76] also adopt the public trace to evaluate their methods. Thus, this chapter also chooses it and finds it works well in evaluating GATS. In addition, the simulation is based on the system model, which is widely adopted and deployed in many production data centers, and is also utilized by many existing studies like [23, 24, 31, 76]. The following simulation experiments demonstrate their effectiveness of GATS with realistic trace data from Google cluster[1]. Here GATS is implemented and coded in MATLAB 2017, and it is executed in a server with a 32-GB DDR4 memory and an Intel Xeon E7-8893 v4 processor at 3.20 GHz.

4.3.1 Parameter Setting

The simulation experiments adopt the public realistic tasks corresponding to three typical types of applications (types 1–3) in Google cluster over 24-h-long period on

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).

May 10, 2011. The task arriving rates are calculated every 5 minutes (*i.e.*, L=5 minutes). It is worth noting that the length of each time slot is important for GATS. If the length is too small, *e.g.*, 5 seconds, because many parameters, *e.g.*, prices of power grid and green energy do not dramatically vary within such a short time, the profit of the CDC provider cannot be largely increased. On the other hand, if the length is too large, *e.g.*, 30 minutes, it needs one time slot to update several time-related parameters, *e.g.*, task arriving rates, in the formulated optimization problem. This means that GATS needs to wait for one time slot time (30 minutes) before its execution, but this is obviously unrealistic for most of existing applications in CDCs. Thus, similar to the work in [76], the length of each time slot in the simulation experiments is set to 5 minutes. In addition, the real-life prices of power grid over 24-h-long period on the same day in capital area of New York state, U.S.[2] are adopted. Besides, this chapter adops public data about wind speed[3] and solar irradiance[4] over 24-h-long period on May 10, 2011.

This section considers that three ISPs transmit data between users and three CDCs where three applications run, *i.e.*, $\mathcal{N}^A$=3, $\mathcal{N}^C$=3 and $\mathcal{N}^K$=3. It is worth noting that more ISPs, applications and CDCs can be considered but the simulation results are similar and do not affect the performance comparison. According to the work in [24], the setting of parameters about energy including power grid, solar energy and wind energy is presented in Table 4.1. The power ratings of wind and solar energy are set to 9×10^8 (W) and 1.65×10^8 (W), respectively.

According to the work in [31], ζ_1=8 (Mb), ζ_2=5 (Mb), ζ_3=2 (Mb), $\hat{B}_1$=4×10^6 (Mbps), $\hat{B}_2$=5×10^6 (Mbps) and $\hat{B}_3$=6×10^6 (Mbps). Besides, according to the work in [207], the execution time of each task of three applications is obtained according to the uniform distribution over the time slot of $(0,L)$. According to the work in [25, 207],

[2]http://www.nyiso.com/public/index.jsp (accessed on May 10, 2019).
[3]http://www.nrel.gov/midc/nwtc_m2/ (accessed on May 10, 2019).
[4]http://www.nrel.gov/midc/srrl_bms/ (accessed on May 10, 2019).

Table 4.1 Parameter Setting of Energy

	Power grid	Solar energy		Wind energy		
	α_c	ψ_{1c}	ψ_{2c} (m^2)	ϕ_1	ϕ_{3c} (m^2)	ϕ_{2c} (kg/m^3)
c=1	1.6	0.3	22500	0.45	37500	1.838
c=2	1.4	0.25	18750	0.375	31250	1.531
c=3	1.2	0.2	15000	0.3	25000	1.225

Table 4.2 Parameter Setting of Three CDCs

	$\overset{\bullet}{\mathbb{N}}_{c,n}$ (tasks/second)			$\check{\Phi}^c_n$ (W)			$\hat{\Phi}^c_n$ (W)			$\hat{N}_{c,n}$		
	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
c=1	0.15	0.3	0.6	300	150	75	600	300	150	1200	1500	1800
c=2	0.15	0.3	0.6	250	125	62.5	500	250	125	1000	1250	1500
c=3	0.15	0.3	0.6	200	100	50	400	200	100	1200	1500	1800

the execution prices of tasks per unit time in each time slot are set and obtained according to the uniform distribution over the ranges of (0.24,0.48), (0.16,0.32), and (0.08,0.16), respectively. In this way, ∇^n_τ is calculated.

Additionally, an SLA is typically specified between the CDC provider and users to guarantee the performance of tasks, and it defines the penalty cost of a refused task of each application [205]. Therefore, the penalty cost is brought to the CDC provider due to the refused tasks. In typical SLAs, ϵ^n_τ is larger than revenue brought by each task of application n in time slot τ. Here, ϵ^n_τ=1.5×∇^n_τ. According to the work in [23, 24], $\mathbb{N}^-_1$=2, $\mathbb{N}^-_2$=8, and $\mathbb{N}^-_3$=16. In addition, according to the work in [24], $\overset{\bullet}{\mathbb{N}}_{c,n}$, $\check{\Phi}^c_n$, $\hat{\Phi}^c_n$, $\hat{\mathbb{Q}}_n$ and $\hat{N}_{c,n}$ are set in Table 4.2.

4.3.2 Experimental Results

Figure 4.1 illustrates the occupied bandwidth of three ISPs. It shows that each ISP's occupied bandwidth differs from each other and this is caused by the differences in bandwidth prices of three ISPs presented in Figure 3.5. This is because GATS aims to maximize the profit of the CDC provider by determining the optimal split of tasks of all applications among multiple available ISPs. It is shown in Figure 4.1 that the number of tasks of all applications that are scheduled to transmit through ISP 1 is the greatest while the number of tasks of all applications that are scheduled to

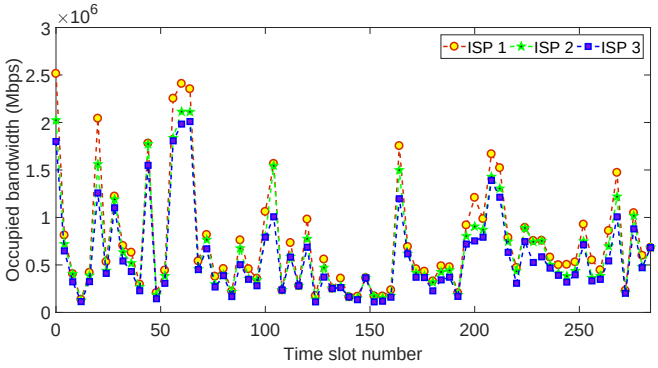


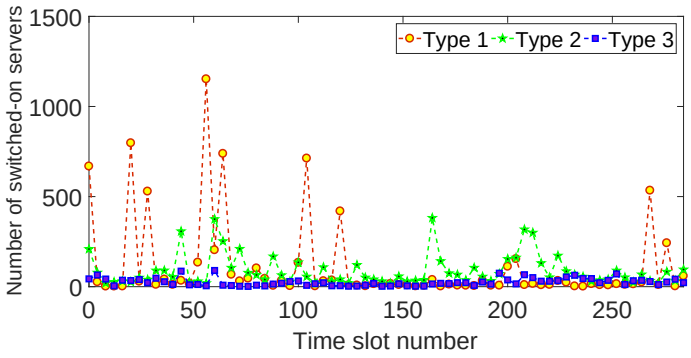
Figure 4.1 Occupied bandwidth of three ISPs.

transmit through ISP 3 is the least among three ISPs. The experimental result keeps consistent with bandwidth prices of three ISPs, *i.e.*, the bandwidth price of ISP 3 is the greatest while that of ISP 1 is the least.

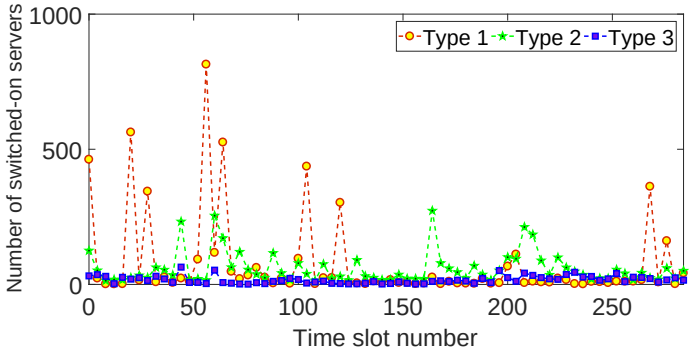
Figure 4.2 shows the number of switched-on servers in each CDC. It is observed that the number of switched-on servers for each type in three CDCs is less than or equal to its corresponding limit. Besides, it is observed that the number of switched-on servers for the same application in three CDCs varies a lot. For instance, the number of switched-on servers for type 2 in CDC 1 is much greater than that for type 2 in CDCs 2 and 3. This reason is explained as follows. The available amount of renewable energy including solar irradiance and wind speed in CDC 1 is the largest among three CDCs. In addition, the price of power grid of CDC 1 is the least among three CDCs. Similarly, the number of switched-on servers for type 2 in CDC 3 is much less than that for type 2 in CDC 2. The reason is that the available amount of renewable energy in CDC 3 is less than that in CDC 2, and the price of power grid of CDC 3 is larger than that in CDC 2.

To prove the effectiveness of the proposed GATS, this section further compares it with two typical task scheduling methods [31, 76] in terms of the throughput and profit of the CDC provider.

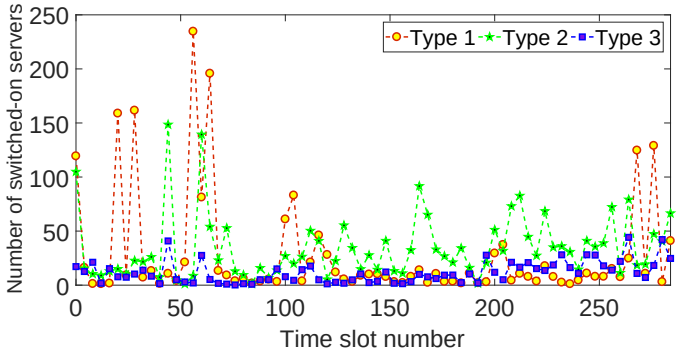
1) A1, like a cheap-electricity-first task scheduling method proposed in [76], schedules tasks of all applications to CDCs based on the ascending order of prices of power grid. It means that the CDC with the highest price of power



(a) CDC 1



(b) CDC 2

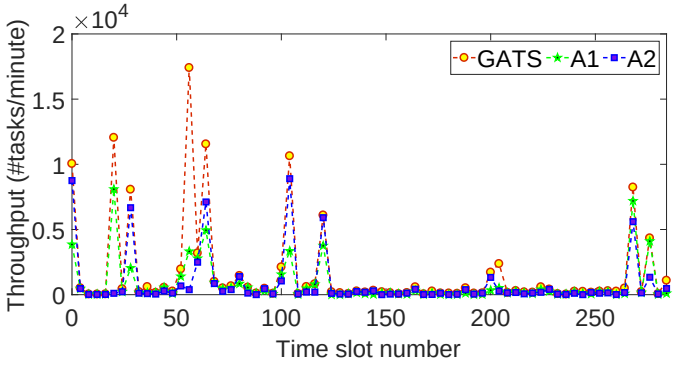


(c) CDC 3

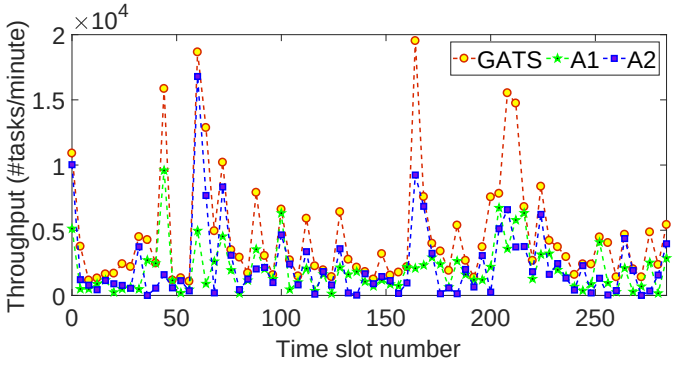
Figure 4.2 Number of switched-on servers in each CDC.

grid receives the smallest number of tasks while the CDC with the lowest price of power grid receives the most tasks.

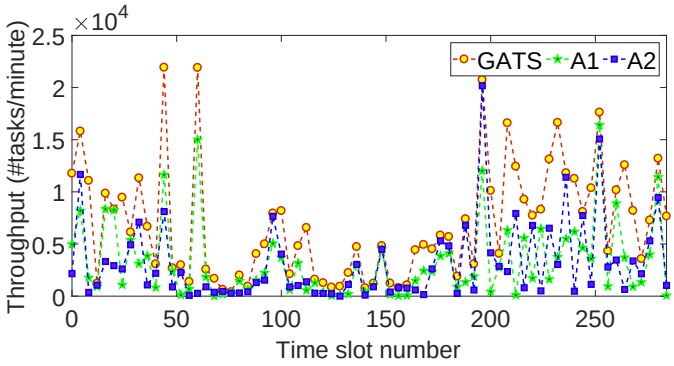
2) A2, like a green energy-first task scheduling method in [31], schedules tasks of all applications to CDCs based on the descending order of the amount of green energy. It means that the CDC with the smallest amount of green energy receives the smallest number of tasks while the CDC with the largest amount of green energy receives the most tasks.



(a) Type 1



(b) Type 2



(c) Type 3

Figure 4.3 Throughput of GATS, A1 and A2.

Figure 4.3 presents the comparison of GATS with A1 and A2 in terms of the throughput that is the number of tasks of all applications scheduled to three CDCs in each time slot. It is shown that the throughput of GATS is higher than those of A1 and A2 in each time slot for all applications. For example, for type 3 application, GATS's throughput is higher than those of A1 and A2 by 50.54% and 49.19% on average, respectively. This is because the available amount of green energy, ISP

bandwidth capacity and the number of servers for all applications in each CDC are all constrained. Consequently, some tasks have to be rejected by CDCs in A1 and A2, and this result demonstrates that CDCs' throughput is very significantly increased by using GATS.

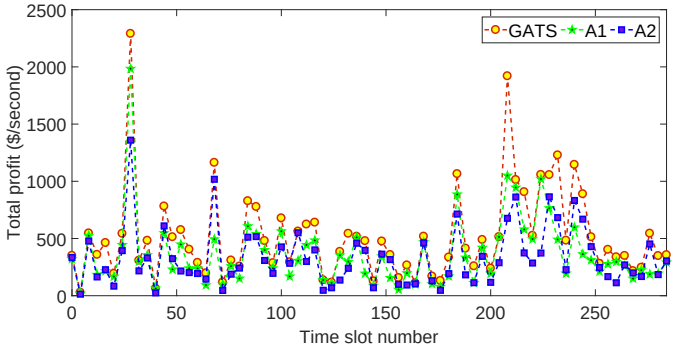


Figure 4.4 Total profit of GATS, A1 and A2.

In addition, the typical average allocation mechanism is used by ISPs in A1 and A2, and it means that tasks of all applications are equally scheduled among multiple available ISPs. Figure 4.4 shows the total profit of GATS, A1, and A2, respectively. In order to guarantee the performance of tasks of each application, an SLA is usually signed among users and the CDC provider to specify the penalty brought to the CDC provider if a task of each application n is rejected by the CDCs in time slot τ [77]. In order to motivate the CDC provider to strictly satisfy delay bound constraints of tasks of all applications, the penalty in SLAs is usually greater than the maximum profit brought by the execution of each task in CDCs in time slot. In Figure 4.4, the total profit in time slot τ is obtained by calculating the difference between the profit of tasks executed in CDCs and the penalty brought by the rejected tasks in time slot τ. Figure 4.4 shows that compared to A1 and A2, the total profit of GATS is increased by 33.74% and 33.93% on average, respectively. This is because GATS intelligently schedules more tasks among multiple available ISPs and CDCs by jointly investigating and using spatial diversity in prices of power grid of CDCs, ISP bandwidth prices and the availability of green energy in CDCs.

4.4 Appendix

To demonstrate the convexity of the formulated optimization problem, this section needs to prove the following three points [208]:

1) The objective function, *i.e.*, f_1-f_2, is concave.

2) The equality constraints are affine.

3) The inequality constraints are convex.

4.4.1 Proof of Objective Function

As p_τ^c, $\overset{\circ}{E}_{\tau,c}$, and $\widetilde{E}_{\tau,c}$ are given parameters whose values are known at the beginning of each time slot. Therefore,

$$\begin{aligned}
\Delta_{\tau,c}^3 &= f_1^{c,*} - p_\tau^c E_\tau^c \\
&= \sum_{n=1}^{\mathcal{N}^A} \left(\left(1-\delta_\tau^{c,n}\right) \nabla_\tau^n \lambda_\tau^{c,n} L - \delta_\tau^{c,n} \epsilon_\tau^n \lambda_\tau^{c,n} L \right) - p_\tau^c E_\tau^c \\
&= \sum_{n=1}^{\mathcal{N}^A} \left(\left(\left(1-\delta_\tau^{c,n}\right) \nabla_\tau^n \lambda_\tau^{c,n} L - \delta_\tau^{c,n} \epsilon_\tau^n \lambda_\tau^{c,n} L \right) - p_\tau^c E_\tau^{nc} \right) \\
&= \sum_{n=1}^{\mathcal{N}^A} \Delta_{\tau,c,n}^3
\end{aligned} \tag{4.27}$$

where

$$\begin{aligned}
\Delta_{\tau,c,n}^3 &= \left(\left(1-\delta_\tau^{c,n}\right) \nabla_\tau^n \lambda_\tau^{c,n} L - \delta_\tau^{c,n} \epsilon_\tau^n \lambda_\tau^{c,n} L \right) - p_\tau^c \frac{\Delta_{c,n}^5 \mu_\tau^{c,n} + \Delta_{c,n}^4 \lambda_\tau^{c,n} \left(1-\delta_\tau^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}} L \\
&= \lambda_\tau^{c,n} L \left(\nabla_\tau^n - \frac{p_\tau^c \Delta_{c,n}^4}{\overset{\bullet}{\mathbb{N}}_{c,n}} \right) - \frac{p_\tau^c \Delta_{c,n}^5 \mu_\tau^{c,n} L}{\overset{\bullet}{\mathbb{N}}_{c,n}} - \delta_\tau^{c,n} \lambda_\tau^{c,n} L \left(\nabla_\tau^n + \epsilon_\tau^n - \frac{p_\tau^c \Delta_{c,n}^4}{\overset{\bullet}{\mathbb{N}}_{c,n}} \right)
\end{aligned} \tag{4.28}$$

It is worth noting that $\lambda_\tau^{c,n} L \left(\nabla_\tau^n - \frac{p_\tau^c \Delta_{c,n}^4}{\overset{\bullet}{\mathbb{N}}_{c,n}} \right)$ and $-\frac{p_\tau^c \Delta_{c,n}^5 \mu_\tau^{c,n} L}{\overset{\bullet}{\mathbb{N}}_{c,n}}$ are concave with respect to $\lambda_\tau^{c,n}$ and $\mu_\tau^{c,n}$, respectively. Therefore, based on Theorem 1, the objective function, *i.e.*, f_1-f_2, is concave if the following function is convex,

$$\begin{aligned}
\Delta_{c,n}^6 &= \lambda_\tau^{c,n} \delta_\tau^{c,n} \\
&= \lambda_\tau^{c,n} \Delta_{\tau,c,n}^1 exp^{-\frac{1}{2} \mathbf{Min}_{\tau \geq 1} \Delta_{\tau,c,n}^2}
\end{aligned} \tag{4.29}$$

Besides, as exp^{-x} is non-increasing with respect to a given variable x, the following function has to be convex.

$$\Delta^6_{c,n}=\underset{\tau\geq 1}{\mathbf{Max}}\left(\lambda^{c,n}_{\tau}\Delta^1_{\tau,c,n}exp^{-\frac{1}{2}\Delta^2_{\tau,c,n}}\right) \tag{4.30}$$

The max function preserves the convexity [208], and therefore, $\Delta^6_{c,n}$ is convex if the following function,

$$\Delta^6_{\tau,c,n}=\lambda^{c,n}_{\tau}\Delta^1_{\tau,c,n}exp^{-\frac{1}{2}\Delta^2_{\tau,c,n}} \tag{4.31}$$

is convex.

Let $t{=}r\sigma_{\tau,c,n}{+}\lambda^{c,n}_{\tau}$. Therefore,

$$\begin{aligned}
&\int_{\mu^{c,n}_{\tau}}^{\infty}(t-\mu^{c,n}_{\tau})\,exp^{-\frac{(t-\lambda^{c,n}_{\tau})^2}{2(\sigma_{\tau,c,n})^2}}dt\\
&{=}\sigma_{\tau,c,n}\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}(r\chi^{c,n}_{\tau}+\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})\,exp^{-\frac{r^2}{2}}dr\\
&{=}\sigma_{\tau,c,n}\left[\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}r\sigma_{\tau,c,n}exp^{-\frac{r^2}{2}}dr+\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}(\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})\,exp^{-\frac{r^2}{2}}dr\right]\\
&{=}\sigma_{\tau,c,n}\left[\left[-\sigma_{\tau,c,n}exp^{-\frac{r^2}{2}}\right]_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}+\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}(\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})\,exp^{-\frac{r^2}{2}}dr\right]\\
&{=}(\sigma_{\tau,c,n})^2\left[exp^{-\frac{(\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau})^2}{2(\sigma_{\tau,c,n})^2}}+\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}\frac{(\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})}{\sigma_{\tau,c,n}}exp^{-\frac{r^2}{2}}dr\right]\\
&{=}(\sigma_{\tau,c,n})^2\left[exp^{-\frac{(\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau})^2}{2(\sigma_{\tau,c,n})^2}}+\frac{\sqrt{2\pi}\,(\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})}{\sigma_{\tau,c,n}}\int_{\frac{\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}}^{\infty}\frac{1}{\sqrt{2\pi}}exp^{-\frac{r^2}{2}}dr\right]\\
&{=}(\sigma_{\tau,c,n})^2[exp^{-\frac{(\mu^{c,n}_{\tau}-\lambda^{c,n}_{\tau})^2}{2(\sigma_{\tau,c,n})^2}}+\frac{\sqrt{2\pi}\,(\lambda^{c,n}_{\tau}-\mu^{c,n}_{\tau})}{\sigma_{\tau,c,n}}(1{-}\Delta^7_{\tau,c,n}(\frac{\mu^{c,n}_{\tau}{-}\lambda^{c,n}_{\tau}}{\sigma_{\tau,c,n}}))]
\end{aligned} \tag{4.32}$$

In equation (4.32), $\Delta^7_{\tau,c,n}$ denotes the function of standard normal distribution. Therefore, based on equation (4.32), $\Delta^1_{\tau,c,n}$ can be rewritten as:

$$\begin{aligned}\Delta^1_{\tau,c,n}&=\frac{\sigma_{\tau,c,n}}{\lambda^{c,n}_\tau\sqrt{2\pi}}\left[1+exp^{\frac{(\mu^{c,n}_\tau-\lambda^{c,n}_\tau)^2}{2(\sigma_{\tau,c,n})^2}}\int_{\frac{\mu^{c,n}_\tau-\lambda^{c,n}_\tau}{\sigma_{\tau,c,n}}}^{\infty}\frac{(\lambda^{c,n}_\tau-\mu^{c,n}_\tau)}{\sigma_{\tau,c,n}}exp^{-\frac{r^2}{2}}dr\right]\\&=\frac{\sigma_{\tau,c,n}}{\lambda^{c,n}_\tau\sqrt{2\pi}}\left[1-\frac{(\mu^{c,n}_\tau-\lambda^{c,n}_\tau)}{\sigma_{\tau,c,n}}exp^{\frac{(\mu^{c,n}_\tau-\lambda^{c,n}_\tau)^2}{2(\sigma_{\tau,c,n})^2}}\int_{\frac{\mu^{c,n}_\tau-\lambda^{c,n}_\tau}{\sigma_{\tau,c,n}}}^{\infty}exp^{-\frac{r^2}{2}}dr\right]\end{aligned}\tag{4.33}$$

Let $\Delta^8_{\tau,c,n}=\frac{\sigma_{\tau,c,n}}{\lambda^{c,n}_\tau}$. Then, equation (4.33) can be written as:

$$\Delta^1_{\tau,c,n}=\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[1-\frac{1}{\Delta^8_{\tau,c,n}}\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)exp^{\frac{\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)^2}{2\left(\Delta^8_{\tau,c,n}\right)^2}}\int_{\frac{1}{\Delta^8_{\tau,c,n}}\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)}^{\infty}exp^{-\frac{r^2}{2}}dr\right]\tag{4.34}$$

Similarly, $\Delta^2_{\tau,c,n}$ can be written as:

$$\begin{aligned}\Delta^2_{\tau,c,n}&=\frac{\left(\hat{T}_n\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}+\tau\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)\right)^2}{\tau\frac{\Gamma_{\tau,c,n}(0)}{(\lambda^{c,n}_\tau)^2}+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{\Gamma_{\tau,c,n}(\ell)}{(\lambda^{c,n}_\tau)^2}}=\frac{\left(\hat{T}_n\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}+\tau\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)\right)^2}{\tau\left(\frac{\sigma_{\tau,c,n}}{\lambda^{c,n}_\tau}\right)^2+2\sum_{\ell=1}^{\tau-\ell}(\tau-l)\frac{\Gamma_{\tau,c,n}(\ell)}{(\lambda^{c,n}_\tau)^2}}\\&=\frac{\left(\hat{T}_n\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}+\tau\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)\right)^2}{\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{\Gamma_{\tau,c,n}(\ell)}{(\lambda^{c,n}_\tau)^2}}=\frac{\left(\hat{T}_n\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}+\tau\left(\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}-1\right)\right)^2}{\Delta^9_{\tau,c,n}}\end{aligned}\tag{4.35}$$

where $\Delta^9_{\tau,c,n}=\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{\Gamma_{\tau,c,n}(\ell)}{(\lambda^{c,n}_\tau)^2}$.

Then, based on equations (4.31), (4.34) and (4.35), it is worth noting that $\Delta^6_{\tau,c,n}$ is the perspective function of the function (4.36) that is obtained by replacing $\frac{\mu^{c,n}_\tau}{\lambda^{c,n}_\tau}$ with $\mu^{c,n}_\tau$ in equations (4.34) and (4.35). Then, (4.36) is obtained as:

$$\hat{\Delta}^6_{\tau,c,n}=\hat{\Delta}^1_{\tau,c,n}exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}}\tag{4.36}$$

$$\hat{\Delta}^1_{\tau,c,n}=\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[1-\frac{1}{\Delta^8_{\tau,c,n}}\left(\mu^{c,n}_\tau-1\right)exp^{\frac{\left(\mu^{c,n}_\tau-1\right)^2}{2\left(\Delta^8_{\tau,c,n}\right)^2}}\int_{\frac{1}{\Delta^8_{\tau,c,n}}\left(\mu^{c,n}_\tau-1\right)}^{\infty}exp^{-\frac{r^2}{2}}dr\right]\tag{4.37}$$

$$\hat{\Delta}^{2}_{\tau,c,n}=\frac{\left(\hat{T}_n\mu^{c,n}_{\tau}+\tau\left(\mu^{c,n}_{\tau}-1\right)\right)^{2}}{\Delta^{9}_{\tau,c,n}}=\frac{\left(\left(\hat{T}_n+\tau\right)\left(\mu^{c,n}_{\tau}-1\right)+\hat{T}_n\right)^{2}}{\Delta^{9}_{\tau,c,n}} \tag{4.38}$$

According to the work in [208], $\Delta^{6}_{\tau,c,n}$ is convex if $\hat{\Delta}^{6}_{\tau,c,n}$ is convex. Therefore, the convexity of $\hat{\Delta}^{6}_{\tau,c,n}$ has to be proven. Based on the work in [208], $\hat{\Delta}^{6}_{\tau,c,n}$ is convex if $(\hat{\Delta}^{6}_{\tau,c,n})^{''}\geq 0$ holds. For simplicity, β is defined as:

$$\beta=\frac{1}{\Delta^{8}_{\tau,c,n}}\left(\mu^{c,n}_{\tau}-1\right) \tag{4.39}$$

Therefore,

$$\hat{\Delta}^{6}_{\tau,c,n}\left(\beta\right)=\hat{\Delta}^{1}_{\tau,c,n}(\beta)exp^{-\frac{1}{2}\hat{\Delta}^{2}_{\tau,c,n}(\beta)} \tag{4.40}$$

where

$$\hat{\Delta}^{1}_{\tau,c,n}(\beta)=\frac{\Delta^{8}_{\tau,c,n}}{\sqrt{2\pi}}\left[1-\beta exp^{\frac{\beta^{2}}{2}}\int_{\beta}^{\infty}exp^{-\frac{r^{2}}{2}}dr\right] \tag{4.41}$$

$$\hat{\Delta}^{2}_{\tau,c,n}(\beta)=\frac{\left(\left(\hat{T}_n+\tau\right)\Delta^{8}_{\tau,c,n}\beta+\hat{T}_n\right)^{2}}{\Delta^{9}_{\tau,c,n}} \tag{4.42}$$

Therefore, $(\hat{\Delta}^{6}_{\tau,c,n}(\beta))^{'}$ is calculated as:

$$\begin{aligned}(\hat{\Delta}^{6}_{\tau,c,n}(\beta))^{'}&=(\hat{\Delta}^{1}_{\tau,c,n}(\beta))^{'}exp^{-\frac{1}{2}\hat{\Delta}^{2}_{\tau,c,n}(\beta)}-\frac{1}{2}\hat{\Delta}^{1}_{\tau,c,n}(\beta)exp^{-\frac{1}{2}\hat{\Delta}^{2}_{\tau,c,n}(\beta)}(\hat{\Delta}^{2}_{\tau,c,n}(\beta))^{'}\\&=exp^{-\frac{1}{2}\hat{\Delta}^{2}_{\tau,c,n}(\beta)}\left[(\hat{\Delta}^{1}_{\tau,c,n}(\beta))^{'}-\frac{1}{2}\hat{\Delta}^{1}_{\tau,c,n}(\beta)(\hat{\Delta}^{2}_{\tau,c,n}(\beta))^{'}\right]\end{aligned} \tag{4.43}$$

Then, $(\hat{\Delta}^{6}_{\tau,c,n}(\beta))^{''}$ is calculated as:

$$\begin{aligned}(\hat{\Delta}^{6}_{\tau,c,n}(\beta))^{''}=exp^{-\frac{1}{2}\hat{\Delta}^{2}_{\tau,c,n}(\beta)}[&(\hat{\Delta}^{1}_{\tau,c,n}(\beta))^{''}-(\hat{\Delta}^{1}_{\tau,c,n}(\beta))^{'}(\hat{\Delta}^{2}_{\tau,c,n}(\beta))^{'}+\\&\frac{1}{4}\left((\hat{\Delta}^{2}_{\tau,c,n}(\beta))^{'}\right)^{2}\hat{\Delta}^{1}_{\tau,c,n}(\beta)-\frac{1}{2}\hat{\Delta}^{1}_{\tau,c,n}(\beta)(\hat{\Delta}^{2}_{\tau,c,n}(\beta))^{''}]\end{aligned} \tag{4.44}$$

$(\hat{\Delta}^1_{\tau,c,n}(\beta))'$ is obtained as:

$$(\hat{\Delta}^1_{\tau,c,n}(\beta))'=\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[-exp^{\frac{\beta^2}{2}}(1+\beta^2)\int_\beta^\infty exp^{-\frac{r^2}{2}}dr+\beta\right] \tag{4.45}$$

Based on equations (4.37), the following equation is obtained.

$$\int_\beta^\infty exp^{-\frac{r^2}{2}}dr=\frac{1-\frac{\sqrt{2\pi}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\Delta^8_{\tau,c,n}}}{\beta exp^{\frac{\beta^2}{2}}} \tag{4.46}$$

Thus, $(\hat{\Delta}^1_{\tau,c,n}(\beta))'$ can be rewritten as:

$$\begin{aligned}(\hat{\Delta}^1_{\tau,c,n}(\beta))'&=\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[-exp^{\frac{\beta^2}{2}}(1+\beta^2)\left(\frac{1-\frac{\sqrt{2\pi}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\Delta^8_{\tau,c,n}}}{\beta exp^{\frac{\beta^2}{2}}}\right)+\beta\right]\\&=\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[\frac{\beta^2+1}{\beta}\left(\frac{\sqrt{2\pi}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\Delta^8_{\tau,c,n}}-1\right)+\beta\right]\\&=\left(\beta+\frac{1}{\beta}\right)\hat{\Delta}^1_{\tau,c,n}(\beta)-\frac{1}{\beta}\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\end{aligned} \tag{4.47}$$

Based on equations (4.41) and (4.47), $(\hat{\Delta}^1_{\tau,c,n}(\beta))''$ can be rewritten as:

$$\begin{aligned}(\hat{\Delta}^1_{\tau,c,n}(\beta))''&=\left(\left(\beta+\frac{1}{\beta}\right)\hat{\Delta}^1_{\tau,c,n}(\beta)\right)'+\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}\beta^2}\\&=\left(1-\frac{1}{\beta^2}\right)\hat{\Delta}^1_{\tau,c,n}(\beta)+\left(\beta+\frac{1}{\beta}\right)(\hat{\Delta}^1_{\tau,c,n}(\beta))'+\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}\beta^2}\\&=\left(1-\frac{1}{\beta^2}\right)\hat{\Delta}^1_{\tau,c,n}(\beta)+\left(\beta+\frac{1}{\beta}\right)\left(\left(\beta+\frac{1}{\beta}\right)\hat{\Delta}^1_{\tau,c,n}(\beta)-\frac{1}{\beta}\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\right)+\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}\beta^2}\\&=\hat{\Delta}^1_{\tau,c,n}(\beta)\left(\beta^2+3\right)-\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\end{aligned} \tag{4.48}$$

Then, $(\hat{\Delta}^6_{\tau,c,n}(\beta))''$ is calculated as:

$$\begin{aligned}(\hat{\Delta}^6_{\tau,c,n}(\beta))''=&exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}(\beta)}[(\beta^2+3)\hat{\Delta}^1_{\tau,c,n}(\beta)-\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}+\hat{\Delta}^1_{\tau,c,n}(\beta)\frac{\left((\hat{\Delta}^2_{\tau,c,n}(\beta))'\right)^2}{4}\\&-\hat{\Delta}^1_{\tau,c,n}(\beta)\frac{(\hat{\Delta}^2_{\tau,c,n}(\beta))''}{2}-(\hat{\Delta}^2_{\tau,c,n}(\beta))'[\frac{\beta^2+1}{\beta}\hat{\Delta}^1_{\tau,c,n}(\beta)-\frac{\hat{\Delta}^1_{\tau,c,n}(\beta)}{\sqrt{2\pi}\beta}]]\\=&\frac{exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}(\beta)}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\beta}[\beta^3-(\hat{\Delta}^2_{\tau,c,n}(\beta))'(\beta^2+1)+\beta(3+\frac{\left((\hat{\Delta}^2_{\tau,c,n}(\beta))'\right)^2}{4}\\&-\frac{(\hat{\Delta}^2_{\tau,c,n}(\beta))''}{2})+\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}\hat{\Delta}^1_{\tau,c,n}(\beta)}\left((\hat{\Delta}^2_{\tau,c,n}(\beta))'-\beta\right)] \qquad (4.49)\end{aligned}$$

Besides, according to the definition of the autocovariance function [209], $\Gamma_{\tau,c,n}(\ell)\leq\Gamma_{\tau,c,n}(0)=(\sigma_{\tau,c,n})^2$. Therefore,2

$$\begin{aligned}\Delta^9_{\tau,c,n}=&\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{\Gamma_{\tau,c,n}(l)}{(\lambda^{c,n}_\tau)^2}\\\leq&\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{\Gamma_{\tau,c,n}(0)}{(\lambda^{c,n}_\tau)^2}\\\leq&\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\frac{(\sigma_{\tau,c,n})^2}{(\lambda^{c,n}_\tau)^2}\\\leq&\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\left(\Delta^8_{\tau,c,n}\right)^2\\\leq&\tau\left(\Delta^8_{\tau,c,n}\right)^2+2\left(\Delta^8_{\tau,c,n}\right)^2\sum_{\ell=1}^{\tau-1}(\tau-\ell)\\\leq&\tau\left(\Delta^8_{\tau,c,n}\right)^2+\left(\Delta^8_{\tau,c,n}\right)^2\tau(\tau-1) \qquad (4.50)\\\leq&\tau^2\left(\Delta^8_{\tau,c,n}\right)^2 \qquad (4.51)\end{aligned}$$

Therefore,

$$\tau\left(\Delta^8_{\tau,c,n}\right)^2\leq\Delta^9_{\tau,c,n}\leq\tau^2\left(\Delta^8_{\tau,c,n}\right)^2 \qquad (4.52)$$

In addition, based on equation (4.42), $(\hat{\Delta}^2_{\tau,c,n}(\beta))'$ and $(\hat{\Delta}^2_{\tau,c,n}(\beta))''$ can be calculated as:

$$(\hat{\Delta}^2_{\tau,c,n}(\beta))' = \frac{\left(\left(\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}\beta+\hat{T}_n\right)^2\right)'}{\Delta^9_{\tau,c,n}}$$
$$= \frac{2\left(\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}\beta+\hat{T}_n\right)\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}}{\Delta^9_{\tau,c,n}} \tag{4.53}$$

$$(\hat{\Delta}^2_{\tau,c,n}(\beta))'' = \frac{2\left(\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}\beta+\hat{T}_n\right)\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2}{\Delta^9_{\tau,c,n}} \tag{4.54}$$

Therefore,

$$\frac{\left((\hat{\Delta}^2_{\tau,c,n}(\beta))'\right)^2}{4} - \frac{(\hat{\Delta}^2_{\tau,c,n}(\beta))''}{2}$$
$$= -\frac{\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2}{\Delta^9_{\tau,c,n}} + \frac{\left(\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}\beta+\hat{T}_n\right)^2\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2}{\left(\Delta^9_{\tau,c,n}\right)^2} \tag{4.55}$$
$$= \frac{1}{\Delta^9_{\tau,c,n}}\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2\left[\frac{1}{\Delta^9_{\tau,c,n}}\left(\left(\hat{T}_n+\tau\right)\Delta^8_{\tau,c,n}\beta+\hat{T}_n\right)^2-1\right]$$
$$> \frac{1}{\Delta^9_{\tau,c,n}}\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2\left[\frac{1}{\Delta^9_{\tau,c,n}}\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2 v^2-1\right]$$
$$\geq \frac{1}{\Delta^9_{\tau,c,n}}\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2\left[\frac{\left(\hat{T}_n+\tau\right)^2\beta^2}{\tau^2}-1\right]$$
$$> \frac{1}{\Delta^9_{\tau,c,n}}\left(\hat{T}_n+\tau\right)^2\left(\Delta^8_{\tau,c,n}\right)^2\left(\beta^2-1\right)$$
$$\geq \frac{\left(\hat{T}_n+\tau\right)^2}{\tau^2}\left(\beta^2-1\right)$$
$$> \beta^2-1 \tag{4.56}$$

According to the lower and upper bounds pointed in [210], the following equation is obtained.

$$\frac{2}{\beta+\sqrt{\beta^2+4}} \leq exp^{\frac{\beta^2}{2}} \int_{\beta}^{\infty} exp^{\frac{-r^2}{2}} dr \leq \frac{2}{\beta+\sqrt{\beta^2+\frac{8}{\pi}}} \tag{4.57}$$

Therefore,

$$0<\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[1-\frac{2\beta}{\beta+\sqrt{\beta^2+\frac{8}{\pi}}}\right] \leq \hat{\Delta}^1_{\tau,c,n}(\beta) \leq \frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}}\left[1-\frac{2\beta}{\beta+\sqrt{\beta^2+4}}\right] \tag{4.58}$$

$$\frac{\Delta^8_{\tau,c,n}}{\sqrt{2\pi}\hat{\Delta}^1_{\tau,c,n}(\beta)} \geq \frac{\left(\beta+\sqrt{\beta^2+4}\right)^2}{4} = \frac{2\beta^2+4+2\beta\sqrt{\beta^2+4}}{4} > = \frac{4\beta^2+4}{4} = \beta^2+1 \tag{4.59}$$

Then, following (4.49), (4.56), and (4.58), the following equation is obtained.

$$\begin{aligned}
\left(\hat{\Delta}^6_{\tau,c,n}(\beta)\right)'' > & \frac{exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}(\beta)}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\beta}[\beta^3-(\hat{\Delta}^2_{\tau,c,n}(\beta))'(\beta^2+1) \\
& +\beta(3+(\beta^2-1))+\left(\beta^2+1\right)\left((\hat{\Delta}^2_{\tau,c,n}(\beta))'-\beta\right)] \\
= & \frac{exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}(\beta)}\hat{\Delta}^1_{\tau,c,n}(\beta)}{\beta}\left(\beta^3+\beta\right) \\
= & exp^{-\frac{1}{2}\hat{\Delta}^2_{\tau,c,n}(\beta)}\hat{\Delta}^1_{\tau,c,n}(\beta)\left(\beta^2+1\right) \geq 0
\end{aligned} \tag{4.60}$$

Consequently, $\hat{\Delta}^6_{\tau,c,n}$ is convex, and therefore, $\Delta^6_{\tau,c,n}$ and $\Delta^6_{c,n}$ are convex. Then, the objective function, *i.e.*, f_1-f_2, is concave.

4.4.2 Proof of Constraints

It is worth noting that constraint (4.23) is affine with respect to $\lambda^{c,n}_{\tau}$. What's more, it is worth noting that constraints (4.19), (4.20), (4.22), (4.23) and (4.25) are all linear. Therefore, the convexity of constraints (4.21) and (4.24) has to be proven.

First, the convexity of constraint (4.21) has to be proven as follows. It is transformed into the following form.

$$\overset{\circ}{E}_{\tau,c}+\widetilde{E}_{\tau,c}+\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau}^{c,n}+\Delta_{c,n}^{4}\lambda_{\tau}^{c,n}\left(1-\delta_{\tau}^{c,n}\right)}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right)\leq 0 \tag{4.61}$$

According to equation (4.29), $\lambda_{\tau}^{c,n}\delta_{\tau}^{c,n}$ is convex, and therefore, constraint (4.21) is convex. Similarly, constraint (4.24) is also convex. Therefore, the proof is complete and the formulated optimization problem is convex.

4.5 Summary

Cloud data centers (CDCs) need a huge amount of bandwidth and energy cost to execute multiple applications. Several existing studies investigate and utilize a profit maximization problem in CDCs. Nevertheless, spatial variations of Internet service provider (ISP) bandwidth prices, availability of green energy, prices of power grid, and revenue brought by the execution of tasks bring a big challenge and opportunity for engineers to maximize the total profit of the CDC provider. This chapter formulates the problem as a convex optimization one and uses an interior point method to solve it. The resulting geography-aware task scheduling approach can maximize the total profit of the CDC provider by jointly and optimally determining the allocation of tasks of all applications among multiple ISPs and task service rates of servers in CDCs. It intelligently schedules the tasks of all applications in comparison with two typical task scheduling methods. Real-life data-driven simulation results show that it increases the total profit and throughput of the CDC provider compared to two typical state-of-the-art task scheduling approaches.

CHAPTER 5

SPATIO-TEMPORAL TASK SCHEDULING FOR HETEROGENEOUS DELAY-TOLERANT APPLICATIONS IN CDCS

This chapter presents the details of the proposed Spatio-Temporal Task Scheduling (STTS) algorithm, and it is organized as follows. Section 5.1 describes the architecture of cloud data centers (CDCs). Section 5.2 formulates an energy cost minimization problem for a CDC provider. The proposed STTS method is presented in Section 5.3. Section 5.4 introduces the details of a Genetic Simulated-annealing-based Particle Swarm Optimization (GSPSO) algorithm. Section 5.5 presents the experimental results by using real-life data. Section 5.6 concludes this chapter.

5.1 Architecture of CDCs

A CDC provider manages multiple CDCs located in different geographical locations and provides heterogeneous applications to global users. Figure 5.1 illustrates a CDC architecture. Each CDC usually manages a server cluster consisting of a great number of servers ranging from several hundreds to several thousands. Besides, to guarantee availability, robustness and response time, each application is deployed in multiple available CDCs. Similar to the work in [110], replicas including data and programs for each application are copied and shared across all CDCs. Therefore, each application and its data are strictly consistent with each other, and this means that tasks can be independently executed within each CDC.

Users' tasks are sent through multiple devices, *e.g.*, smart phones, computers, laptops and servers. Component *Classifier* classifies tasks according to the types of their applications and specifies the arriving rate of each application's tasks. Then, arriving tasks of the same application are enqueued into a separate First-Come-First-Served (FCFS) queue. Figure 5.1 shows that the information of each FCFS queue

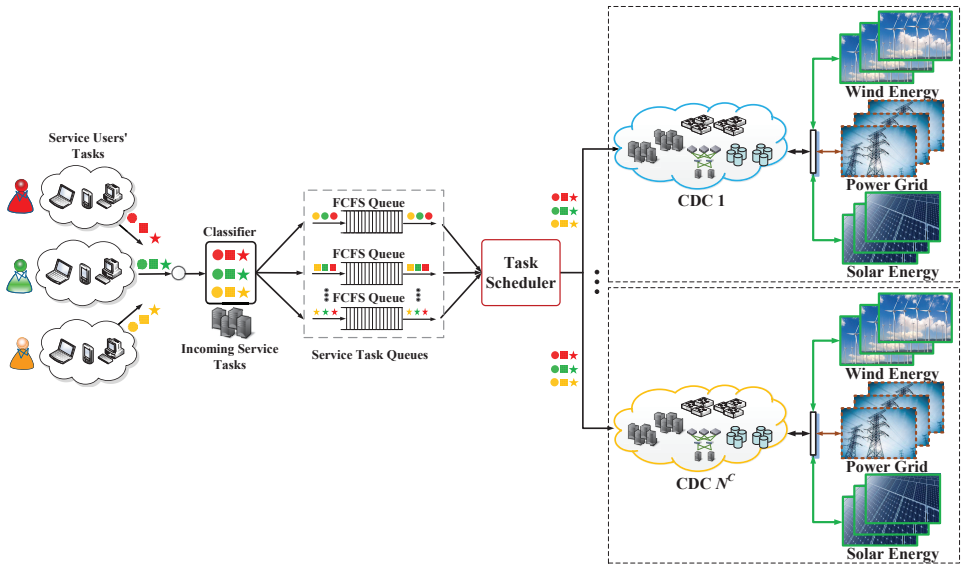


Figure 5.1 Architecture of CDCs.

is transmitted to *Task Scheduler* that is the focus of this chapter. STTS runs in it, and periodically determines the optimal strategy that schedules all arriving tasks to CDCs while strictly meeting their delay bound constraints.

As is shown in Figure 5.1, each CDC obtains energy from three types of energy suppliers, *i.e.*, power grid, solar panels and wind turbines. Their information includes prices of power grid, conversion rate of wind to electricity, density of on-site air, rotor area of wind turbines, wind speed, active irradiance area of solar panels, conversion rate of solar irradiance to electricity, solar irradiance, *etc.* It is periodically obtained and transmitted to *Task scheduler*.

The energy cost minimization problem for a CDC provider is formulated as a nonlinear constrained optimization one in Section 5.2. It is solved by GSPSO in each iteration of STTS. According to the final solution of GSPSO, STTS intends to minimize energy cost of a CDC provider while strictly meeting tasks' delay bound constraints for their tasks. Based on the scheduling strategy, *Task Scheduler* jointly specifies the optimal split of all arriving tasks among CDCs, and determines the optimal setting of servers in each CDC in each time slot. Then, each server in each CDC is configured accordingly.

5.2 Problem Formulation

Based on the architecture in Figure 5.1, this section formulates the energy cost minimization problem for the CDC provider. Similar to the work in [21, 117], each CDC is modeled as a discrete-time system that evolves with time slots. Nowadays, a growing number of high-performance servers are widely deployed in large-scale data centers. Hence, similar to the work in [21], it is reasonable to assume that each task can completely finish its execution in a single time slot.

5.2.1 Delay Bound Constraint

Let $\lambda_{\tau+b}^{c,n}$ denote the task arriving rate of application n in CDC c in τ+b ($0 \leq b \leq B_n$). Let $\mu_{\tau+b}^{c,n}$ denote the service rate of servers of application n in CDC c in time slot τ+b. $\mathcal{N}^A$ denotes the number of applications in each CDC. Let $\Upsilon = max_{n \in \{1,2,\cdots,\mathcal{N}^A\}}(B_n)$. The task service rate of each application n denotes the rate at which its tasks are scheduled to execute in CDCs and removed from its own FCFS queue in each time slot. Let $\overset{+}{\mathbb{N}}_{\tau,c,n}$ denote the number of tasks of application n in CDC c that have arrived at the start of each time slot τ. $\overset{+}{\lambda}_{\ell,c,n}$ ($1 \leq \ell \leq \tau$) denotes the accumulated task arriving rate of application n in CDC c in ℓ. $\delta_{\ell}^{c,n}$ denotes the loss possibility of application n's tasks in CDC c in time slot ℓ. Let L denote the length of each time slot. Let $\overset{\diamond}{\mathbb{N}}_{\tau,c,n}$ denote the accumulated number of scheduled tasks of application n in CDC c by time slot τ. Then, $\overset{+}{\mathbb{N}}_{\tau,c,n}$ and $\overset{\diamond}{\mathbb{N}}_{\tau,c,n}$ are obtained as:

$$\overset{+}{\mathbb{N}}_{\tau,c,n} = \sum_{\ell=1}^{\tau} \lambda_{\ell}^{c,n} L \tag{5.1}$$

$$\overset{\diamond}{\mathbb{N}}_{\tau,c,n} = \sum_{\ell=1}^{\tau} \left(\overset{+}{\lambda}_{\ell,c,n} (1 - \delta_{\ell}^{c,n}) L \right) \tag{5.2}$$

$\lambda_{\tau+b}^{\dagger,c,n}$ denotes the remaining task arriving rate of application n in CDC c in τ+b. To meet application n's delay bound constraint, all its tasks in $\tau - B_n$ or before must be scheduled to CDCs. Let $\lambda_{\tau+b}^{c,n}$ denote the task arriving rate of application n in

CDC c in $\tau+b$. Therefore, $\lambda_{\ell}^{\dagger,c,n}=0$ ($\ell \leq \tau-B_n-1$). Then,

$$\overset{+}{\lambda}_{\tau+b,c,n}=\lambda_{\tau+b}^{c,n}+\sum_{\ell=\tau+b-B_n}^{\tau+b-1}\lambda_{\ell}^{\dagger,c,n} \tag{5.3}$$

Similar to the work in [110], this chapter models application n's servers in CDC c as an $M/M/1/\hat{Q}_n^c/\infty$ queueing system. Let $\hat{Q}_n^c$ denote the capacity limit of the task queue of application n in CDC c, and it is the maximum number of tasks that application n's servers in CDC c can execute. Then,

$$\delta_{\tau+b}^{c,n}=\begin{cases}\dfrac{1-\frac{\overset{+}{\lambda}_{\tau+b,c,n}}{\mu_{\tau+b}^{c,n}}}{1-\left(\frac{\overset{+}{\lambda}_{\tau+b,c,n}}{\mu_{\tau+b}^{c,n}}\right)^{\hat{Q}_n^c+1}}\left(\frac{\overset{+}{\lambda}_{\tau+b,c,n}}{\mu_{\tau+b}^{c,n}}\right)^{\hat{Q}_n^c} & \mu_{\tau+b}^{c,n}>0, \\ 1 & \mu_{\tau+b}^{c,n}=0. \end{cases} \tag{5.4}$$

To guarantee the stability of the task queue of application n in CDC c, $\overset{+}{\lambda}_{\tau+b,c,n}$ must be less than its task service rate in time slot $\tau+b$, *i.e.*,

$$\overset{+}{\lambda}_{\tau+b,c,n}<\mu_{\tau+b}^{c,n},\ 0\leq b\leq B_n \tag{5.5}$$

Let $\lambda_{\tau+b}^{n}$ denote the task arriving rate of application n in time slot $\tau+b$. In this slot, the sum of task arriving rates of application n in CDCs must be equal to its task arriving rate. $\mathcal{N}^C$ denotes the number of CDCs. Then,

$$\lambda_{\tau+b}^{n}=\sum_{c=1}^{\mathcal{N}^C}\lambda_{\tau+b}^{c,n},\ 0\leq b\leq B_n \tag{5.6}$$

It is worth noting that tasks of application n in CDC c must have been scheduled to execute in CDCs within their delay bound B_n. This means that all tasks of application n in CDC c that arrive in time slot $\tau+b$ must have been scheduled from $\tau+b$ to $\tau+b+B_n$. Therefore, all tasks of application n in CDC c in time slot $\tau+b-B_n$

or before must have been executed by time slot τ+b. Then,

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}+\sum_{\ell=\tau-B_n}^{\tau-B_n+b}\left(\lambda_\ell^{c,n}L\right)\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,c,n}+\sum_{\ell=\tau}^{\tau+b}\left(\overset{+}{\lambda}_{\ell,c,n}\left(1-\delta_\ell^{c,n}\right)L\right),0\leq b\leq B_n \quad (5.7)$$

In constraint (5.7), it is worth noting that in time slot τ+b, $\overset{+}{\lambda}_{\ell,c,n}$ (τ+b+1$\leq\ell\leq\tau$+b+B_n) can be well predicted by many existing prediction algorithms, *e.g.*, stacked auto-encoder [211] and deep neural networks [212]. Nevertheless, they are not the focus of this chapter and therefore, at the start of time slot τ+b, $\overset{+}{\lambda}_{\ell,c,n}$ is set to $\lambda_\ell^{c,n}$, *i.e.*, $\overset{+}{\lambda}_{\ell,c,n}$=$\lambda_\ell^{c,n}$ (τ+b+1$\leq\ell\leq\tau$+b+B_n).

In addition, $\overset{+}{\mathbb{N}}_{\tau,c,n}$ is the sum of $\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}$ and $\sum_{\ell=\tau-B_n}^{\tau}\left(\lambda_\ell^{c,n}L\right)$. Then, $\overset{+}{\mathbb{N}}_{\tau,c,n}$ is obtained as follows.

$$\overset{+}{\mathbb{N}}_{\tau,c,n}=\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}+\sum_{\ell=\tau-B_n}^{\tau}\left(\lambda_\ell^{c,n}L\right) \quad (5.8)$$

The delay bound constraint of tasks of application n in CDC c that arrive in time slot τ is strictly met if they are scheduled to execute in CDCs from τ to τ+B_n. At the start of time slot τ, the number of tasks of application n in CDC c scheduled in τ+b ($0\leq b\leq B_n$) is $\overset{+}{\lambda}_{\tau+b,c,n}(1-\delta_{\tau+b}^{c,n})L$. Then, the number of tasks of application n in CDC c executed by τ+B_n is obtained as:

$$\overset{\diamond}{\mathbb{N}}_{\tau+B_n,c,n}=\overset{\diamond}{\mathbb{N}}_{\tau-1,c,n}+\sum_{\ell=\tau}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,c,n}(1-\delta_\ell^{c,n})L\right)$$

Thus, due to conservation of application n's tasks, $\overset{+}{\mathbb{N}}_{\tau,c,n}$ should equal $\overset{\diamond}{\mathbb{N}}_{\tau+B_n,c,n}$, *i.e.*, $\overset{+}{\mathbb{N}}_{\tau,c,n}$=$\overset{\diamond}{\mathbb{N}}_{\tau+B_n,c,n}$. Then,

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}+\sum_{\ell=\tau-B_n}^{\tau}\left(\lambda_\ell^{c,n}L\right)=\overset{\diamond}{\mathbb{N}}_{\tau-1,c,n}+\sum_{\ell=\tau}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,c,n}(1-\delta_\ell^{c,n})L\right) \quad (5.9)$$

Therefore, constraints (5.7) and (5.9) can ensure that delay bound constraints of tasks of all applications must be met.

5.2.2 Power Consumption Model

Then, the power consumption model adopted in CDCs is introduced here. Similar to the work in [110], it is assumed that servers of each application are homogeneous, and the energy consumed by servers of the same application is identical. Let $\overset{\bullet}{\mathbb{N}}_{c,n}$ denote the number of tasks of application n executed by each of its switched-on servers in each minute in CDC c. $\overset{o}{N}_{\tau+b,c,n}$ denotes the number of switched-on servers of application n in CDC c in time slot $\tau+b$. Then, $\overset{o}{N}_{\tau+b,c,n}$ is obtained as:

$$\overset{o}{N}_{\tau+b,c,n}=\frac{\mu^{c,n}_{\tau+b}}{\overset{\bullet}{\mathbb{N}}_{c,n}} \tag{5.10}$$

Let $\hat{N}_{c,n}$ denote the maximum available number of servers of application n in CDC c. Then, $\overset{o}{N}_{\tau+b,c,n}$ needs to be less than or equal to $\hat{N}_{c,n}$, *i.e.*,

$$\overset{o}{N}_{\tau+b,c,n}\leq\hat{N}_{c,n},\ 0\leq b\leq B_n \tag{5.11}$$

Then, the total energy consumption of CDCs by calculating the sum of the energy consumed by all servers of all applications and the energy consumed by facilities, *e.g.*, lighting and cooling, is obtained. Let $\hat{\Phi}^c_n$ denote the peak power of a server of application n in CDC c. Let $\check{\Phi}^c_n$ denote the idle power of a server of application n in CDC c. Let α_c denote the power usage effectiveness (PUE) of CDC c. PUE is an important metric to evaluate the energy efficiency of data centers [213]. It is calculated as the ratio of energy consumed by a CDC to the total energy of its servers. Currently, the value of PUE is typically 1.2–2.0 for most large-scale existing commercial data centers.

In addition, let $u^{c,n}_{\tau+b}$ denote the average CPU utilization of servers of application n in CDC c in time slot $\tau+b$. Therefore, following the work in [24], the total power consumption of CDCs in time slot $\tau+b$ is obtained as:

$$P_{\tau+b}=\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}(\overset{o}{N}_{\tau+b,c,n}(\check{\Phi}^c_n+(\alpha_c-1)\hat{\Phi}^c_n+(\hat{\Phi}^c_n-\check{\Phi}^c_n)u^{c,n}_{\tau+b})) \tag{5.12}$$

The number of tasks executed by each switched-on server of application n in time slot $\tau+b$ is obtained as:

$$\frac{L\left(1-\delta_{\tau+b}^{c,n}\right)\overset{+}{\lambda}_{\tau+b,c,n}}{\overset{o}{N}_{\tau+b,c,n}} \tag{5.13}$$

Besides, the busy period of each switched-on server of application n in CDC c in time slot $\tau+b$ is $\frac{L\left(1-\delta_{\tau+b}^{c,n}\right)\overset{+}{\lambda}_{\tau+b,c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}\overset{o}{N}_{\tau+b,c,n}}$ minutes. Then, $u_{\tau+b}^{c,n}$ is obtained as:

$$u_{\tau+b}^{n}=\frac{L\left(1-\delta_{\tau+b}^{c,n}\right)\overset{+}{\lambda}_{\tau+b,c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}\overset{o}{N}_{\tau+b,c,n}} \tag{5.14}$$

$E_{\tau+b}^{c}$ denotes the total energy consumed by the execution of tasks of all applications in CDC c in time slot $\tau+b$. Based on equations (5.10), (5.12), and (5.14) $E_{\tau+b}^{c}$ $(0\leq b\leq\Upsilon)$ is obtained as:

$$E_{\tau+b}^{c}=\sum_{n=1}^{\mathcal{N}^{A}}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau+b}^{c,n}+\Delta_{c,n}^{4}\overset{+}{\lambda}_{\tau+b,c,n}(1-\delta_{\tau+b}^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right) \tag{5.15}$$

where

$$\Delta_{c,n}^{5}\triangleq\breve{\Phi}_{n}^{c}+(\alpha_{c}-1)\hat{\Phi}_{n}^{c}$$

$$\Delta_{c,n}^{4}\triangleq\hat{\Phi}_{n}^{c}-\breve{\Phi}_{n}^{c}$$

Let $\hat{\mathbb{E}}_{c}$ denote the maximum amount of energy in CDC c in each time slot. Therefore, the total energy consumed by CDC c in time slot $\tau+b$ $(0\leq b\leq\Upsilon)$ cannot exceed $\hat{\mathbb{E}}_{c}$, *i.e.*,

$$\sum_{n=1}^{\mathcal{N}^{A}}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau+b}^{c,n}+\Delta_{c,n}^{4}\overset{+}{\lambda}_{\tau+b,c,n}(1-\delta_{\tau+b}^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right)\leq\hat{\mathbb{E}}_{c} \tag{5.16}$$

Typically, different types of green energy facilities, *e.g.*, wind turbines and solar panels, are installed in CDCs. This chapter focuses on two types of renewable energy

including solar and wind energy. The use of green energy can decrease the power grid energy consumption of CDCs, and reduce its detrimental effect to the global environment. Similar to the work in [76], the length of each time slot is small enough and it is reasonable to assume that the wind and solar energy stay the same within each time slot. Let $\overset{\circ}{E}_{\tau+b,c}$ denote the amount of solar energy produced by solar panels in CDC c in time slot $\tau+b$. Let $\widetilde{E}_{\tau+b,c}$ denote the amount of wind energy produced by wind turbines in CDC c in time slot $\tau+b$. Similar to Chapter 3, $\overset{\circ}{E}_{\tau+b,c}$ and $\widetilde{E}_{\tau+b,c}$ are calculated with equations (3.5) and (3.6) in the green energy model introduced in Section 3.2, respectively.

5.2.3 Constrained Optimization Problem

Let f_{22} denote the cost of power grid energy consumed by the execution of tasks of all applications in CDCs from time slots τ to $\tau+\Gamma$. Then, f_{22} is calculated as:

$$f_{22}=\underset{\lambda^{c,n}_{\tau+b},\mu^{c,n}_{\tau+b}}{\mathbf{Min}} \left\{ \sum_{b=0}^{\Upsilon} \left(\sum_{c=1}^{\mathcal{N}^C} \left(p^c_{\tau+b} \left(\max \left(E^c_{\tau+b} - \overset{\circ}{E}_{\tau+b,c} - \widetilde{E}_{\tau+b,c}, 0 \right) \right) \right) \right) \right\} \quad (5.17)$$

$p^c_{\tau+b}$ $(0\leq b\leq\Upsilon)$ is the price of power grid in CDC c in time slot $\tau+b$. $E^c_{\tau+b}$ $(0\leq b\leq\Upsilon)$ denotes the total amount of energy consumed by the execution of tasks of all applications in CDC c in time slot $\tau+b$. It is assumed that the prices of power grid differ from a time slot to another but keep the same in each time slot. The amount of grid energy consumed by CDCs in time slot $\tau+b$ is obtained as $\max(E^c_{\tau+b} - \overset{\circ}{E}_{\tau+b,c} - \widetilde{E}_{\tau+b,c}, 0)$. The objective in the nonlinear constrained optimization model is to minimize f_{22}. Thus, the energy cost minimization problem is formulated as:

$$\underset{\lambda^{c,n}_{\tau+b},\mu^{c,n}_{\tau+b}}{\mathbf{Min}} \{f_{22}\}$$

subject to

$$\overset{o}{N}_{\tau+b,c,n}\leq\hat{N}_{c,n} \quad (5.18)$$

$$\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_{c,n}^5\mu_{\tau+b}^{c,n}+\Delta_{c,n}^4\overset{+}{\lambda}_{\tau+b,c,n}(1-\delta_{\tau+b}^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right)\leq\hat{\mathbb{E}}_c \tag{5.19}$$

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}+\sum_{\ell=\tau-B_n}^{\tau-B_n+b}\left(\lambda_\ell^{c,n}L\right)\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,c,n}+\sum_{\ell=\tau}^{\tau+b}\left(\overset{+}{\lambda}_{\ell,c,n}\left(1-\delta_\ell^{c,n}\right)L\right) \tag{5.20}$$

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,c,n}+\sum_{\ell=\tau-B_n}^{\tau}\left(\lambda_\ell^{c,n}L\right)=\overset{\diamond}{\mathbb{N}}_{\tau-1,c,n}+\sum_{\ell=\tau}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,c,n}(1-\delta_\ell^{c,n})L\right) \tag{5.21}$$

$$\overset{+}{\lambda}_{\tau+b,c,n}<\mu_{\tau+b}^{c,n} \tag{5.22}$$

$$\lambda_{\tau+b}^n=\sum_{c=1}^{\mathcal{N}^C}\lambda_{\tau+b}^{c,n} \tag{5.23}$$

$$\mu_{\tau+b}^{c,n}\geq 0,\ \lambda_{\tau+b}^{c,n}\geq 0,\ 0\leq b\leq B_n \tag{5.24}$$

$$\mu_{\tau+b}^{c,n}=0,\ \lambda_{\tau+b}^{c,n}=0,\ B_n<b\leq\Upsilon \tag{5.25}$$

Constraints (5.24) and (5.25) give valid ranges of decision variables including $\lambda_{\tau+b}^{c,n}$ and $\mu_{\tau+b}^{c,n}$. Let $\phi_{\tau+b,4c}$ and $\psi_{\tau+b,3c}$ denote the wind speed and the solar irradiance of CDC c in time slot $\tau+b$. In addition, similar to the work in [76], it is assumed that time-related parameters, *e.g.*, $p_{\tau+b}^c$, $\phi_{\tau+b,4c}$ and $\psi_{\tau+b,3c}$ can be well predicted by using typical prediction algorithms, *e.g.*, deep neural networks, at the beginning of each time slot $\tau+b$. Then, the algorithm that solves the energy cost minimization problem is presented in Section 5.4. The final decision variables transformed by the optimal solution to the energy cost minimization problem determine the optimal task scheduling among CDCs in each time slot. In this way, the energy cost of the CDC provider is minimized while delay bound constraints of tasks of all applications are strictly met.

5.3 Spatio-Temporal Task Scheduling

It is worth noting that the objective function in the energy cost minimization problem is nonlinear in terms of $\lambda_{\tau+b}^{c,n}$ and $\mu_{\tau+b}^{c,n}$. Therefore, it is a constrained nonlinear

programming problem. There are many complex constraints and this section uses a penalty function method [214] to transform each constraint into its corresponding penalty. In this way, the energy cost minimization problem is transformed into an unconstrained optimization one that is much easier to solve. Let $\boldsymbol{x}$ denote the vector of decision variables including $\lambda_{\tau+b}^{c,n}$ and $\mu_{\tau+b}^{c,n}$. Then,

$$\underset{\boldsymbol{x}}{\mathbf{Min}} \left\{ \widetilde{f_{22}} = \overset{\infty}{\mathcal{N}} \mho + f_{22} \right\} \tag{5.26}$$

In equation (5.26), $\widetilde{f_{22}}$ denotes the augmented objective function and $\overset{\infty}{\mathcal{N}}$ is a large positive integer. Let $\mho$ denote the penalty of all constraints if they are violated and it is obtained as:

$$\mho = \sum_{l=1}^{\mathcal{N}^{\neq}} (\max\{0, -g_l(\boldsymbol{x})\})^{\overset{0}{\gamma}_1} + \sum_{m=1}^{\mathbb{N}^{=}} \left| h_m(\boldsymbol{x}) \right|^{\overset{0}{\gamma}_2} \tag{5.27}$$

In equation (5.27), $\overset{0}{\gamma}_1$ and $\overset{0}{\gamma}_2$ are two positive constants. Each equality constraint $m(1 \leq m \leq \mathbb{N})$ is transformed into $h_m(\boldsymbol{x})=0$. If it is met, its penalty is 0; otherwise, its penalty is $\left| h_m(\boldsymbol{x}) \right|^{\overset{0}{\gamma}_2}$. Similarly, each inequality constraint $l(1 \leq l \leq \mathcal{N}^{\neq})$ is transformed into $g_l(\boldsymbol{x}) \geq 0$. If it is met, its penalty is 0; otherwise, its penalty is $(\max\{0, -g_l(\boldsymbol{x})\})^{\overset{0}{\gamma}_1}$. For example, constraint (6.25) is first transformed into $\hat{N}_{c,n} - \overset{o}{N}_{\tau+b,c,n} \geq 0$ $(0 \leq b \leq \Upsilon)$, and its penalty is $(\max\{0, -(\hat{N}_{c,n} - \overset{o}{N}_{\tau+b,c,n})\})^{\overset{0}{\gamma}_1}$. Then, the unconstrained optimization problem is obtained and it is solved by using the proposed GSPSO presented in Section 5.4.

Algorithm 2 shows the pseudo codes of STTS. Line 1 initializes $\lambda_{\tau}^{c,n} (\Upsilon - B_n \leq \tau \leq \Upsilon - 1)$, $\overset{+}{\mathbb{N}}_{\Upsilon - B_n - 1, c, n}$ and $\overset{\diamond}{\mathbb{N}}_{\Upsilon - 1, c, n}$ with 0. Line 2 initializes $\lambda_{\tau}^{\dagger, c, n}$ and $\overset{+}{\lambda}_{\tau, c, n}$ $(\Upsilon \leq \tau \leq \mathcal{N}^*)$ with $\lambda_{\tau}^{c,n}$. $\mathcal{N}^*$ denotes the total number of time slots. $\overset{+}{\lambda}_{\tau, c, n}$ is calculated according to equation (5.3). Line 6 obtains decision variables including $\lambda_{\tau+b}^{c,n}$ and $\mu_{\tau+b}^{c,n}$ by solving the unconstrained optimization problem with GSPSO. Lines 7–9 schedule $(\overset{+}{\lambda}_{\tau, c, n}(1 - \delta_{\tau}^{c,n})L)$ tasks to CDCs. $\lambda_{\ell}^{\dagger, c, n} (\tau - B_n \leq \ell \leq \tau)$ is updated in Lines 10–20. Lines 8–9 update $\overset{+}{\mathbb{N}}_{\tau - B_n, c, n}$ and $\overset{\diamond}{\mathbb{N}}_{\tau, c, n}$.

Algorithm 2 STTS (Spatio-Temporal Task Scheduling)

Initialize $\lambda_{\tau}^{c,n}(\Upsilon - B_n \leq \tau \leq \Upsilon - 1)$, $\overset{+}{\mathbb{N}}_{\Upsilon - B_n - 1,c,n}$ and $\overset{\diamond}{\mathbb{N}}_{\Upsilon - 1,c,n}$ with 0
Initialize $\lambda_{\tau}^{\dagger,c,n}$ and $\overset{+}{\lambda}_{\tau,c,n}$ $(\Upsilon \leq \tau \leq \mathcal{N}^*)$ with $\lambda_{\tau}^{c,n}$
$\tau \leftarrow \Upsilon$
while $\tau \leq \mathcal{N}^*$ **do**
  Calculate $\overset{+}{\lambda}_{\tau,c,n}$ based on equation (5.3).
  Solve the unconstrained optimization problem to obtain $\lambda_{\tau}^{c,n}, \mu_{\tau}^{c,n}$ *via* GSPSO
  Schedule $(\overset{+}{\lambda}_{\tau,c,n}(1 - \delta_{\tau}^{c,n})L)$ tasks to CDCs
  $\overset{+}{\mathbb{N}}_{\tau - B_n,c,n} \leftarrow \overset{+}{\mathbb{N}}_{\tau - B_n - 1,c,n} + \lambda_{\tau - B_n}^{c,n} L$
  $\overset{\diamond}{\mathbb{N}}_{\tau,c,n} \leftarrow \overset{\diamond}{\mathbb{N}}_{\tau - 1,c,n} + (\overset{+}{\lambda}_{\tau,c,n}(1 - \delta_{\tau}^{c,n})L)$
  $temp = (\overset{+}{\lambda}_{\tau,c,n}(1 - \delta_{\tau}^{c,n})L)$
  for $\ell \leftarrow \tau - B_n$ to τ **do**
    if $\lambda_{\ell}^{\dagger,c,n} L <= temp$ **then**
      $temp \leftarrow temp - \lambda_{\ell}^{\dagger,c,n} L$
      $\lambda_{\ell}^{\dagger,c,n} \leftarrow 0$
    else
      $\lambda_{\ell}^{\dagger,c,n} \leftarrow \lambda_{\ell}^{\dagger,c,n} - \frac{temp}{L}$
      $temp \leftarrow 0$
      break
    end if
  end for
  $\tau \leftarrow \tau + 1$
end while

5.4 Genetic Simulated-annealing-based Particle Swarm Optimization

Currently, there are several existing exact and deterministic algorithms, *e.g.*, fuzzy adaptive dynamic programming [215] and conjugate gradient descent [216], to solve the formulated unconstrained optimization problem. Nevertheless, they usually depend on specific structures of objective functions, *e.g.*, those having the first-order or second-order derivatives. In addition, the search space increases exponentially with the problem size, and therefore, the quality of their final solutions is usually unsatisfying when they are adopted to solve these problems.

Many researchers adopt meta-heuristic optimization algorithms [217] due to their advantages including wide applicability, robustness, easy implementation, good exploitation and exploration ability, and fast speed. They can prevent disadvantages

of these deterministic algorithms and have been commonly applied to solve many complicated problems. On the contrary, each meta-heuristic algorithm has its advantages and disadvantages [218]–[220] . For example, particle swarm optimization (PSO) is commonly used to solve optimization problems because of its quick convergence [219]. Nevertheless, it usually traps into locally optimal solutions when it is applied and adopted to solve problems with high-dimension solution spaces [218]. Besides, simulated annealing (SA) is widely adopted to solve different problems. Its Metropolis acceptance rule enables moves that worsen the objective function value. Consequently, SA can escape from local optima and finally converge to global one by specifying a suitable temperature cooling rate. On the other hand, its convergence process is very slow [220]. In addition, the genetic operations of genetic algorithm (GA) can increase the diversity of particles in PSO, and improve both efficiency and accuracy of global search. This chapter proposes a hybrid meta-heuristic algorithm named GSPSO that incorporates genetic operations of GA and the Metropolis acceptance rule of SA into the PSO.

Particles in PSO update velocities and positions according to learning experiences of individuals and the current swarm. The size of the swarm is denoted by $|\mathbb{X}|$. θ_1^i and $\mathbf{x}_i$ denote the velocity and position of particle i (i=1, 2, ..., $|\mathbb{X}|$). Each position is designed as a $\mathbb{N}^D$-dimension vector. The first $\mathcal{N}^A{*}\mathcal{N}^C{*}(\Upsilon{+}1)$ elements of each vector store $\lambda_{\tau+b}^{c,n}$ ($0{\leq}b{\leq}\Upsilon$). The next $\mathcal{N}^A{*}\mathcal{N}^C{*}(\Upsilon{+}1)$ elements of each vector store $\mu_{\tau+b}^{c,n}$ ($0{\leq}b{\leq}\Upsilon$). The last element of each vector stores the value of $\widetilde{f_{22}}$. Consequently, $\mathbb{N}^D{=}2{*}\mathcal{N}^A{*}\mathcal{N}^C{*}(\Upsilon{+}1){+}1$. $\check{\mathbf{x}}_i$ denotes the locally optimal position of particle i. The globally optimal position of the current swarm is denoted by $\hat{\mathbf{x}}$. Specifically, θ_1^i and $\mathbf{x}_i$ are updated as follows.

$$\theta_1^i{=}\theta_5^P{\cdot}\theta_1^i + \check{\theta}_1^P w_9\left(\check{\mathbf{x}}_i - \mathbf{x}_i\right) + \hat{\theta}_1^P w_{10}\left(\hat{\mathbf{x}} - \mathbf{x}_i\right) \tag{5.28}$$

$$\mathbf{x}_i{=}\mathbf{x}_i{+}\theta_1^i \tag{5.29}$$

w_9 and w_{10} denote two random numbers that are uniformly produced in the range of $(0, 1)$. $\check{\theta}_1^P$ and $\hat{\theta}_1^P$ denote coefficients of individual and social acceleration reflecting the influence of $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$, respectively. However, PSO is easy to trap into locally optimal solutions and it leads to premature convergence. Besides, the optimization process oscillates when $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ vary greatly. Genetic operations of GA can breed superior particles and improve their global search ability [58].

Therefore, let $\acute{\mathbf{x}}_i^P$ denote the position of a superior particle corresponding to particle i and it guides the optimization process of particles in PSO. $\acute{\mathbf{x}}_i^P$ is the combination of $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$, *i.e.*,

$$\acute{\mathbf{x}}_i^P = \frac{\check{\theta}_1^P w_9 \check{\mathbf{x}}_i + \hat{\theta}_1^P w_{10} \hat{\mathbf{x}}}{\check{\theta}_1^P w_9 + \hat{\theta}_1^P w_{10}} \tag{5.30}$$

Let ω denote the inertia weight. Then, θ_1^i and $\mathbf{x}_i$ of particle i are updated as:

$$\theta_1^i = \omega \cdot \theta_1^i + \theta_2^P * w_{11} \left(\acute{\mathbf{x}}_i^P - \mathbf{x}_i \right) \tag{5.31}$$

$$\mathbf{x}_i = \mathbf{x}_i + \theta_1^i \tag{5.32}$$

In equation (5.31), θ_2^P denotes the superior acceleration coefficient, and w_{11} denotes a vector, each element of which is a random number uniformly produced in $(0, 1)$. Besides, $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ are coded with binary encoding where their velocities and positions are coded as a string of binary bits [221]. Let θ_7 and θ_8 denote the crossover rate and the mutation rate, respectively. The single point crossover operation is implemented on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ to obtain the offspring $\breve{\mathbf{x}}_i^P$ for particle i with a probability w_{23}. Then, the mutation operation is performed on each bit of $\breve{\mathbf{x}}_i^P$ with a small probability w_{24}. In this way, the early trapping into locally optimal solutions can be avoided. Then, the selection operation is performed to determine whether $\breve{\mathbf{x}}_i^P$ or $\acute{\mathbf{x}}_i^P$ is selected to guide the optimization process of particle i.

Specifically, if $\widetilde{f_{22}}(\breve{\mathbf{x}}_i^P) \leq \widetilde{f_{22}}(\acute{\mathbf{x}}_i^P)$, $\breve{\mathbf{x}}_i^P$ is selected as a superior particle corresponding to particle i; otherwise, $\acute{\mathbf{x}}_i^P$ is selected as the superior particle. Thus, a superior particle is obtained for each particle. Let $\mathbf{x}_i^g$ and $\mathbf{x}_i^{g+1}$ denote particle i's positions in iterations g and g+1. Then, $\mathbf{x}_i^{g+1}$ is updated as:

$$\theta_1^i = \omega \cdot \theta_1^i + \theta_2^P \cdot w_{11} \cdot \left(\acute{\mathbf{x}}_i^P - \mathbf{x}_i^g \right) \tag{5.33}$$

$$\mathbf{x}_i^{g+1} = \mathbf{x}_i^g + \theta_1^i \tag{5.34}$$

If $\widetilde{f_{22}}(x_i^{g+1}) \leq \widetilde{f_{22}}(x_i^g)$, $\mathbf{x}_i^{g+1}$ is accepted; otherwise, it is conditionally accepted if

$$exp\left(\frac{\widetilde{f_{22}}\left(\mathbf{x}_i^g\right) - \widetilde{f_{22}}\left(\mathbf{x}_i^{g+1}\right)}{\theta_2^g} \right) > w_{12} \tag{5.35}$$

where w_{12} denotes a random number uniformly distributed in (0,1) and θ_2^g denotes the temperature in iteration g.

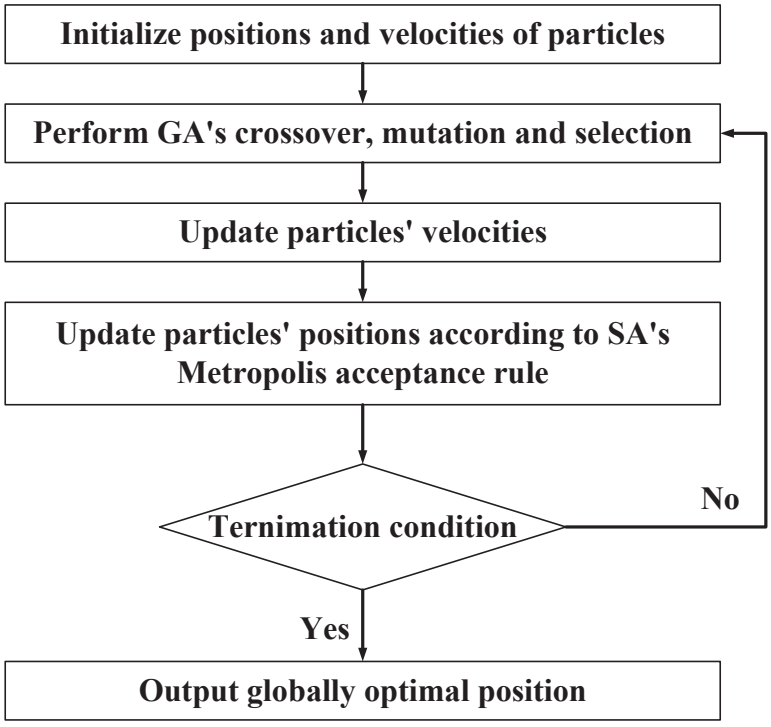


Figure 5.2 Flowchart of GSPSO.

A hybrid optimization algorithm called Genetic Simulated-annealing-based Particle Swarm Optimization (GSPSO) is proposed. It integrates advantages of SA, PSO and GA. It adopts genetic operations of GA on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ to generate superior particles, and updates particles' positions with the Metropolis acceptance rule of SA. Thus, GSPSO improves the search quality of PSO.

Algorithm 3 describes the pseudo codes of GSPSO and its flowchart is shown in Figure 5.2. Velocities and positions of particles in PSO are randomly initialized in Line 1. Velocities of particles are limited to $[-\theta_3^P, \theta_3^P]$. The fitness value of each particle is calculated according to $\widetilde{f_{22}}$ in the problem (5.26) in Line 2. Let $\hat{\mathbf{x}}$ denote the globally optimal position of the swarm. Let $\check{\mathbf{x}}_i$ denote a locally optimal position of particle i. Line 3 updates $\hat{\mathbf{x}}$ and $\check{\mathbf{x}}_i$. The parameters related to GA, PSO and SA are initialized in Line 4. The total number of iterations is denoted by $\hat{g}$. Let θ_4^P denote the percentage of particles with identical fitness values. Let $\hat{\theta}_4^P$ denote the specified maximum percentage. Line 6 shows that the while loop terminates if $\hat{g}$ is exceeded, or $\theta_4^P > \hat{\theta}_4^P$. Line 7 conducts GA's classical crossover on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$.

The obtained offspring $\check{\mathbf{x}}_i^P$ undergoes GA's classical mutation with a bounded probability in Line 8. Then, the classical selection of GA is conducted to determine whether $\check{\mathbf{x}}_i^P$ or $\acute{\mathbf{x}}_i^P$ is chosen in Line 9. Line 10 updates velocities of particles and Line 11 updates their positions according to the Metropolis acceptance rule of SA. Lines 12–13 calculate fitness values of particles, and update $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ accordingly. In addition, θ_2^0 denotes the initial temperature and θ_3 denotes the temperature cooling rate. Line 14 decreases temperature by θ_3. Let $\check{\theta}_5^P$ and $\hat{\theta}_5^P$ denote lower and upper bounds of inertia weight θ_5^P, respectively. In addition, θ_5^P is decreased linearly from $\hat{\theta}_5^P$ to $\check{\theta}_5^P$ in Line 15. Then, Line 16 updates θ_4^P. Line 19 outputs $\hat{\mathbf{x}}$ and transforms it into decision variables including $\lambda_{\tau+b}^{c,n}$ and $\mu_{\tau+b}^{c,n}$ ($0 \leq b \leq \Upsilon$, $1 \leq n \leq \mathcal{N}^A$, $1 \leq c \leq \mathcal{N}^C$). To discuss the overhead of STTS, the complexity analysis of Algorithms 2 and 3 is given as follows.

Algorithm 3 GSPSO (Genetic Simulated-annealing-based PSO)

1: Initialize positions and velocities of particles
2: Calculate the fitness value $\widetilde{f_{22}}$ of each particle
3: Update $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$
4: Initialize parameters related to GA, PSO and SA
5: $g \leftarrow 1$
6: **while** $\theta_4^P \leq \hat{\theta}_4^P$ and $g \leq \hat{g}$ **do**
7: Perform GA's classical crossover on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$
8: Perform GA's classical mutation on offspring $\breve{\mathbf{x}}_i^P$
9: Perform GA's classical selection
10: Update velocity of each particle
11: Update position of each particle according to Metropolis acceptance rule of SA
12: Calculate fitness value $\widetilde{f_{22}}$ of each particle
13: Update $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$
14: Decrease temperature by θ_3
15: Decrease θ_5^P linearly from $\hat{\theta}_5^P$ to $\check{\theta}_5^P$
16: Update θ_4^P
17: $g \leftarrow g + 1$
18: **end while**
19: Output $\hat{\mathbf{x}}$

In Algorithm 3, most of the execution overhead is caused by the while loop. In the worst case, the while loop stops after $\hat{g}$ iterations. As is shown in Lines 7–17, the time complexity of each iteration is $\mathcal{O}(|\mathbb{X}|\mathbb{N}^D)$. In addition, it is worth noting that $\mathbb{N}^D = 2 * \mathcal{N}^A * \mathcal{N}^C * (\Upsilon + 1) + 1$, and the time complexity of each iteration is $\mathcal{O}(|\mathbb{X}|\mathcal{N}^A\mathcal{N}^C\Upsilon)$. Thus, the time complexity of Algorithm 3 is $\mathcal{O}(\hat{g}|\mathbb{X}|\mathcal{N}^A\mathcal{N}^C\Upsilon)$. Most of the execution overhead in Algorithm 2 is caused by its while loop where $\mathcal{N}^* - \Upsilon + 1$ iterations are executed and the overhead of each iteration is determined by GSPSO in Line 6. Besides, $\mathcal{N}^*$ is typically much greater than Υ. Consequently, the time complexity of Algorithm 2 is $\mathcal{O}(\mathcal{N}^*\hat{g}|\mathbb{X}|\mathcal{N}^A\mathcal{N}^C\Upsilon)$.

Then, the assumptions of STTS are discussed. Similar to the work in [21, 222], this chapter considers a simpler situation where it is assumed that each task can be executed independently in each CDC and its execution does not rely on other tasks. In a real-life and more complex situation, *e.g.*, Hadoop MapReduce, each task belongs to a single job and a job consists of multiple tasks. The execution of each task is associated with other tasks. Thus, the execution of each job depends

on correlations among its tasks, and each job can be treated as a typical workflow where a job is partitioned into multiple tasks. There are many existing workflow scheduling algorithms that are proposed recently to tackle the scheduling of jobs in clouds [102, 223]. However, it has to be left out because this is beyond the scope of this chapter.

This chapter chooses delay bound constraints and throughput of tasks as the main quality of service (QoS) parameters that are widely utilized in the existing studies [21, 76, 110, 222, 224]. There are other important QoS parameters, *e.g.*, resource utilization, availability, security, reliability and data integrity [109]. Nevertheless, this chapter focuses on how to minimize energy cost by determining a task scheduling strategy, and future research would further consider other QoS parameters. What's more, similar to the work in [21, 76, 110, 222, 224], it is assumed that tasks of the same application have the same priority and they can only be executed by their corresponding cluster of servers. Thus, though tasks of different applications have diverse priorities, they are executed on separate clusters of servers. Similar to the work in [21, 76, 110, 222, 224], it is also assumed that all tasks of each application are scheduled to CDCs in a non-preemptive manner.

5.5 Performance Evaluation

This section evaluates STTS with real-life data, *i.e.*, arriving tasks in Google cluster[1], prices of grid, wind and solar energy. STTS is coded and implemented in MATLAB 2017 and it is executed in a computer with an Intel Xeon E-2124G processor at 4.50 GHz and a 32-GB DDR4 memory. This section adopts real-life tasks of three heterogeneous delay-tolerant applications executed in Google cluster for a whole day on May 10, 2011.

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).

5.5.1 Parameter Setting

This chapter calculates task arriving rates of three applications (types 1, 2 and 3) that are sampled every 5 minutes, *i.e.*, $\mathcal{N}^A$=3 and L=5 minutes. Following the work in [21], this chapter sets delay bound constraints of three applications to 3, 4 and 5 time slots, *i.e.*, B_1=3, B_2=4 and B_3=5. This chapter considers three CDCs, *i.e.*, $\mathcal{N}^C$=3, and uses real-life power grid prices on the same day in capital region of New York, U.S.[2] to simulate power grid prices of three CDCs.

Table 5.1 Parameter Setting of Three Energy Sources

	Power grid		Wind energy			Solar energy	
	α_c	$\hat{\mathbb{E}}_c$ (WH)	ϕ_{1c}	ϕ_{3c} (m^2)	ϕ_{2c} (kg/m^3)	ψ_{1c}	ψ_{2c} (m^2)
c=1	1.2	7.2×10^8	0.45	37500	1.8375	0.3	22500
c=2	1.4	4.5×10^8	0.375	31250	1.5313	0.25	18750
c=3	1.6	3.6×10^8	0.3	25000	1.225	0.2	15000

According to the work in [31, 76], the parameter setting of three energy sources is shown in Table 5.1. According to the work in [24], parameters in the power consumption model are set in Tables 5.2 and 5.3. Besides, this chapter adopts real-life data of solar irradiance[3] and wind speed[4] on the same day. It is worth noting that most of typical meta-heuristic algorithms are sensitive to their parameter setting. Thus, based on existing studies [24, 25, 76], a series of experiments are conducted to investigate the optimal parameter setting for GSPSO.

The minimization of energy cost in each iteration is considered for GSPSO. The energy cost of the CDC provider continues to decrease as iterations proceed, and the optimal number of iterations is 200. Further increasing the number of iterations from 200 cannot continue to reduce the energy cost, but increases the execution time of GSPSO. Similarly, the optimal balance between the diversity of solutions and the energy cost is achieved when the size of the swarm is 100.

[2]http://www.nyiso.com/public/index.jsp (accessed on May 10, 2019).
[3]http://www.nrel.gov/midc/srrl_bms/ (accessed on May 10, 2019).
[4]http://www.nrel.gov/midc/nwtc_m2/ (accessed on May 10, 2019).

Table 5.2 Parameter Setting in Power Consumption Model-Part 1

	$\check{\Phi}_n^c$ (W)			$\hat{\Phi}_n^c$ (W)			$\hat{N}_{c,n}$		
	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
c=1	200	100	50	400	200	100	12000	15000	18000
c=2	250	125	62.5	500	250	125	10000	12500	15000
c=3	300	150	75	600	300	150	8000	10000	12000

Table 5.3 Parameter Setting in Power Consumption Model-Part 2

	$\hat{Q}_n^c$			$\dot{\mathbb{N}}_{c,n}$ (tasks/second)		
	n=1	n=2	n=3	n=1	n=2	n=3
c=1	18.75	37.5	75	0.075	0.15	0.3
c=2	15.625	31.25	62.5	0.0625	0.125	0.25
c=3	12.5	25	50	0.05	0.1	0.2

In addition, the energy cost is high when the crossover rate is very low, and therefore, GSPSO tends to converge to a locally optimal solution. The energy cost is continuously decreasing by increasing the crossover rate, and the optimal crossover rate of 0.6 is selected for GSPSO. The energy cost increases when the crossover rate is further increased from 0.6 due to the early stagnation of solutions leading to sub-optimal energy cost. Besides, the close-to-optimal energy cost is achieved by GSPSO when the mutation rate is set to 0.1%. There is some divergence when the mutation rate is less than 0.1%, and it leads to a sub-optimal solution. Higher diversity in the solution is achieved when the mutation rate is larger than 0.1%, but it makes GSPSO difficult to converge to the final close-to-optimal solution.

In the above process, the optimal values of crossover and mutation rates are obtained by evaluating the performance of GSPSO while all other parameters, *e.g.*, the size of the swarm, the number of iterations and weight θ_5^P, are kept as constants. Then, the size of the swarm and the number of iterations are set similarly. Other parameters of GSPSO are specified in a similar way. The final values of the parameters for GSPSO are set as: $\check{\theta}_5^P$=0.4, $\hat{\theta}_5^P$=0.95, $\hat{\theta}_4^P$=95%, $\check{\theta}_1^P$=$\hat{\theta}_1^P$=0.5, θ_3=0.975, θ_2^0=10^{30}, $\hat{g}$=200, $|\mathbb{X}|$=100, θ_7=0.6 and θ_8=0.1%. In addition, $\overset{\infty}{\mathcal{N}}$=$10^{20}$ and $\overset{0}{\gamma}_1$=$\overset{0}{\gamma}_2$=2.

5.5.2 Experimental Results

To demonstrate GSPSO's performance, this section compares it with three typical meta-heuristic algorithms including SA, PSO and bat algorithm (BA). The reasons of choosing them are explained as follows. SA can converge to globally optimal solutions in theory through suitable temperature cooling rate because it can escape from locally optimal solutions [220]. Consequently, the comparison between GSPSO and SA can demonstrate the precision of GSPSO's final solution. In addition, the convergence speeds of PSO and BA are fast [218]. Thus, the comparison among GSPSO, PSO and BA can prove GSPSO's convergence speed.

Figure 5.3 shows the comparison of execution time among GSPSO, BA, PSO and SA. It is shown that the average execution time of SA is 7.48 (seconds), which is 4.18, 10.39 and 22 times larger than that of GSPSO, 1.79 (seconds), that of BA, 0.72 (seconds) and that of PSO, 0.34 (seconds), respectively. In addition, it is worth noting that the execution time of BA and PSO is much less than those of SA and GSPSO. This is because BA and PSO are easy to quickly trap into locally optimal solutions. Figure 5.4 shows the comparison of energy cost in each iteration of GSPSO, BA, PSO and SA in time slot 50. Here an iteration in GSPSO denotes Lines 7–17 in Algorithm 3. Similar to GSPSO, iterations of BA, PSO and SA have the same meaning. Besides, Figure 5.5 shows the corresponding penalty calculated according to (3.21) in each iteration in time slot 50. Figure 5.5 shows that PSO and BA converge after 225 and 250 iterations, respectively. The penalty of final solutions of PSO and BA is 4.22×10^6 and 3.98×10^6, respectively. It demonstrates that PSO and BA cannot find final solutions that meet all constraints in the energy cost minimization problem. Consequently, their solutions are not satisfying due to their quick convergence to locally optimal solutions.

Besides, SA converges to its final solution after 850 iterations, and the energy cost of its final solution is 19.04 and 11.43 times less than those of PSO and BA,

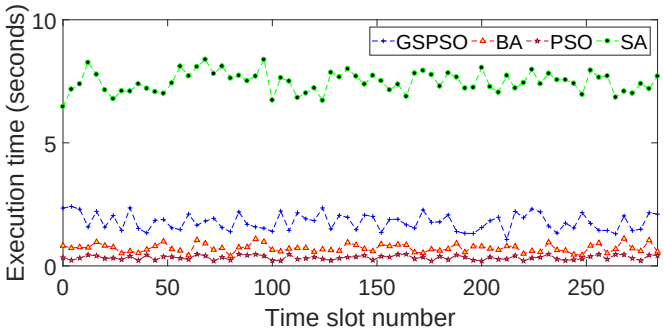


Figure 5.3 Comparison of execution time.

respectively, but still larger than that of GSPSO. GSPSO converges to its final solution after 300 iterations, and the energy cost of its final solution is \$5941.06. Therefore, GSPSO decreases the energy cost by \$694.05 in much fewer iterations than SA. In addition, it is shown in Figure 5.5 that the penalty of the final solution of GSPSO is 0. It demonstrates that GSPSO is able to obtain a satisfying solution meeting all constraints. Therefore, Figures 5.4–5.5 prove that GSPSO increases the search speed and precision, and brings less energy cost for the CDC provider by integrating SA's Metropolis acceptance rule and GA's genetic operations into PSO.

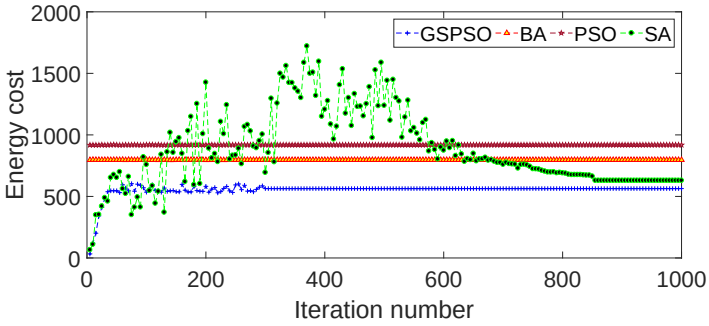


Figure 5.4 Energy cost of each iteration in time slot 50.

Figure 5.6 illustrates the cumulative scheduled tasks (CSTs) and the cumulative arriving tasks (CATs) for each application. It is illustrated that all arriving tasks of each application are scheduled and executed to CDCs within their delay bound constraints. For example, Figure 5.6(b) shows that the number of CATs of type 2 application in time slot 155 is equal to that of CSTs in time slot 159. This means that

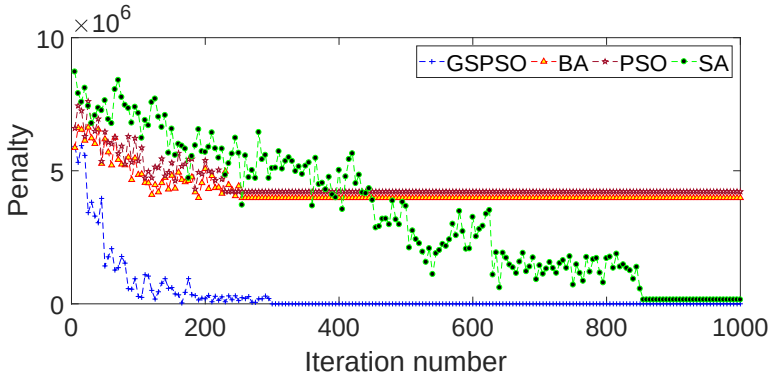


Figure 5.5 Penalty of each iteration in time slot 50.

tasks of type 2 application arriving in time slot 155 or before are all scheduled and executed to CDCs by time slot 159. Consequently, it demonstrates that the proposed STTS strictly meets delay bound constraints of tasks of each application.

5.5.3 Results of Comparison

To demonstrate STTS's performance, it is compared with four typical scheduling methods [31, 76, 99] in terms of energy cost and throughput of the CDC provider.

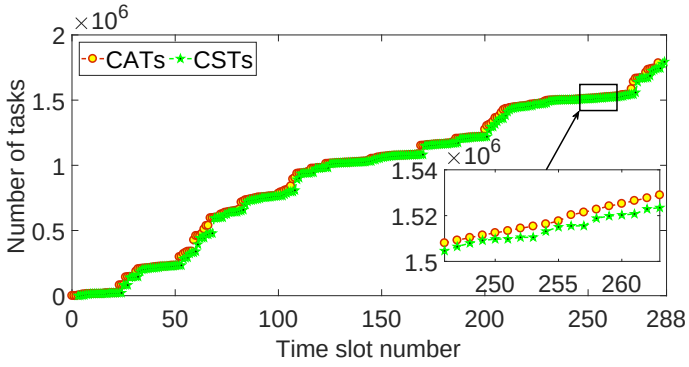
1) M1. Similar to a green energy-first scheduling approach in [31], schedules all tasks to CDCs in the time slot when the total amount of renewable energy is the maximum within tasks' delay bound constraints.

2) M2. Similar to a cheap-electricity-first scheduling approach in [76], schedules all tasks to CDCs in the time slot when the power grid price is the minimum within tasks' delay bound constraints.

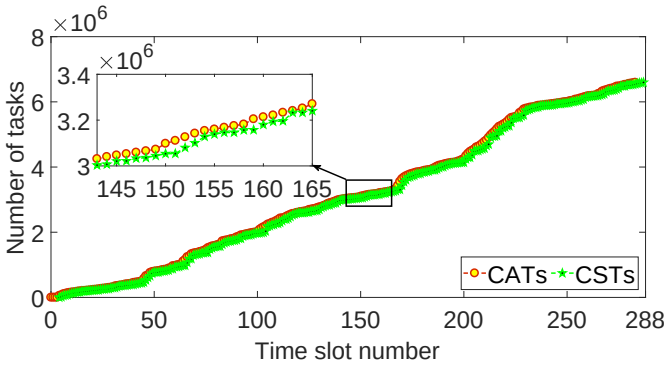
3) M3 [99]. It ignores the temporal differences in power grid prices, wind speed and solar irradiance. All tasks are directly scheduled to CDCs in their arriving time slot.

4) Spatio-temporal load balancing (STLB) [222]. It investigates both temporal and spatial variations of power grid prices to schedule tasks to CDCs. A task of application n arriving in current time slot τ is scheduled to one CDC during the next B_n+1 time slots beginning from τ.

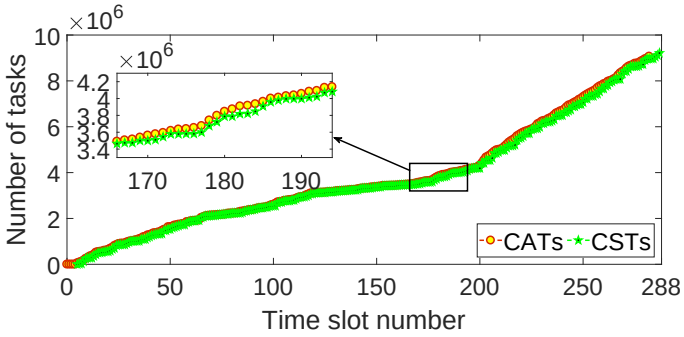
The parameters of M1–M3 and STLB are set in the same way as those of GSPSO are set in Section 5.5. Figures 5.7–5.9 compare STTS with M1–M3 and STLB in terms of the cumulative throughput obtained by calculating the sum of the number of tasks executed from τ to $\tau+B_n$. For example, in each time slot τ, the



(a) Type 1



(b) Type 2



(c) Type 3

Figure 5.6 CATs and CSTs of each application.

cumulative throughput of type 3 application is obtained by calculating the sum of the number of tasks executed from τ to τ+5. In addition, it is shown that the cumulative throughput of each application with STTS is higher than those with M1–M3 and STLB. For example, the cumulative throughput of type 1 application with STTS is higher than those with M1–M3 and STLB by 73.01%, 73.94%, 50.74% and 14.92% on average, respectively. The reason is that the maximum amount of energy in each CDC

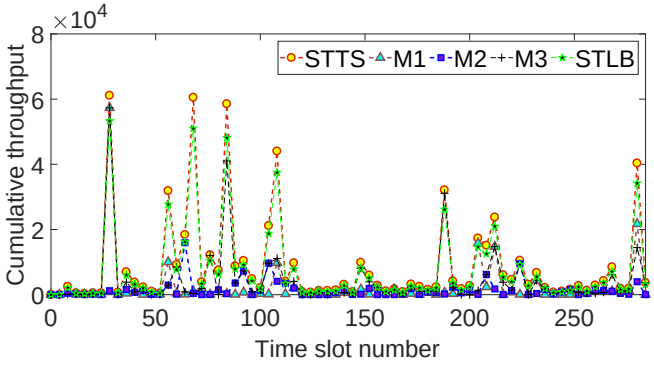


Figure 5.7 Cumulative throughput comparison of type 1 application.

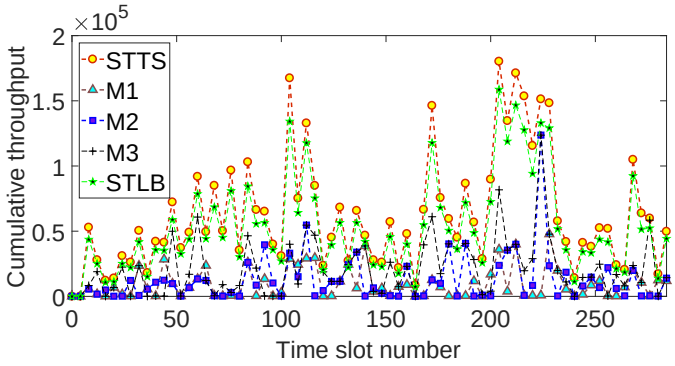


Figure 5.8 Cumulative throughput comparison of type 2 application.

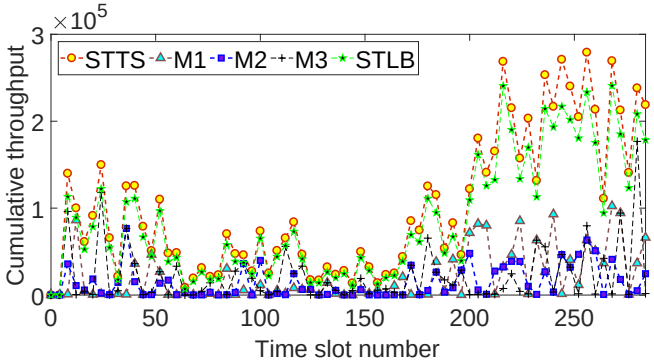


Figure 5.9 Cumulative throughput comparison of type 3 application.

in each time slot is constrained. Therefore, some tasks of each application have to be rejected and not executed in CDCs in M1–M3. STLB ignores spatial and temporal variations of solar and wind energy and fails to consider the differences of energy consumption of servers in different CDCs. Therefore, Figures 5.7–5.9 illustrate that STTS effectively increases the cumulative throughput of CDCs.

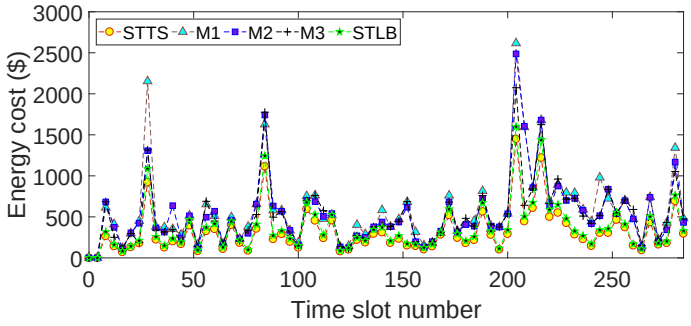


Figure 5.10 Energy cost of STTS, M1–M3 and STLB.

Figure 5.10 illustrates the energy cost of STTS, M1–M3 and STLB, respectively. Typically, the penalty cost is predefined in SLAs [20] for rejected tasks. It is signed between users and the CDC provider, and each rejected task causes the penalty for the latter. Let ϵ_τ^n denote the penalty cost brought by the rejection of each task of application n in time slot τ. To guarantee their performance, ϵ_τ^n is typically higher than the maximum energy cost caused by each task of application n from τ to $\tau+B_n$. $\triangle_\tau^n$ denotes the energy cost caused by each task of application n in time slot τ. Thus, $\epsilon_\tau^n=max_{b\in\{0,1,\ldots,B_n\}}(\triangle_{\tau+b}^n)$. Similar to Figure 3.12, energy cost in time slot τ is defined as the sum of energy cost of executed tasks, and the penalty caused by rejected tasks from τ to $\tau+\Gamma$. It is shown in Figure 5.10 that compared with M1–M3 and STLB, the energy cost of STTS is decreased by 43.27%, 38.54%, 36.96% and 12.64% on average, respectively. This is because STTS intelligently schedules more tasks to CDCs by jointly investigating the spatio-temporal variations in prices of power grid, and the total amount of wind and solar energy in CDCs during the allowed delay of tasks.

5.6 Summary

Types and number of delay-tolerant applications in cloud data centers (CDCs) increase significantly. The dramatic growth of arriving tasks increases the amount of energy consumed by execution of tasks in CDCs. Spatial and temporal variations in prices of power grid and availability of green energy in CDCs bring a challenge of how to cost-effectively schedule all arriving tasks of multiple applications among

CDCs. This chapter proposes a Spatio-Temporal Task Scheduling (STTS) method that exploits such spatial and temporal variations. STTS can smartly schedule all tasks of applications to CDCs while strictly meeting tasks' delay bound constraints. In each iteration of STTS, energy cost for a CDC provider is formulated and solved with a hybrid meta-heuristic algorithm named genetic simulated-annealing-based particle swarm optimization. Extensive trace-driven experimental results demonstrate that it achieves higher throughput and lower energy cost for the CDC provider in comparison with several recent scheduling approaches while meeting delay bound constraints of all tasks of heterogeneous applications.

CHAPTER 6

TEMPORAL TASK SCHEDULING OF MULTIPLE DELAY-CONSTRAINED APPLICATIONS IN HYBRID CLOUD DATA CENTER

This chapter presents the details of the proposed Temporal Task Scheduling (TTS) algorithm, and it is organized as follows. A hybrid cloud data center (CDC) architecture is presented in Section 6.1. Based on a hybrid CDC architecture, a task scheduling problem that aims to maximize the total profit of the hybrid CDC is formulated in Section 6.2. The proposed TTS algorithm is presented in Section 6.3. The experimental results and their analysis based on real-life data are shown in Section 6.4. Section 6.5 concludes this chapter.

6.1 Hybrid CDC Architecture

A data center provider owns a CDC and executes some tasks to public clouds when its resources are limited or the use of virtual machines (VMs) in public clouds is cheaper. The hybrid CDC architecture is presented in Figure 6.1. Users' tasks are sent to the hybrid CDC through multiple devices, *e.g.*, computers, smart phones, laptops and servers. Then, arriving tasks are classified and enqueued into FCFS queues according to their application types. Then, task arriving rates of all applications are set. Let $\mathcal{N}^A$ denote the number of applications. In Figure 6.1, μ_τ^n and $\overset{+}{\lambda}_{\tau,n}$ ($1 \leq n \leq \mathcal{N}^A$) denote the task service rate and the accumulated task arriving rate of application n in τ, respectively. This chapter focuses on *Task Scheduler* where the TTS algorithm periodically schedules all tasks to the hybrid CDC within their delay bound constraints.

At the start of each time slot, tasks of each application are dispatched by the *Task Scheduler* from their FCFS queues. *Task Scheduler* schedules tasks based on the optimal task scheduling strategy produced by TTS. Specifically, TTS first determines

the optimal task service rate of each server in the CDC. Then, the number of tasks scheduled to the CDC is obtained as $\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L$, and the number of tasks scheduled to public clouds is obtained as $\sum_{v=1}^{\mathcal{N}^C} z_\tau^{v,n} \overset{\#}{N}_{\tau,v,n}$. Here, L denotes the length of a time slot. δ_τ^n denotes the loss possibility of application n's tasks in τ. $\mathcal{N}^C$ denotes the number of public clouds. Besides, if tasks of application n are scheduled to public cloud v in time slot τ, $z_\tau^{v,n}$=1; otherwise, $z_\tau^{v,n}$=0. $\overset{\#}{N}_{\tau,v,n}$ denotes the number of tasks of application n scheduled to public cloud v ($1 \leq v \leq \mathcal{N}^C$) in time slot τ. At the start of each time slot, *Task Scheduler* schedules $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L + \sum_{v=1}^{\mathcal{N}^C} z_\tau^{v,n} \overset{\#}{N}_{\tau,v,n})$ tasks and removes them from their FCFS queues.

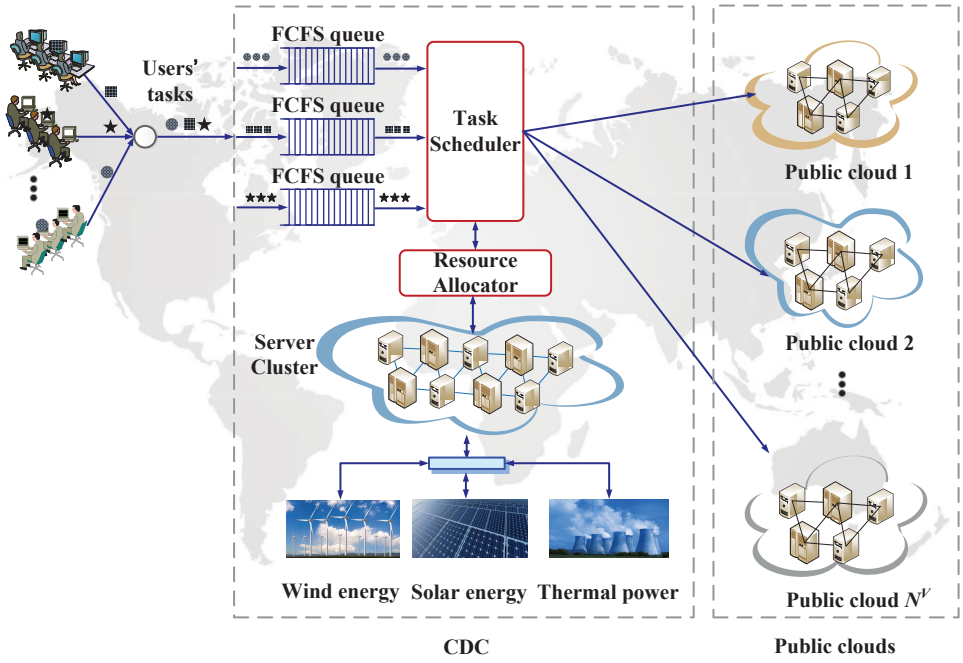


Figure 6.1 Hybrid CDC architecture.

The CDC obtains power grid energy from three types of sources: power grid, solar panels and wind turbines. The *Task Scheduler* periodically collects the information of power grid, solar panels and wind turbines. Section 6.2 formulates the task scheduling problem as a constrained Mixed Integer NonLinear Program (MINLP) [225]. Then, it is solved by a hybrid algorithm called GSPSO, which is described in Chapter 5, in each iteration of TTS to obtain the scheduling strategy. Based on it,

TTS is proposed to achieve profit maximization for the hybrid CDC while strictly meeting delay bound constraints of all tasks. Then, the task service rate for each application in each time slot is specified by *Task Scheduler*. Then, *Resource Allocator* configures each server in the CDC. It is assumed that servers for the same application are homogeneous and servers for different applications are heterogeneous with respect to hardware setting [18].

6.2 Problem Formulation

Based on the hybrid CDC architecture, this section formulates the task scheduling problem that aims to maximize the total profit of the hybrid CDC. This problem is formulated as a constrained MINLP and solved by GSPSO.

The modeling of delay bound constraints of each delay-constrained application is first described. They aim to guarantee that delay bound constraints of tasks of all applications are strictly met. δ_τ^n is important because it explicitly presents the mathematical relation between the number of tasks refused and task service rates of the CDC. Let B_n denote the delay bound constraint of tasks of application n. Based on it, the number of tasks executed in the CDC and the total revenue of the hybrid CDC brought by tasks of all applications executed in time slots τ and $\tau+b$ ($1 \leq b \leq B_n$) is determined. Let $\Upsilon = max_{n \in \{1,2,\cdots,\mathcal{N}^A\}} B_n$. The total energy consumption of the CDC in time slot τ or $\tau+b$ ($1 \leq b \leq \Upsilon$) is calculated. Then, the grid energy cost of the CDC and the execution cost of VMs in public clouds are obtained. In this way, the profit of the hybrid CDC is obtained. Thus, this equation provides an explicit and more accurate relation between task service rates and the profit of the hybrid CDC.

Similar to the work in [226], all servers for application n are modeled as an $M/M/1/\hat{\mathbb{Q}}_n/\infty$ system. The largest number of tasks that application n's servers can execute is denoted by $\hat{\mathbb{Q}}_n$. Similar to the work in [117], the hybrid CDC is modeled as a discrete-time system evolving with time slots. A growing number of high-performance

servers are deployed in CDCs. Thus, similar to the work in [25], it is reasonable to assume each task can finish its execution in one single time slot.

This means that before time slot τ passes, application n's tasks arriving in time slot $\tau - B_n$ or before are all executed in the hybrid CDC. Task arriving rate of application n in time slot τ is denoted by $\overset{+}{\lambda}_{\tau,n}$. Task service rates of servers corresponding to application n in time slots τ and $\tau+b$ ($1 \leq b \leq B_n$) are denoted by μ_τ^n and $\mu_{\tau+b}^n$, respectively. The task service rate denotes the rate at which tasks of application n are removed from their FCFS queue and executed in the hybrid CDC in time slot τ. Then,

$$\delta_\tau^n = \begin{cases} \dfrac{1-\frac{\overset{+}{\lambda}_{\tau,n}}{\mu_\tau^n}}{1-\left(\frac{\overset{+}{\lambda}_{\tau,n}}{\mu_\tau^n}\right)^{\hat{\mathbb{Q}}_n+1}} \left(\dfrac{\overset{+}{\lambda}_{\tau,n}}{\mu_\tau^n}\right)^{\hat{\mathbb{Q}}_n} & \mu_\tau^n > 0, \\ 1 & \mu_\tau^n = 0. \end{cases} \tag{6.1}$$

The number of tasks of application n accumulated before time slot $\tau+1$ is denoted by $\overset{+}{\mathbb{N}}_{\tau,n}$. Then,

$$\overset{+}{\mathbb{N}}_{\tau,n} = \sum_{\ell=1}^{\tau} \overset{+}{\lambda}_{\ell,n} L \tag{6.2}$$

The number of tasks of application n executed by time slot τ is denoted by $\overset{\diamond}{\mathbb{N}}_{\tau,n}$. The number of tasks of application n executed in the CDC in time slot ℓ ($1 \leq \ell \leq \tau$) is $\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L\right)$. The number of tasks of application n executed in public cloud v ($1 \leq v \leq \mathcal{N}^V$) in time slot ℓ ($1 \leq \ell \leq \tau$) is $z_\ell^{v,n} \overset{\#}{N}_{\ell,v,n}$. Then, the number of tasks of application n executed in all public clouds in time slot ℓ ($1 \leq \ell \leq \tau$) is $\sum_{v=1}^{\mathcal{N}^V} z_\ell^{v,n} \overset{\#}{N}_{\ell,v,n}$. The remaining arriving rate of application n's tasks in τ is denoted by $\lambda_\tau^{\dagger,n}$. Thus, $\overset{\diamond}{\mathbb{N}}_{\tau,n}$ is obtained as follows.

$$\overset{\diamond}{\mathbb{N}}_{\tau,n} = \sum_{\ell=1}^{\tau} \left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L + \sum_{v=1}^{\mathcal{N}^V} z_\ell^{v,n} \overset{\#}{N}_{\ell,v,n} \right) \tag{6.3}$$

Thus, all application n's tasks that arrive in $\tau-B_n$ or earlier are executed in the hybrid CDC before τ passes. Thus, $\lambda_\ell^{\dagger,n}=0$ ($\ell\leq\tau-B_n-1$). At the start of time slot τ, the total remaining arriving rate of application n's tasks from $\tau-B_n$ to $\tau-1$ is $\sum_{\ell=\tau-B_n}^{\tau-1}\lambda_\ell^{\dagger,n}$. Thus, $\overset{+}{\lambda}_{\tau,n}$ is the sum of λ_τ^n and $\sum_{\ell=\tau-B_n}^{\tau-1}\lambda_\ell^{\dagger,n}$. Then,

$$\overset{+}{\lambda}_{\tau,n}=\lambda_\tau^n+\sum_{\ell=\tau-B_n}^{\tau-1}\lambda_\ell^{\dagger,n} \tag{6.4}$$

At the start of τ, the number of application n's tasks arriving in $\tau-B_n$ or earlier is $\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\lambda_{\tau-B_n}^n L$. The number of application n's tasks executed by τ is the sum of $\overset{\diamond}{\mathbb{N}}_{\tau-1,n}$, and the number of application n's tasks executed in the hybrid CDC in τ, *i.e.*, $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n})$. All application n's tasks arriving in τ have to be executed in time slots τ to $\tau+B_n$. It means that all application n's tasks in $\tau-B_n$ or earlier must be executed before τ passes. Then,

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\lambda_{\tau-B_n}^n L\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n} \tag{6.5}$$

Let $\overset{+}{\lambda}_{\ell,n}$ denote the accumulated arriving rate of application n's tasks in ℓ ($\tau+1\leq\ell\leq\tau+B_n$). Let μ_ℓ^n denote the task service rate of servers of application n in ℓ ($\tau+1\leq\ell\leq\tau+B_n$). There are several prediction algorithms, *e.g.*, deep neural networks [212], to well predict $\overset{+}{\lambda}_{\ell,n}$. It is assumed that $\overset{+}{\lambda}_{\ell,n}$ is known in advance.

Similarly, at the beginning of time slot $\tau+b$, the number of application n's tasks that arrive in $\tau+b-B_n$ or earlier is $\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau-B_n+b}(\lambda_\ell^n L)$. The number of application n's tasks scheduled to the hybrid CDC by $\tau+b$ is the sum of $\overset{\diamond}{\mathbb{N}}_{\tau-1,n}$, and the number of application n's tasks executed in the hybrid CDC from τ to $\tau+b$. The number of application n's tasks executed in hybrid CDC in τ is $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n})$. Besides, the number of application n's tasks executed in the hybrid CDC in ℓ ($\tau+1\leq\ell\leq\tau+b$) is $(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}$. It is worth noting that all application n's tasks in $\tau+b-B_n$ or earlier have to be executed before $\tau+b$

passes. Then,

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau-B_n+b}\lambda_\ell^n L\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}+$$
$$\sum_{\ell=\tau+1}^{\tau+b}\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}\right) \tag{6.6}$$

The number of application n's tasks from $\tau-B_n$ to τ is $\sum_{\ell=\tau-B_n}^{\tau}\lambda_\ell^n L$. At the start of τ, $\overset{+}{\mathbb{N}}_{\tau,n}$ is the sum of $\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}$ and $\sum_{\ell=\tau-B_n}^{\tau}\lambda_\ell^n L$. Thus,

$$\overset{+}{\mathbb{N}}_{\tau,n}=\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau}\lambda_\ell^n L \tag{6.7}$$

At the beginning of τ, the number of tasks of application n executed in the hybrid CDC in τ is $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n})$. Besides, at the beginning of τ, the total number of application n's tasks executed in the hybrid CDC in time slot ℓ ($\tau+1\leq\ell\leq\tau+B_n$) is $(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n})$. Thus, in time slot τ, the total number of application n's tasks executed before $\tau+B_n$ passes is:

$$\overset{\diamond}{\mathbb{N}}_{\tau+B_n,n}=\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}+$$
$$\sum_{\ell=\tau+1}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}\right) \tag{6.8}$$

Thus, the conservation of application n's tasks requires that $\overset{+}{\mathbb{N}}_{\tau,n}$ should equal $\overset{\diamond}{\mathbb{N}}_{\tau+B_n,n}$, *i.e.*, $\overset{+}{\mathbb{N}}_{\tau,n}=\overset{\diamond}{\mathbb{N}}_{\tau+B_n,n}$. Then,

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau}\lambda_\ell^n L=\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}+$$
$$\sum_{\ell=\tau+1}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}\right) \tag{6.9}$$

Thus, constraints (6.5), (6.6), and (6.9) ensure delay bound constraints of tasks of all applications are strictly met.

Then, the power consumption model is introduced. The total amount of energy consumed in the CDC is the sum of the total amount of energy consumed in servers and facilities, *e.g.*, cooling and lighting. Similar to the work in [18], it is assumed that the energy consumed by each server for the same application is identical. The number of application n's tasks executed by its switched-on server per minute is denoted by $\overset{\bullet}{\mathbb{N}}_n$. Besides, $\overset{o}{N}_{\tau,n}$ denotes the number of such servers in τ. The service rate of servers for application n in τ is denoted by μ_τ^n. Then,

$$\mu_\tau^n = \overset{\bullet}{\mathbb{N}}_n \overset{o}{N}_{\tau,n} \tag{6.10}$$

The total number of application n's servers is denoted by $\hat{N}_n$. $\overset{o}{N}_{\tau+b,n}$ denotes the number of switched-on servers for application n in $\tau+b$ and it cannot exceed $\hat{N}_n$,

$$\overset{o}{N}_{\tau+b,n} \leq \hat{N}_n, 0 \leq b \leq \Upsilon \tag{6.11}$$

The total energy consumption of the hybrid CDC is the sum of the energy consumed by servers and facilities. The power usage effectiveness (PUE) is a critical metric to measure energy efficiency of a data center and it is the ratio of the total energy consumption of the CDC to the total energy consumed by servers. PUE of the CDC, denoted by α, is 1.2–2.0 [24]. Let $\check{\Phi}_n$ and $\hat{\Phi}_n$ denote the idle and peak power of each server for application n, respectively. Besides, its average CPU utilization in τ is denoted by u_τ^n. Thus, based on the work in [24], the total power consumed by the CDC in τ is obtained as:

$$\tilde{P}_\tau = \sum_{n=1}^{\mathcal{N}^A} \left(\overset{o}{N}_{\tau,n} (\check{\Phi}_n + (\gamma - 1)\hat{\Phi}_n + (\hat{\Phi}_n - \check{\Phi}_n) u_\tau^n) \right) \tag{6.12}$$

where δ_τ^n denotes the task loss possibility. The number of tasks that an application n's switched-on server executes in τ is calculated as:

$$\frac{L\left(1-\delta_\tau^n\right)\overset{+}{\lambda}_{\tau,n}}{\overset{o}{N}_{\tau,n}} \tag{6.13}$$

The busy time of each switched-on server for application n is $\frac{L\left(1-\delta_\tau^n\right)^+\lambda_{\tau,n}}{\overset{\bullet}{\mathbb{N}}_n \overset{o}{N}_{\tau,n}}$ minutes. u_τ^n is obtained as:

$$u_\tau^n = \frac{\left(1-\delta_\tau^n\right)^+\lambda_{\tau,n}}{\overset{\bullet}{\mathbb{N}}_n \overset{o}{N}_{\tau,n}} \tag{6.14}$$

Let E_τ denote the total energy consumption of the CDC in τ. Based on equations (6.10), (6.12), and (6.14), it is obtained as:

$$E_\tau = \sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_n^5 \mu_\tau^n + \Delta_n^4 {}^+\lambda_{\tau,n}(1-\delta_\tau^n)}{\overset{\bullet}{\mathbb{N}}_n} L \right) \tag{6.15}$$

where

$$\Delta_n^5 \triangleq \check{\Phi}_n + (\alpha - 1)\hat{\Phi}_n \tag{6.16}$$

$$\Delta_n^4 \triangleq \hat{\Phi}_n - \check{\Phi}_n \tag{6.17}$$

The available amount of energy in the CDC is denoted by $\hat{\mathbb{E}}$. Thus, the total energy consumed in τ or $\tau+b$ $(1 \leq b \leq \Upsilon)$ satisfies:

$$\sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_n^5 \mu_\tau^n + \Delta_n^4 {}^+\lambda_{\tau,n}(1-\delta_\tau^n)}{\overset{\bullet}{\mathbb{N}}_n} L \right) \leq \hat{\mathbb{E}} \tag{6.18}$$

$$\sum_{n=1}^{\mathcal{N}^A} \left(\frac{\Delta_n^5 \mu_{\tau+b}^n + \Delta_n^4 {}^+\lambda_{\tau+b,n}(1-\delta_{\tau+b}^n)}{\overset{\bullet}{\mathbb{N}}_n} L \right) \leq \hat{\mathbb{E}}, 1 \leq b \leq \Upsilon \tag{6.19}$$

Let $\overset{\circ}{E}_\tau$ and $\widetilde{E}_\tau$ denote the solar and wind energy consumed by tasks of all applications in the CDC in time slot τ, respectively. Besides, $\overset{\circ}{E}_\tau$ and $\widetilde{E}_\tau$ are obtained with equations (3.5) and (3.6) in the green energy model introduced in Section 6.2. Then, the way to calculate the total revenue and the total cost of the hybrid CDC is introduced, and the total profit of the hybrid CDC is given. Then, based on above constraints, the task scheduling problem is formulated as a constrained MINLP and solved by GSPSO.

Typically, an SLA is signed between the hybrid CDC and cloud users, and it guarantees the performance for tasks of each application. The revenues corresponding to the execution of application n's tasks in τ and $\tau+b$ are denoted by $f_{1,\tau}^{*,n}$ and $f_{1,\tau+b}^{*,n}$, respectively. The number of application n's tasks executed in the hybrid CDC in τ is $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V} z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n})$. Let ∇_τ^n denote the payment of each task of application n executed in τ. Then,

$$\eth_\tau^n=\left(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V} z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}\right)\nabla_\tau^n \tag{6.20}$$

The number of tasks of application n scheduled to execute in the hybrid CDC in $\tau+b$ is $(\overset{+}{\lambda}_{\tau+b,n}(1-\delta_{\tau+b}^n)L+\sum_{v=1}^{\mathcal{N}^V} z_{\tau+b}^{v,n}\overset{\#}{N}_{\tau+b,v,n})$. Let $\nabla_{\tau+b}^n$ denote the payment of each task of application n scheduled to execute in time slot $\tau+b$. Then, $f_{1,\tau+b}^{*,n}$ is obtained as follows.

$$f_{1,\tau+b}^{*,n}=\left(\overset{+}{\lambda}_{\tau+b,n}(1-\delta_{\tau+b}^n)L+\sum_{v=1}^{\mathcal{N}^V} z_{\tau+b}^{v,n}\overset{\#}{N}_{\tau+b,v,n}\right)\nabla_{\tau+b}^n \tag{6.21}$$

Let f_1 denote the total revenue of the hybrid CDC brought by tasks of all applications executed from time slots τ to $\tau+B_n$. In this way, f_1 is obtained and calculated as follows.

$$f_1=\sum_{n=1}^{\mathcal{N}^A}\left(\nabla_\tau^n+\sum_{b=1}^{B_n}\nabla_{\tau+b}^n\right) \tag{6.22}$$

Besides, prices of power grid in τ and $\tau+b$ are denoted by $\mathbb{p}_\tau$ and $\mathbb{p}_{\tau+b}$, respectively. Let f_2 denote the total cost of the hybrid CDC. f_2 includes two parts that are the grid energy cost of the CDC and the execution cost of VMs in public clouds. Thus, the amount of grid energy of the hybrid CDC in τ is $\mathbf{max}(E_\tau-\overset{\circ}{E}_\tau-\widetilde{E}_\tau,0)$. In addition, the amount of grid energy of the hybrid CDC in $\tau+b$ is $\mathbf{max}(E_{\tau+b}-\overset{\circ}{E}_{\tau+b}-\widetilde{E}_{\tau+b},0)$. The grid energy cost of the CDC brought by tasks of all applications in τ is $p_\tau(\mathbf{max}(E_\tau-\overset{\circ}{E}_\tau-\widetilde{E}_\tau,0))$.

Similarly, the grid energy cost of the CDC brought by tasks of all applications in $\tau+b$ $(1\leq b\leq \Upsilon)$ is $\mathbb{p}_{\tau+b}(\mathbf{max}(E_{\tau+b}-\overset{\circ}{E}_{\tau+b}-\widetilde{E}_{\tau+b},0))$. Besides, if application n's tasks are executed in public cloud v in $\tau+b$, $z^{v,n}_{\tau+b}=1$; otherwise, $z^{v,n}_{\tau+b}=0$. The number of application n's tasks executed in public cloud v in $\tau+b$ is denoted by $\overset{\#}{N}_{\tau+b,v,n}$. The prices of VMs for application n in public cloud v in τ and $\tau+b$ are denoted by $\nu^{v,n}_{\tau}$ and $\nu^{v,n}_{\tau+b}$, respectively. The average execution time of tasks of application n executed in public cloud v in τ and $\tau+b$ is denoted by $\beth^{v,n}_{\tau}$ and $\beth^{v,n}_{\tau+b}$, respectively. Hence, the execution cost of VMs in public clouds in τ is $\sum_{n=1}^{\mathcal{N}^A}\sum_{v=1}^{\mathcal{N}^V}(z^{v,n}_{\tau}\nu^{v,n}_{\tau}\beth^{v,n}_{\tau}\overset{\#}{N}_{\tau,v,n})$. Similarly, the execution cost of VMs in public clouds in $\tau+b$ is $\sum_{n=1}^{\mathcal{N}^A}\sum_{b=1}^{B_n}(\sum_{v=1}^{\mathcal{N}^C}(z^{v,n}_{\tau+b}\nu^{v,n}_{\tau+b}\beth^{v,n}_{\tau+b}\overset{\#}{N}_{\tau+b,v,n}))$. Then,

$$\begin{aligned} f_2= &\left(p_{\tau}(\mathbf{max}(E_{\tau}-\overset{\circ}{E}_{\tau}-\widetilde{E}_{\tau},0))+\sum_{n=1}^{\mathcal{N}^A}\sum_{v=1}^{\mathcal{N}^V}(z^{v,n}_{\tau}\nu^{v,n}_{\tau}\beth^{v,n}_{\tau}\overset{\#}{N}_{\tau,v,n})\right)\\ &+\sum_{b=1}^{\Upsilon}\left(\mathbb{p}_{\tau+b}(\mathbf{max}(E_{\tau+b}-\overset{\circ}{E}_{\tau+b}-\widetilde{E}_{\tau+b},0))\right)\\ &+\sum_{n=1}^{\mathcal{N}^A}\sum_{b=1}^{B_n}\left(\sum_{v=1}^{\mathcal{N}^C}(z^{v,n}_{\tau+b}\nu^{v,n}_{\tau+b}\beth^{v,n}_{\tau+b}\overset{\#}{N}_{\tau+b,v,n})\right) \end{aligned} \tag{6.23}$$

Let F_1 denote the profit of the hybrid CDC. F_1 is the difference between the total revenue of the hybrid CDC, f_1, and the total cost of the hybrid CDC, f_2 brought by tasks of all applications executed in time slots τ to $\tau+b$ $(1\leq b\leq\Upsilon)$. Then,

$$F_1=f_1-f_2 \tag{6.24}$$

Then, the profit maximization problem of the hybrid CDC is obtained as:

$$\mathbf{Max}\ \{F_1\}$$

subject to

$$\overset{o}{N}_{\tau+b,n}\leq\hat{N}_n, 0\leq b\leq\Upsilon \tag{6.25}$$

$$\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_n^5\mu_\tau^n+\Delta_n^4\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)}{\overset{\bullet}{\mathbb{N}}_n}L\right)\leq\hat{\mathbb{E}} \tag{6.26}$$

$$\sum_{n=1}^{\mathcal{N}^A}\left(\frac{\Delta_n^5\mu_{\tau+b}^n+\Delta_n^4\overset{+}{\lambda}_{\tau+b,n}(1-\delta_{\tau+b}^n)}{\overset{\bullet}{\mathbb{N}}_n}L\right)\leq\hat{\mathbb{E}}, 1\leq b\leq\Upsilon \tag{6.27}$$

$$\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\lambda_{\tau-B_n}^n L\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n} \tag{6.28}$$

$$\begin{aligned}\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau-B_n+b}\lambda_\ell^n L\leq\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}+\\ +\sum_{\ell=\tau+1}^{\tau+b}\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}\right)\end{aligned} \tag{6.29}$$

$$\begin{aligned}\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}+\sum_{\ell=\tau-B_n}^{\tau}\lambda_\ell^n L=\overset{\diamond}{\mathbb{N}}_{\tau-1,n}+\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}\\ +\sum_{\ell=\tau+1}^{\tau+B_n}\left(\overset{+}{\lambda}_{\ell,n}(1-\delta_\ell^n)L+\sum_{v=1}^{\mathcal{N}^V}z_\ell^{v,n}\overset{\#}{N}_{\ell,v,n}\right)\end{aligned} \tag{6.30}$$

$$\sum_{v=1}^{\mathcal{N}^V}z_\tau^{v,n}\leq 1, \sum_{v=1}^{\mathcal{N}^V}z_{\tau+b}^{v,n}\leq 1, 1\leq b\leq B_n \tag{6.31}$$

$$z_\tau^{v,n}, z_{\tau+b}^{v,n}\in\{0,1\}, 1\leq b\leq B_n \tag{6.32}$$

$$\mu_\tau^n\geq 0, \mu_{\tau+b}^n\geq 0, \overset{\#}{N}_{\tau,v,n}\geq 0, \overset{\#}{N}_{\tau+b,v,n}\geq 0, 1\leq b\leq B_n \tag{6.33}$$

$$\mu_{\tau+b}^n=0, \overset{\#}{N}_{\tau+b,v,n}=0, z_{\tau+b}^{v,n}=0, B_n<b\leq\Upsilon \tag{6.34}$$

The ranges of decision variables are given in constraints (6.33) and (6.34). The method to solve the formulated problem is presented in Section 6.3.

6.3 Temporal Task Scheduling

This section introduces the details of TTS. At the start of each time slot, TTS investigates the temporal variations in prices of power grid, revenue, and green energy in the CDC, and prices of VMs in public clouds within tasks' delay bound constraints.

Algorithm 4 TTS (Temporal Task Scheduling)

1: Set $\lambda_{\tau}^{n}(\Upsilon-B_n\leq\tau\leq\Upsilon-1)$, $\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}$ and $\overset{\diamond}{\mathbb{N}}_{\tau-1,n}$ to 0
2: Set $\overset{+}{\lambda}_{\tau,n}$ and $\lambda_{\tau}^{\dagger,n}$ $(\Upsilon\leq\tau\leq\mathcal{N}^{*})$ to λ_{τ}^{n}
3: $\tau \leftarrow \daleth$
4: **while** $\tau\leq\mathcal{N}^{*}$ **do**
5: Update $\overset{+}{\lambda}_{\tau,n}$ based on equation (6.4).
6: Solve the unconstrained problem with GSPSO
7: Execute $(\overset{+}{\lambda}_{\tau,n}(1-\delta_{\tau}^{n})L)$ tasks in the CDC, and schedule $z_{\tau}^{v,n}\overset{\#}{N}_{\tau,v,n}$ tasks to public cloud v
8: Update $\overset{+}{\lambda}_{\tau,n}$ and $\lambda_{\ell}^{\dagger,n}$ $(\tau-B_n\leq\ell\leq\tau)$
9: $\overset{+}{\mathbb{N}}_{\tau,n} \leftarrow \overset{+}{\mathbb{N}}_{\tau-1,n}+(\overset{+}{\lambda}_{\tau,n}(1-\delta_{\tau}^{n})L+\sum_{v=1}^{\mathcal{N}^{V}} z_{\tau}^{v,n}\overset{\#}{N}_{\tau,v,n})$
10: $\overset{\diamond}{\mathbb{N}}_{\tau-B_n,n} \leftarrow \overset{\diamond}{\mathbb{N}}_{\tau-B_n-1,n}+\lambda_{\tau-B_n}^{n}L$
11: $\tau \leftarrow \tau+1$
12: **end while**

Then, it is executed to determine the optimal task schedule that maximizes the profit of the hybrid CDC such that tasks' delay bound constraints are strictly met. In the formulated problem, the objective function is nonlinear with respect to decision variables. Decision variables $z_{\tau}^{v,n}$ and $z_{\tau+b}^{v,n}$ are integer variables while μ_{τ}^{n}, $\mu_{\tau+b}^{n}$, $\overset{\#}{N}_{\tau,v,n}$ and $\overset{\#}{N}_{\tau+b,v,n}$ $(1\leq b\leq\Upsilon,\ 1\leq n\leq\mathcal{N}^{A},\ 1\leq v\leq\mathcal{N}^{V})$ are continuous. Hence, the the formulated problem is a constrained MINLP. Similar to Section 3.3, this section adopts a penalty function method to convert the formulated problem into an unconstrained one. The vector of all decision variables is denoted by $\boldsymbol{x}$ and it consists of $z_{\tau}^{v,n}$, $z_{\tau+b}^{v,n}$, μ_{τ}^{n}, $\mu_{\tau+b}^{n}$, $\overset{\#}{N}_{\tau,v,n}$ and $\overset{\#}{N}_{\tau+b,v,n}$. Thus,

$$\underset{\boldsymbol{x}}{\mathbf{Min}} \left\{\widetilde{F_1}=\overset{\infty}{\mathcal{N}}\mho-F_1\right\} \tag{6.35}$$

$\widetilde{F_1}$ denotes the augmented objective function, $\overset{\infty}{\mathcal{N}}$ is a large positive constant and $\mho$ is the penalty of all constraints. $\mho$ is obtained with equation (3.21) in Chapter 3.

The pseudo codes of TTS are shown and explained in Algorithm 4. Line 1 initializes $\lambda_{\tau}^{n}(\Upsilon-B_n\leq\tau\leq\Upsilon-1)$, $\overset{+}{\mathbb{N}}_{\tau-B_n-1,n}$ and $\overset{\diamond}{\mathbb{N}}_{\tau-1,n}$ to 0. Line 2 initializes $\lambda_{\tau}^{\dagger,n}$ and $\overset{+}{\lambda}_{\tau,n}$ $(\Upsilon\leq\tau\leq\mathcal{N}^{*})$ with λ_{τ}^{n}. Let $\mathcal{N}^{*}$ denote the total number of time slots. $\overset{+}{\lambda}_{\tau,n}$ is

updated based on equation (6.4) in Line 5. Line 6 solves the unconstrained problem with GSPSO to determine μ_τ^n, $z_\tau^{v,n}$ and $\overset{\#}{N}_{\tau,v,n}$. Line 7 executes $(\overset{+}{\lambda}_{\tau,n}(1-\delta_\tau^n)L)$ tasks in the CDC, and schedules $z_\tau^{v,n}\overset{\#}{N}_{\tau,v,n}$ tasks to public cloud v. Then, Lines 9–10 update $\overset{+}{\mathbb{N}}_{\tau,n}$ and $\overset{\diamond}{\mathbb{N}}_{\tau-B_n,n}$.

It is worth noting that the unconstrained problem is an MINLP. Currently, there are many existing algorithms including a conjugate gradient method and sequential quadratic programming to solve it. However, they often depend on derivatives of objective function $\widetilde{F_1}$. Therefore, they are only useful when optimization problems own special mathematical characteristics. Besides, their optimization process is complicated and solutions are usually low-quality. Meta-heuristic algorithms own advantages including easy implementation and robustness, and they can avoid drawbacks of above algorithms. Nevertheless, each meta-heuristic algorithm has its advantages and disadvantages [60]. Thus, this chapter applies a hybrid algorithm called GSPSO, which is described in Chapter 5.

In PSO, the position and velocity of each particle are updated according to learning experiences of its own and the swarm. The size of the swarm is denoted by $|\mathbb{X}|$. The velocity of particle i (i=1, 2, ..., $|\mathbb{X}|$) is denoted by θ_1^i. The dimension of each particle's position is denoted by $\mathbb{N}^D$. μ_τ^n and $\mu_{\tau+b}^n(1\leq b\leq\Upsilon)$ are stored in the first $(\Upsilon+1)*\mathcal{N}^A$ elements of each vector. $z_\tau^{v,n}$ and $z_{\tau+b}^{v,n}(1\leq b\leq\Upsilon)$ are stored in the next $(\Upsilon+1)*\mathcal{N}^A*\mathcal{N}^C$ elements of each vector. $\overset{\#}{N}_{\tau,v,n}$ and $\overset{\#}{N}_{\tau+b,v,n}(1\leq b\leq\Upsilon)$ are stored in the following $(\Upsilon+1)*\mathcal{N}^A*\mathcal{N}^C$ elements of each vector. The value of $\widetilde{F_1}$ is stored in the last element of each vector. Therefore, $\mathbb{N}^D=(\Upsilon+1)*\mathcal{N}^A*(2*\mathcal{N}^C+1)+1$. At last, the globally optimal position of the swarm is transformed into decision variables including μ_τ^n, $\mu_{\tau+b}^n$, $z_\tau^{v,n}$, $z_{\tau+b}^{v,n}$, $\overset{\#}{N}_{\tau,v,n}$, and $\overset{\#}{N}_{\tau+b,v,n}$ ($1\leq b\leq\Upsilon$, $1\leq n\leq\mathcal{N}^A$, $1\leq v\leq\mathcal{N}^C$).

The complexity of Algorithm 4 is analyzed as follows. According to Chapter 5, the time complexity of GSPSO is $\mathcal{O}(\hat{g}|\mathbb{X}|\mathbb{N}^D)$. $\hat{g}$ denotes the total number of iterations in GSPSO. Similarly, in Algorithm 4, most of the running time is caused by the **while**

loop where $\mathcal{N}^*-\Upsilon+1$ iterations are executed. In Algorithm 4, the time complexity of each iteration in the **while** loop is mainly determined by GSPSO in Line 6. In addition, $\mathcal{N}^*$ is usually much larger than Υ. Consequently, the time complexity of Algorithm 4 is $\mathcal{O}(\mathcal{N}^*\hat{g}|\mathbb{X}|\mathbb{N}^D)$.

6.4 Performance Evaluation

This section evaluates TTS with real-life data, *e.g.*, VM prices, prices of power grid, arriving tasks and green energy data.

6.4.1 Parameter Setting

Similar to the work in [21], delay bound constraints are set to 3, 4 and 5 time slots, *i.e.*, B_1=3, B_2=4 and B_3=5. Besides, the real-life prices of power grid in 24 hours in New York, U.S. are chosen. According to the work in [24], the parameters in the power consumption model are set as: $\check{\Phi}_1$=200 (W), $\check{\Phi}_2$=100 (W), $\check{\Phi}_3$=50 (W), $\hat{\Phi}_1$=400 (W), $\hat{\Phi}_2$=200 (W), $\hat{\Phi}_3$=100 (W), $\hat{\mathbb{E}}$=5 (MWH), $\overset{\bullet}{\mathbb{N}}_1$=0.05 tasks/minute, $\overset{\bullet}{\mathbb{N}}_2$=0.1 tasks/minute, $\overset{\bullet}{\mathbb{N}}_3$=0.2 tasks/minute and α=1.2. Based on the work in [31], $\hat{N}_1$=3×10^6, $\hat{N}_2$=1.5×10^6, $\hat{N}_3$=3×10^6, $\hat{\mathbb{Q}}_1$=12, $\hat{\mathbb{Q}}_2$=25 and $\hat{\mathbb{Q}}_3$=50.

Similar to the work in [21], it is assumed that a single task finishes its execution in the hybrid CDC in each time slot. Thus, tasks' execution time for each application is randomly produced based on the uniform distribution in the range of $(0,L)$. Based on the work in [99], the prices (\$/hour) of tasks executed in the hybrid CDC are randomly produced based on the uniform distribution in the ranges of [0.24,0.48], [0.16,0.32] and [0.08,0.16], respectively. In this way, ∇_τ^n is obtained.

It is worth noting that many algorithms are sensitive to the parameter setting. Thus, based on the setting of parameters in existing studies [25, 76, 207], the parameters are set as follows. ψ_1 denotes the solar-irradiance-to-electricity conversion rate of CDC. ψ_2 denotes the active irradiance area of solar panels in CDC. ϕ_1 denotes the conversion rate of wind to electricity in CDC. ϕ_2 denotes the on-site air density

Table 6.1 Ranges of VM Prices ($/hour)

Public clouds	Small	Large	Xlarge
1	[0.07,0.08]	[0.06,0.07]	[0.18,0.20]
2	[0.14,0.16]	[0.12,0.14]	[0.10,0.12]
3	[0.22,0.24]	[0.20,0.22]	[0.05,0.06]

in CDC. ϕ_3 denotes the rotor area of wind turbines in CDC. According to the work in [76], ψ_1=0.2, ψ_2=1.5$*$10^5 (m^2), ϕ_1=0.3, ϕ_2=1.225 (kg/m^3) and ϕ_3=2.5$*$10^5 (m^2). The power ratings of wind and solar energy are set to 9$*$10^8 (W) and 1.65$*$10^8 (W), respectively. Based on the work in [25], GSPSO's parameters in Algorithm 3 in Chapter 5 are set as: $\hat{\theta}_5^P$=0.95, $\check{\theta}_5^P$=0.4, w_{24}=0.01%, $\hat{\theta}_4^P$=95%, $|\mathbb{X}|$=100, $\hat{g}$=200, $\check{\theta}_1^P$=$\hat{\theta}_1^P$=0.5. The acceleration coefficient reflecting the influence of a super particle is 1.5. Besides, θ_3=0.975, θ_2^0=10^{30}, $\overset{\infty}{\mathcal{N}}$=10^{20} and $\overset{0}{\gamma}_1$=$\overset{0}{\gamma}_2$=2.

According to the pricing model in Amazon EC2[1], three types of VM instances including Small, Large and Xlarge in commercial public clouds are adopted. Similar to the work in [207], two most typical types of resources including CPU and memory are chosen to configure and describe VMs as they are the most important configurations in selecting a VM instance in public clouds. Specifically, the number of CPUs and memory of a Small VM instance are 1 and 1.7 GB, respectively. Those of a Large VM instance are 4 and 7.5 GB, respectively. Those of a Xlarge VM instance are 8 and 15 GB, respectively. In addition, it is assumed that applications 1–3 are executed in Small, Large and Xlarge VMs, respectively. Each cloud has three types of Small, Large and Xlarge instances. According to the work in [25], Table 6.1 presents the price ranges of three VM instances in public clouds.

6.4.2 Experimental Results

Figure 6.2 shows the hybrid CDC's profit and penalty of the final solution of TTS. It is observed that the penalty in almost all time slots is nearly zero. It demonstrates that TTS can converge to the close-to-optimal solution that meets all constraints in

[1]https://aws.amazon.com/cn/ec2/ (accessed on March 9, 2019).

the formulated problem. As is shown in (6.35), Figure 6.2 also demonstrates that TTS can maximize the profit of the hybrid CDC.

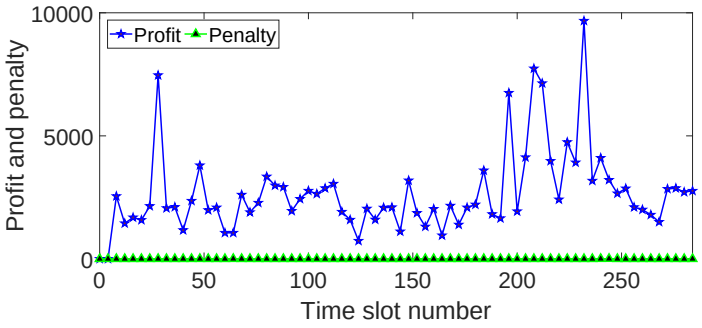


Figure 6.2 Profit and penalty in each time slot.

To demonstrate its performance, GSPSO is compared with PSO and SA. The reasons of choosing them for comparison are described here. It is demonstrated that SA is able to finally find global optima in theory by carefully setting the temperature cooling rate due to the fact that it is able to conditionally escape from local optima. Consequently, the comparison between them demonstrates the accuracy of GSPSO's final solution. In addition, it is shown that the convergence speed of PSO is quick [60]. The comparison among PSO, SA and GSPSO demonstrates the convergence speed of GSPSO.

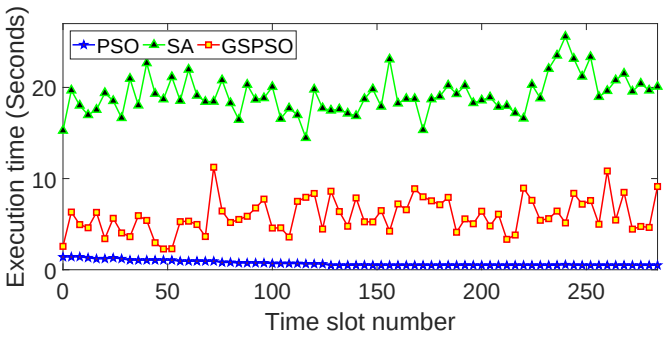


Figure 6.3 Comparison of execution time.

The execution time comparison of GSPSO, PSO and SA is shown in Figure 6.3. The average execution time of SA is 19.13 (seconds) and it is larger than that of PSO, 0.70 (seconds), and that of GSPSO, 5.91 (seconds). PSO's execution time is the least

because of its quick trap into locally optimal solutions. The profit comparison of GSPSO, PSO and SA in each iteration of time slot 50 is presented in Figure 6.4. The meaning of iterations in SA and PSO is similar to that of GSPSO. The evolutionary curve of penalty calculated according to equation (3.21) is shown in Figure 6.5.

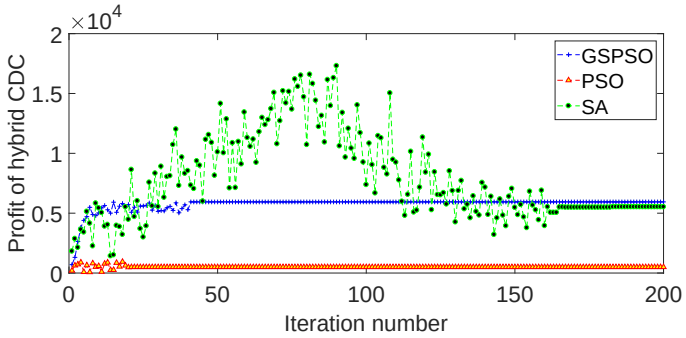


Figure 6.4 Profit of each iteration in time slot 50.

It is shown that PSO converges and loses its search ability after the least number of iterations. However, it is shown in Figure 6.5 that the penalty of the final solution of PSO is large (about 4×10^4). This shows that PSO's final solution cannot meet all constraints in the formulated problem. Thus, the final solution of PSO is the worst. Besides, SA converges to its final solution after about 165 iterations. The profit of its final solution is 11.11 times more than that of PSO but less than that of GSPSO. GSPSO converges to its final solution after 41 iterations and its profit is \$5941.06. GSPSO's profit is increased by \$414.42 in much fewer iterations and much less time than SA.

In addition, it is shown in Figure 6.5 that the penalty of the final solution of GSPSO is nearly 0. This result demonstrates that GSPSO converges to a high-quality solution satisfying all constraints in the formulated problem. Consequently, Figures 6.3–6.5 show that the incorporation of superior particles produced by SA's Metropolis acceptance rule and GA's genetic operations in PSO increases the quality of GSPSO's final solution and leads to larger profit for the hybrid CDC.

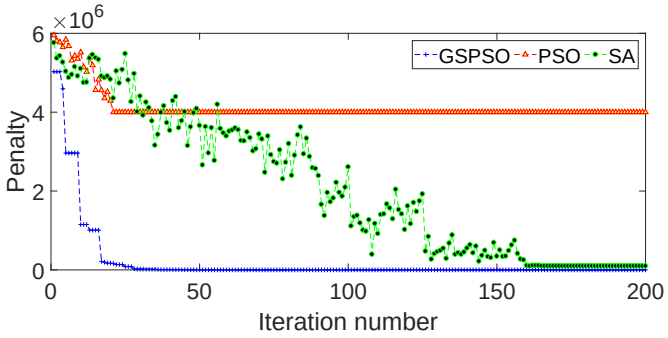


Figure 6.5 Penalty of each iteration in time slot 50.

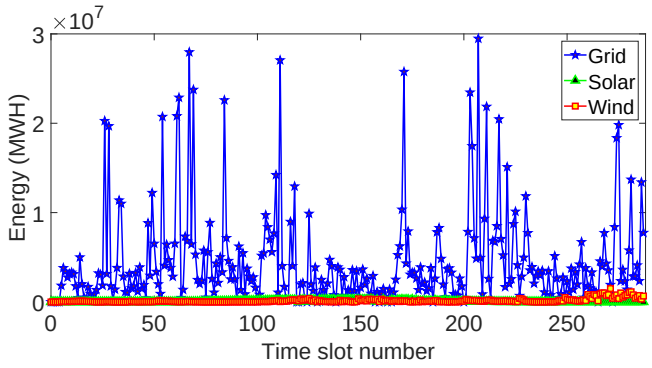
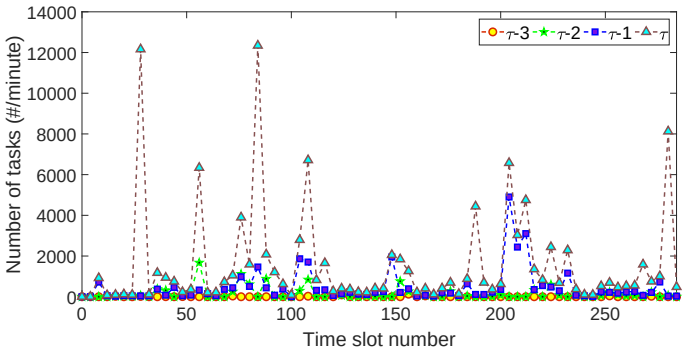


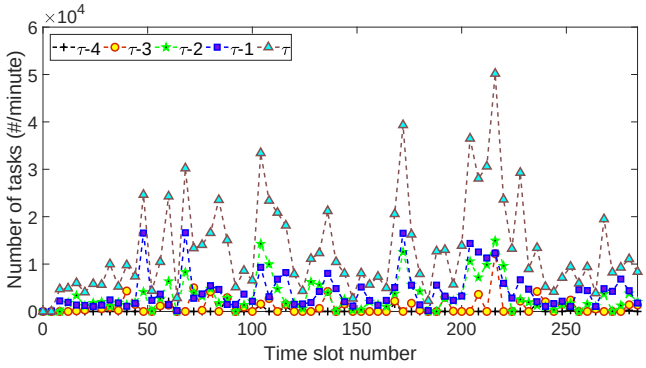
Figure 6.6 Consumption of grid and green energy.

Figure 6.6 illustrates the amount of grid and green energy consumed by the CDC. Here, $\hat{\mathbb{E}}$=5×10^7 (MWH) and Figure 6.6 illustrates that the sum of grid and green energy is less than $\hat{\mathbb{E}}$ in each time slot. It is pointed out that TTS prefers to first adopt wind and solar energy, and then the grid energy purchased from the power grid. This means that the grid energy is consumed only if the sum of solar and wind energy is not sufficient to execute all arriving tasks of each application. It is worth noting that TTS prefers to first adopt wind energy rather than the solar energy when they are both available. The reason is that the power rating of wind energy is more than five times larger than solar energy.

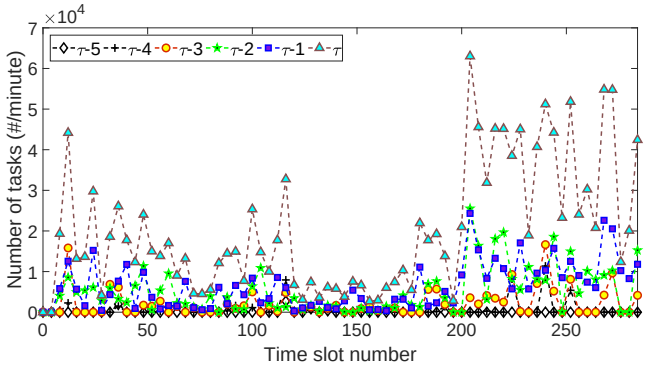
Figure 6.7 illustrates the cumulative arriving and remaining tasks of each application in private CDC and public clouds. For example, Figure 6.7(b) presents them for application 2 where $\overset{+}{\lambda}_{\tau,2}$=$\lambda^2_\tau$ + $\lambda^{\dagger,2}_{\tau-1}$+$\lambda^{\dagger,2}_{\tau-2}$+$\lambda^{\dagger,2}_{\tau-3}$+$\lambda^{\dagger,2}_{\tau-4}$. Here, the curve for time slot τ shows the variation of $\overset{+}{\lambda}_{\tau,2}$, and those for time slots τ−1, τ−2, τ−3 and



(a) Type 1



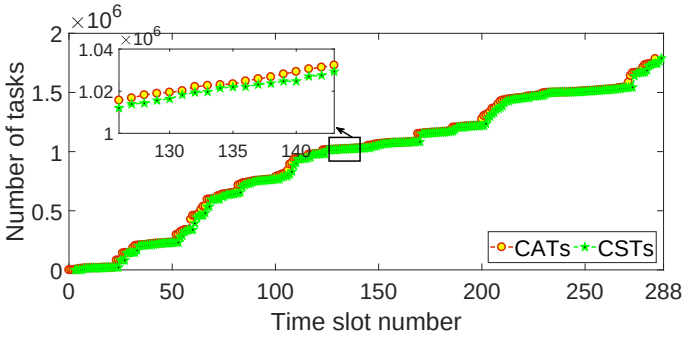
(b) Type 2



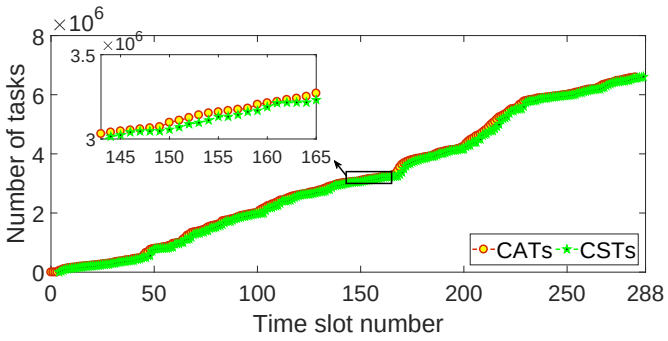
(c) Type 3

Figure 6.7 Accumulated and remaining tasks.

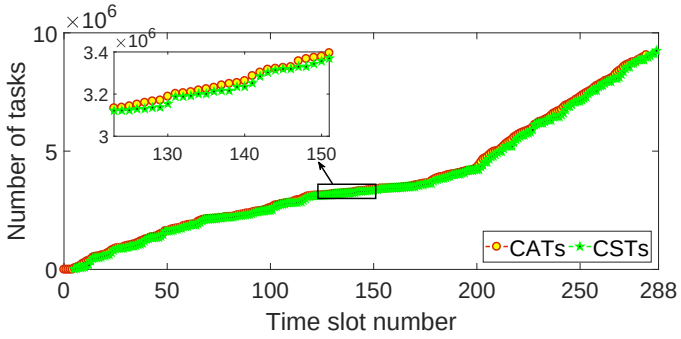
$\tau-4$ show the variations of $\lambda^{\dagger,2}_{\tau-1}$, $\lambda^{\dagger,2}_{\tau-2}$, $\lambda^{\dagger,2}_{\tau-3}$, and $\lambda^{\dagger,2}_{\tau-4}$, respectively. It is observed that $\lambda^{\dagger,2}_{\tau-4}$ is least among $\lambda^{\dagger,2}_{\tau-1}$, $\lambda^{\dagger,2}_{\tau-2}$, $\lambda^{\dagger,2}_{\tau-3}$, and $\lambda^{\dagger,2}_{\tau-4}$. The reason is that TTS puts all arriving tasks of each application into its seperate FCFS queues, and prefers to schedule earlier-arrived tasks. TTS aims to guarantee that by time slot τ, all tasks of application 2 in time slot $\tau-4$ or before must have been scheduled to the CDC and public clouds before time slot τ passes. Tasks that arrive in time slots $\tau-3$, $\tau-2$



(a) Type 1



(b) Type 2

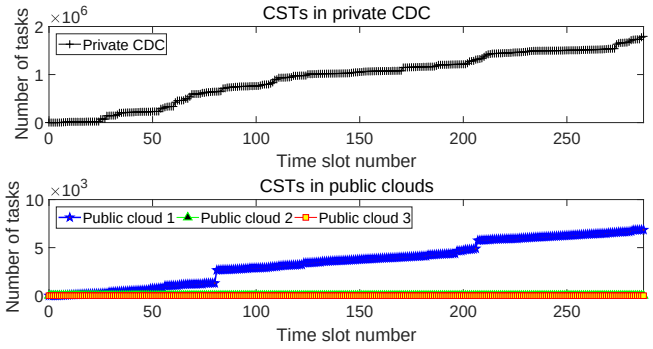


(c) Type 3

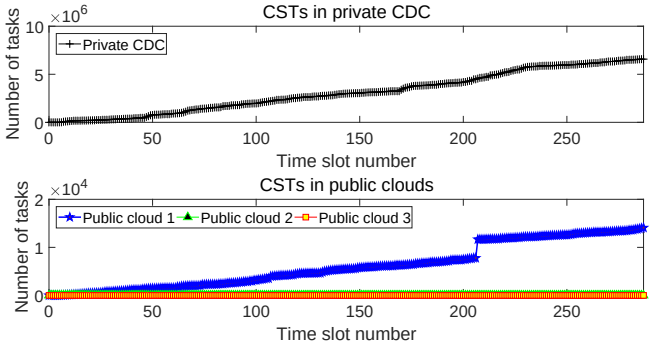
Figure 6.8 CATs and CSTs.

and $\tau-1$ can only be executed when $\lambda^{\dagger,2}_{\tau-4}=0$. Similarly, tasks that arrive in time slots $\tau-2$ and $\tau-1$ can only be executed when $\lambda^{\dagger,2}_{\tau-4}=0$ and $\lambda^{\dagger,2}_{\tau-3}=0$. Tasks arriving in time slot $\tau-1$ are executed when all tasks arriving before have been executed, *i.e.*, $\lambda^{\dagger,2}_{\tau-4}=0$, $\lambda^{\dagger,2}_{\tau-3}=0$ and $\lambda^{\dagger,2}_{\tau-2}=0$. Then, it is shown that $\lambda^{\dagger,2}_{\tau-1}$ is larger than $\lambda^{\dagger,2}_{\tau-4}$, $\lambda^{\dagger,2}_{\tau-3}$ and $\lambda^{\dagger,2}_{\tau-2}$ in each time slot.

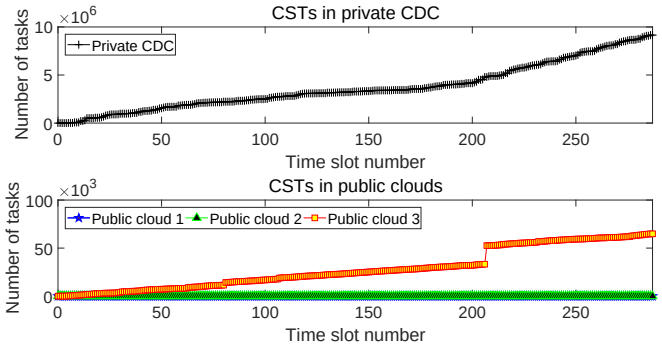
The cumulative scheduled tasks (CSTs) and cumulative arriving tasks (CATs) for each application are shown in Figure 6.8. It is shown that tasks of all applications



(a) Type 1



(b) Type 2



(c) Type 3

Figure 6.9 CSTs in the CDC and public clouds.

are scheduled to the CDC and public clouds within their delay bound constraints. For example, the number of CATs of application 1 in time slot 143 equals that of CSTs in time slot 140. This means that tasks of application 1 arriving in time slot 140 or before are all scheduled to the CDC and public clouds before time slot 143 passes. Therefore, it demonstrates that TTS strictly meets delay bound constraints of tasks of all applications.

Figure 6.9 shows CSTs in the private CDC and public clouds for each application in each time slot. It is shown that the number of tasks executed in the CDC is much larger than that of any public cloud. The reason is that the CDC aims to schedule all in the cost-effective way. TTS tries to execute tasks in the CDC, and therefore, it maximizes the profit of hybrid CDC. In addition, the number of tasks executed in different public clouds shows the differences of prices of VMs. For example, it is shown in Figure 6.9(a) that the number of application 1 tasks executed in public cloud 1 is much larger than those of public clouds 2 and 3 in each time slot. The reason is that the price of VMs in public cloud 1 is smaller than those of public clouds 2 and 3. Consequently, the number of tasks of each type executed in three public clouds is the reflection of the variations in prices of their VMs. It demonstrates that the profit of the hybrid CDC is maximized by smartly scheduling tasks of all applications among the private CDC and public clouds.

TTS is further compared with several typical algorithms [31, 76, 99] to demonstrate its effectiveness with respect to the profit of the hybrid CDC. The following algorithms are adopted to evaluate TTS.

1) A1. It is based on the cheap-electricity-first scheduling [76] and schedules all arriving tasks in the selected time slot when the power grid price is minimum within their corresponding delay bound constraints.

2) A2. It ignores the temporal variations in revenue, prices of public clouds, prices of power grid, solar irradiance and wind speed [99]. Therefore, all arriving tasks of each application are immediately and directly executed among the CDC and public clouds in the time slot when they arrive.

3) A3. It is based on a renewable energy-first scheduling algorithm [31] and schedules all arriving tasks in the selected time slot in which the total amount of solar and wind energy is maximum within their delay bound constraints.

A1–A3 schedule arriving tasks in a best-effort manner, since they depend on the number of arriving tasks, the total number of servers and the amount of energy in the CDC. Besides, in TTS, the final solution in time slot τ is obtained by solving P_2 given the predicted information over L.

Furthermore, the performance of TTS versus A1–A3 is evaluated with respect to the cumulative total profit. In addition, the reason why the proposed algorithm outperforms A1–A3 is further explained as follows. Note that the total number of servers and the maximum amount of available energy in the CDC in any time slot are fixed and limited. Therefore, some arriving tasks of applications might be refused by the CDC, and they have to be scheduled to public clouds. Note that the hybrid CDC has to pay the execution cost of VMs to public clouds due to tasks scheduled to them, and the execution cost of the CDC is larger than those of public clouds in each time slot according to the SLA specified between the hybrid CDC and users [25, 99]. This means A1–A3 have to schedule more tasks to public clouds, and they bring more execution cost to the hybrid CDC and reduce its profit. Consequently, to maximize the profit of the hybrid CDC, TTS inclines to schedule more arriving tasks to the CDC.

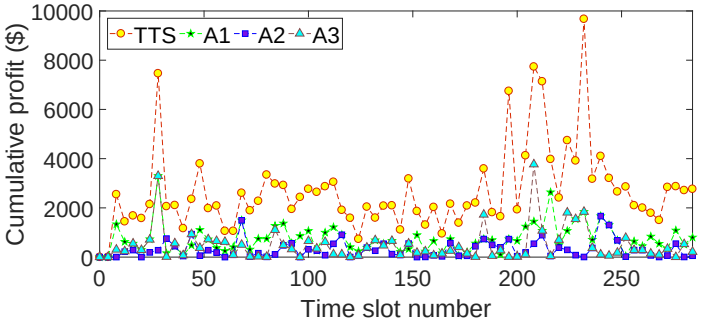


Figure 6.10 Cumulative total profits of TTS, A1–A3.

The cumulative total profits of TTS, and A1–A3 are presented in Figure 6.10. The cumulative total profit in τ is obtained by calculating the sum of the profit of the hybrid CDC brought by the tasks executed in time slots τ to $\tau+\Upsilon$. Figure 6.10 shows that compared with A1–A3, TTS's cumulative total profit is much higher. This is because TTS smartly executes tasks among CDC and public clouds by investigating the temporal variations in prices of power grid, revenue, solar, and wind energy in the CDC, and prices of public clouds within delay bound constraints of tasks.

6.5 Summary

An increasing number of companies deploy their delay-constrained applications in cloud data centers (CDCs). The unprecedented growth of tasks significantly increases the energy consumption and therefore, a hybrid cloud scheme is growingly chosen to tackle aperiodicity and uncertainty in tasks. The temporal variations in prices of power grid, revenue, solar irradiance, wind speed and prices of public clouds bring a big challenge to cost-effectively execute all tasks among the CDC and public clouds while strictly satisfying all tasks' delay bound constraints. This chapter presents a Temporal Task Scheduling (TTS) algorithm that investigates the temporal variations. It can smartly schedule all tasks to the CDC and public clouds within delay bound constraints. Besides, the mathematical relation between task refusal and service rates is explicitly presented. Then, the profit maximization problem is solved with a novel hybrid optimization algorithm. Extensive simulation experiments demonstrate that TTS outperforms several existing scheduling algorithms in terms of profit.

CHAPTER 7

PROFIT AND QOS-OPTIMIZED TASK SCHEDULING FOR DISTRIBUTED GREEN DATA CENTERS

This chapter presents the details of the proposed Profit and Quality of service (QoS)-optimized Task Scheduling (PQTS) algorithm, and it is organized as follows. Section 7.1 formulates a profit maximization and average task loss possibility minimization problem for a cloud data center (CDC) provider as a bi-objective optimization problem. The proposed Simulated-annealing-based Bi-objective Differential Evolution (SBDE) algorithm is described in Section 7.2. According to SBDE, a PQTS method is proposed to trade off the profit of the CDC provider against the average task loss possibility of all applications. Then, PQTS is evaluated by using realistic data, and its experimental results and analysis are shown in Section 7.3. Section 7.4 concludes this chapter.

7.1 Problem Formulation

This section formulates a bi-objective optimization problem for a CDC provider. Figure 7.1 illustrates a system architecture of CDCs. Users send their tasks through different types of electronic devices. Then, these tasks are classified by *Classifier* according to their types [6, 109, 227]. Tasks of the same application are enqueued into their separate First-In-First-Out (FIFO) queues. To guarantee the application availability and QoS [228] of tasks, an application is deployed in multiple CDCs located in different locations. Then, these tasks are delivered to CDCs through K ISPs. Each CDC manages a cluster of high-performance servers. It is assumed that it obtains energy from three types of sources including solar panels, wind turbines and non-renewable power grid. By exploiting the architecture of CDCs, this chapter focuses on *Task Scheduler* that executes the proposed PQTS and specifies the Pareto

optimal task service rates and task split among ISPs for CDCs in each time slot to jointly optimize profit and QoS.

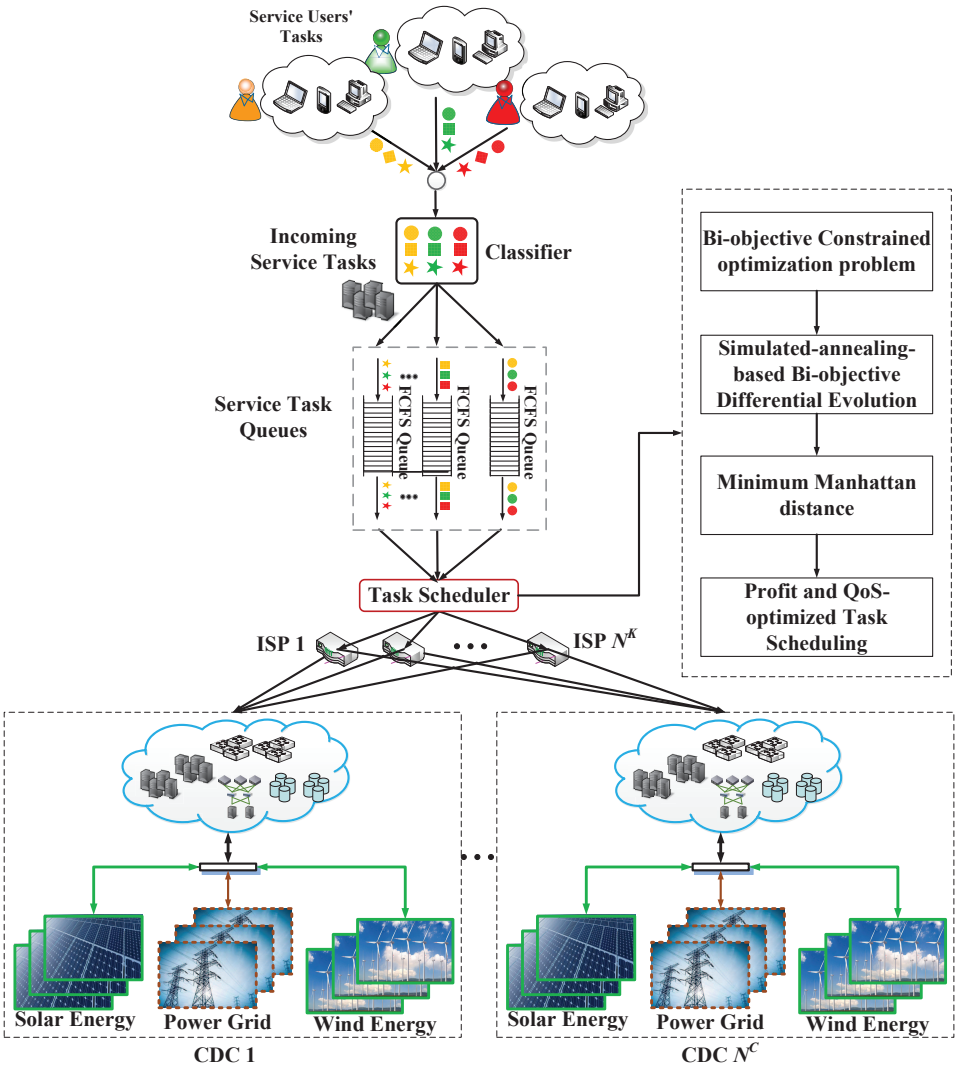


Figure 7.1 Architecture of CDCs.

7.1.1 Delay Constraint

Let $\mathcal{N}^C$ denote the number of CDCs. Let $\mathcal{N}^A$ denote the number of applications. Similar to the work in [110], this chapter adopts an $M/M/1/\hat{Q}_n^c/\infty$ queueing system to model the performance of servers of application n ($1 \leq n \leq \mathcal{N}^A$) in CDC c ($1 \leq c \leq \mathcal{N}^C$). Let $\hat{Q}_n^c$ denote the capacity of a task queue corresponding to application n in CDC c. Let $\hat{T}_n$ denote a specified response time constraint of tasks of application n. Let $\lambda_\tau^{c,n}$ denote the arriving rate of tasks of application n in CDC c in time slot τ. Let $\mu_\tau^{c,n}$

denote the service rate of servers corresponding to application n in CDC c in time slot τ. Let $T_\tau^{c,n}$ denote the average response time of tasks of application n in CDC c at time slot τ. According to the work in [110], $T_\tau^{c,n}$ is obtained with (7.1) and it needs to be less than or equal to $\hat{T}_n$, *i.e.*,

$$T_\tau^{c,n}=\frac{\Delta_{\tau,c,n}^{11}}{\mu_\tau^{c,n}\left(1-\Delta_\tau^{12,c,n}\right)}\leq\hat{T}_n \tag{7.1}$$

where

$$\Delta_{\tau,c,n}^{11}=\frac{\rho_\tau^{c,n}}{1-\rho_\tau^{c,n}}-\frac{\left(\hat{Q}_n^c+1\right)\left(\rho_\tau^{c,n}\right)^{\hat{Q}_n^c+1}}{1-\left(\rho_\tau^{c,n}\right)^{\hat{Q}_n^c+1}}$$
$$\Delta_\tau^{12,c,n}=\frac{1-\rho_\tau^{c,n}}{1-\left(\rho_\tau^{c,n}\right)^{\hat{Q}_n^c+1}}$$
$$\rho_\tau^{c,n}=\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}$$

The service rate of tasks of application n means the rate at which tasks are scheduled to CDCs and removed from their corresponding FCFS queue. Then, the task loss possibility of application n in CDC c in time slot τ is:

$$\delta_\tau^{c,n}=\begin{cases}\frac{1-\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}}{1-\left(\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}\right)^{\hat{Q}_n^c+1}}\left(\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}\right)^{\hat{Q}_n^c} & \mu_\tau^{c,n}>0,\\ 1 & \mu_\tau^{c,n}=0.\end{cases} \tag{7.2}$$

To ensure the task queue stability of application n in CDC c, $\lambda_\tau^{c,n}$ must be less than $\mu_\tau^{c,n}$ in time slot τ. Let $\lambda_\tau^{k,c,n}$ denote the arriving rate of tasks of application n in CDC c delivered through ISP k in time slot τ. Then,

$$\lambda_\tau^{c,n}=\sum_{k=1}^{\mathcal{N}^K}\lambda_\tau^{k,c,n}<\mu_\tau^{c,n} \tag{7.3}$$

Let λ_τ^n denote the task arriving rate of application n in time slot τ. In each time slot, the sum of arriving rates of tasks of application n delivered through all ISPs to

all CDCs must equal λ_τ^n, *i.e.*,

$$\lambda_\tau^n = \sum_{c=1}^{\mathcal{N}^C} \lambda_\tau^{c,n} = \sum_{c=1}^{\mathcal{N}^C} \sum_{k=1}^{\mathcal{N}^K} \lambda_\tau^{k,c,n} \tag{7.4}$$

7.1.2 Energy Consumption Model

The energy consumption model used in CDCs is introduced here. Similar to the work in [110], it is assumed that the servers corresponding to the same application are homogeneous, and they consume the same energy. Let $\overset{\bullet}{\mathbb{N}}_{c,n}$ denote the number of application n's tasks executed by each corresponding switched-on server per minute in CDC c. Let $\overset{o}{N}_{\tau,c,n}$ denote the number of switched-on servers for application n in CDC c in time slot τ. Thus, $\overset{o}{N}_{\tau,c,n} = \mu_\tau^{c,n} / \overset{\bullet}{\mathbb{N}}_{c,n}$. Let $\hat{N}_{c,n}$ denote the maximum number of servers corresponding to application n in CDC c. Then, $\overset{o}{N}_{\tau,c,n}$ must not exceed $\hat{N}_{c,n}$, *i.e.*,

$$\overset{o}{N}_{\tau,c,n} \leq \hat{N}_{c,n} \tag{7.5}$$

Then, the total energy consumed by CDCs is the sum of energy consumption of all servers and the facilities including cooling and lighting in them. Let $\hat{\Phi}_n^c$ and $\check{\Phi}_n^c$ denote the peak and idle power of each server of application n in CDC c, respectively. The power usage effectiveness value of CDC c is denoted by α_c, and it is the ratio of the energy consumption of a CDC to that of its all servers. The power usage effectiveness is a significant metric to indicate the energy efficiency of a CDC, and its value is usually 1.2–2.0 for existing data centers [229].

Let L denote the length of each time slot. The number of tasks executed by each switched-on server corresponding to application n in time slot τ is calculated as:

$$\frac{\left(1-\delta_\tau^{c,n}\right) \lambda_\tau^{c,n} L}{\overset{o}{N}_{\tau,c,n}} \tag{7.6}$$

The busy period of each switched-on server of application n in CDC c in time slot τ is $\frac{\left(1-\delta_\tau^{c,n}\right)\lambda_\tau^{c,n} L}{\overset{\bullet}{\mathbb{N}}_{c,n} \overset{o}{N}_{\tau,c,n}}$ minutes. Let $u_\tau^{c,n}$ denote the average CPU utilization of servers

corresponding to application n in CDC c in time slot τ. Then,

$$u_{\tau}^{n}=\frac{(1-\delta_{\tau}^{c,n})\,\lambda_{\tau}^{c,n}}{\overset{\bullet}{\mathbb{N}}_{c,n}\overset{o}{N}_{\tau,c,n}} \tag{7.7}$$

Thus, following the work in [23], the total power of CDC c in time slot τ is obtained as:

$$P_{\tau}^{c}=\sum_{n=1}^{\mathcal{N}^{A}}(\overset{o}{N}_{\tau,c,n}(\check{\Phi}_{n}^{c}+(\alpha_{c}-1)\hat{\Phi}_{n}^{c}+(\hat{\Phi}_{n}^{c}-\check{\Phi}_{n}^{c})u_{\tau}^{c,n})) \tag{7.8}$$

Let E_{τ}^{c} denote the total energy consumed by tasks of all applications in CDC c in time slot τ. According to equations (7.7) and (7.8),

$$E_{\tau}^{c}=\sum_{n=1}^{\mathcal{N}^{A}}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau}^{c,n}+\Delta_{c,n}^{4}\lambda_{\tau}^{c,n}(1-\delta_{\tau}^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right) \tag{7.9}$$

where

$$\Delta_{c,n}^{5}\triangleq\check{\Phi}_{n}^{c}+(\alpha_{c}-1)\hat{\Phi}_{n}^{c},\ \Delta_{c,n}^{4}\triangleq\hat{\Phi}_{n}^{c}-\check{\Phi}_{n}^{c}$$

Let $\hat{\mathbb{E}}_{c}$ denote the available amount of the total energy in CDC c. Thus, E_{τ}^{c} must not exceed $\hat{\mathbb{E}}_{c}$, *i.e.*,

$$E_{\tau}^{c}=\sum_{n=1}^{\mathcal{N}^{A}}\left(\frac{\Delta_{c,n}^{5}\mu_{\tau}^{c,n}+\Delta_{c,n}^{4}\lambda_{\tau}^{c,n}(1-\delta_{\tau}^{c,n})}{\overset{\bullet}{\mathbb{N}}_{c,n}}L\right)\leq\hat{\mathbb{E}}_{c} \tag{7.10}$$

The solar and wind energy models are introduced. Let $\overset{\circ}{E}_{\tau,c}$ denote the amount of solar energy generated in CDC c in time slot τ. Let $\widetilde{E}_{\tau,c}$ denote the amount of wind energy generated in CDC c in time slot τ. Similar to Chapter 3, $\overset{\circ}{E}_{\tau+b,c}$ and $\widetilde{E}_{\tau+b,c}$ are calculated with equations (3.5) and (3.6) in the green energy model introduced in Section 3.2 in Chapter 3, respectively.

7.1.3 Constrained Optimization Problem

Let $\hat{B}_{k}$ denote the bandwidth limit of ISP k. Let ζ_{n} denote the average size of each task of application n. Then, the total bandwidth required by tasks of all applications

transmitted through ISP k in time slot τ satisfies:

$$\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}\left(\lambda_\tau^{k,c,n}\zeta_n\right)\leq\hat{B}_k \tag{7.11}$$

In real scenarios, users negotiate with current existing CDC providers and establish a service-level agreement (SLA) [20]. SLA specifies the revenue brought by tasks if their performance requirements are strictly fulfilled. In addition, the penalty cost paid by CDC providers is also specified in SLA if its performance requirement of each task is unmet.

Let ∇_τ^n denote the revenue brought by the execution of each task of application n in time slot τ. Let ϵ_τ^n denote the penalty cost brought by each rejected task of application n in time slot τ. The total revenue brought by tasks of application n in all CDCs in time slot τ is obtained as:

$$f_1^{*,n}=\sum_{c=1}^{\mathcal{N}^C}\left(\left(1-\delta_\tau^{c,n}\right)\lambda_\tau^{c,n}\nabla_\tau^n-\delta_\tau^{c,n}\lambda_\tau^{c,n}\epsilon_\tau^n\right)L \tag{7.12}$$

Then, the total revenue brought by tasks of all applications in all CDCs in time slot τ is obtained as:

$$f_1=\sum_{n=1}^{\mathcal{N}^A}f_1^{*,n} \tag{7.13}$$

The total cost of the CDC provider in time slot τ, denoted as f_2, consists of energy cost and ISP bandwidth cost of all tasks. Let f_{21} denote the ISP bandwidth cost caused by the transmission of tasks among users and CDCs in time slot τ. Let f_{22} denote the energy cost due to all tasks in all CDCs in time slot τ.

$$f_2=f_{21}+f_{22} \tag{7.14}$$

Let b_τ^k denote the unit bandwidth price of ISP k in time slot τ. Then,

$$f_{21}=\sum_{k=1}^{\mathcal{N}^K}\left(b_\tau^k\left(\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}\left(\lambda_\tau^{k,c,n}\zeta_n\right)\right)\right) \tag{7.15}$$

It is assumed that the solar and wind energy are cost-free once they are purchased and installed in CDCs. The amount of power grid energy consumed by all tasks is $\max\left(E_{\tau}^{c}-\overset{\circ}{E}_{\tau,c}-\widetilde{E}_{\tau,c},0\right)$. Let p_{τ}^{c} denote the price of power grid of CDC c in time slot τ. Thus, f_{22} is calculated as:

$$f_{22}=\sum_{c=1}^{\mathcal{N}^C}\left(p_{\tau}^{c}\left(\max\left(E_{\tau}^{c}-\overset{\circ}{E}_{\tau,c}-\widetilde{E}_{\tau,c},0\right)\right)\right) \tag{7.16}$$

Let F_1 denote the profit of CDC providers brought by the execution of all tasks. F_1 is calculated as:

$$F_1=f_1-f_2 \tag{7.17}$$

The first objective is to maximize F_1, *i.e.*,

$$\underset{\lambda_{\tau}^{k,c,n},\mu_{\tau}^{c,n}}{\mathbf{Max}}\ \{F_1\} \tag{7.18}$$

The average task loss possibility in time slot τ is:

$$\Delta_{\tau}^{13}=\frac{\sum_{c=1}^{\mathcal{N}^C}\sum_{n=1}^{\mathcal{N}^A}(\delta_{\tau}^{c,n})}{\mathcal{N}^C} \tag{7.19}$$

The second objective is to minimize Δ_{τ}^{13}, *i.e.*,

$$\underset{\lambda_{\tau}^{k,c,n},\mu_{\tau}^{c,n}}{\mathbf{Min}}\ \left\{\Delta_{\tau}^{13}\right\} \tag{7.20}$$

Then, this chapter formulates a bi-objective optimization problem as:

$$\underset{\lambda_{\tau}^{k,c,n},\mu_{\tau}^{c,n}}{\mathbf{Max}}\ \{F_1\}$$

$$\underset{\lambda_{\tau}^{k,c,n},\mu_{\tau}^{c,n}}{\mathbf{Min}}\ \left\{\Delta_{\tau}^{13}\right\}$$

subject to constraints (7.1), (7.3)–(7.5), (7.10), (7.11) and

$$\lambda_{\tau}^{k,c,n}\geq 0,\mu_{\tau}^{c,n}\geq 0 \tag{7.21}$$

Constraint (7.21) specifies ranges of decision variables, *i.e.*, $\lambda_\tau^{k,c,n}$ and $\mu_\tau^{c,n}$. There are many prediction algorithms, *e.g.*, radial basis function neural networks and support vector regression [230]–[232], to well predict time-related parameters, *e.g.*, p_τ^c, b_τ^k, λ_τ^n, $\psi_{\tau,3c}$ and $\phi_{\tau,4c}$. Therefore, similar to the work in [23], it is assumed that they are already known at the start of each time slot τ.

7.2 Simulated-annealing-based Bi-objective Differential Evolution

It is worth noting that F_1 and Δ_τ^{13} are nonlinear in terms of $\lambda_\tau^{k,c,n}$ and $\mu_\tau^{c,n}$. Consequently, the problem is a constrained nonlinear bi-objective optimization problem [233]–[235] . Constraints (7.1), (7.3)–(7.5), (7.10) and (7.11) are nonlinear and complex. Therefore, to well handle them, this section adopts a penalty function method to transform the problem into its corresponding unconstrained optimization one given as follows.

$$\begin{aligned} &\underset{\lambda_\tau^{k,c,n},\mu_\tau^{c,n}}{\textbf{Min}} \quad \left\{\widetilde{F_1}=\overset{\infty}{\mathcal{N}}\mho-F_1\right\} \\ &\underset{\lambda_\tau^{k,c,n},\mu_\tau^{c,n}}{\textbf{Min}} \quad \left\{\widetilde{\Delta_\tau^{13}}=\overset{\infty}{\mathcal{N}}\mho+\Delta_\tau^{13}\right\} \end{aligned} \tag{7.22}$$

where $\widetilde{F_1}$ and $\widetilde{\Delta_\tau^{13}}$ denote new objective functions, $\overset{\infty}{\mathcal{N}}$ denotes a large positive integer and $\mho$ denotes the total penalty corresponding to all constraints. Let $\boldsymbol{x}$ denote a vector including $\lambda_\tau^{k,c,n}$ and $\mu_\tau^{c,n}$. $\mho$ is calculated with equation (3.21) in Chapter 3. In this way, the unconstrained optimization problem is obtained and calculated. Currently, there are many traditional multi-objective optimization algorithms, *e.g.*, a weighted sum method, an ϵ-constraint method and a weighted metric method, to solve it. Each method has its own advantages and disadvantages. For example, it is difficult for the weighted sum method to specify its weight vectors to produce a Pareto-optimal solution in a selected objective space. Nevertheless, it cannot locate certain Pareto-optimal solutions in a non-convex objective space. An ϵ-constraint method is applicable to convex and non-convex optimization problems, but its ϵ

vector should be carefully set because it has to be in the valid range of each objective function. In addition, all these traditional approaches only produce one candidate solution. It is more desired to produce a set of candidate solutions. Then, the CDC providers can select their favorite solution from the set.

Thus, different multi-objective optimization algorithms are adopted to realize it. Though there are many similar algorithms that can be applied, the multi-objective differential evolutionary (MODE) algorithm has a simple, robust and efficient structure, and can tackle complex constraints and nonlinear objective functions. There are only a few parameters that need to be adjusted to obtain final solutions [236]. In addition, MODE has been validated by many real-life applications, *e.g.*, power flow optimization and facial expression recognition.

Consequently, this chapter adopts a Simulated-annealing-based Bi-objective Differential Evolution (SBDE) algorithm to solve the unconstrained optimization problem. It is a population-based algorithm for obtaining a limited set of Pareto optimal solutions. The Pareto dominance in iterations is enhanced and therefore, quality of solutions is improved. In SBDE, evolutionary operations, *e.g.*, adaptive mutation, simulated annealing (SA)-based crossover and SA-based selection, are performed to improve its convergence speed and accuracy. The adaptive elitist archive update mechanism is used to maintain the diversity of solutions, and an approximate well-located Pareto front is thus achieved.

Typically, the Pareto optimal set and Pareto front include numerous points. Numerically and practically, the Pareto front is approximated with a limited number of points, from which a final representative solution named a knee is selected among them. The knee is the most acceptable because it means a strategy where two objectives are traded off but the best overall performance is achieved [237]. The minimum Manhattan distance [238] is adopted to specify the knee from an approximate Pareto optimal set. It has several pros including matrix

computation-enabled efficient implementation, simple selection of the knee and geometrical representation Consequently, SBDE is widely adopted to solve different kinds of multi-objective constrained optimization problems. Then, according to the knee, the final task scheduling strategy is determined to specify the optimal task service rates and task split among ISPs for each CDC in each time slot.

7.2.1 Population Initialization

The population is randomly initialized according to a uniform distribution within the feasible search space of decision variables. Let $\mathbb{N}^D$ denote the number of decision variables. Let $\overset{0}{\mathbf{x}}_{i,d}$ denote the value of decision variable d of individual i ($i \in \{1, 2, \ldots, |\mathbb{X}|\}$) in the first generation, and $|\mathbb{X}|$ denotes the size of the population. The population is initialized as:

$$\overset{0}{\mathbf{x}}_{i,d} = \check{\boldsymbol{\theta}}_5^d + w_{13} * \left(\hat{\boldsymbol{\theta}}_5^d - \check{\boldsymbol{\theta}}_5^d\right) \tag{7.23}$$

where $\hat{\boldsymbol{\theta}}_5^d$ and $\check{\boldsymbol{\theta}}_5^d$ denote upper and lower bounds of decision variable d, respectively, and w_{13} is a random number uniformly distributed in (0,1).

7.2.2 Adaptive Mutation

Let $\hat{g}$ denote the total number of generations. In the traditional MODE, in each generation g ($g \in \{1, 2, \ldots, \hat{g}\}$), one widely used mutation operation in practice is known as $DE/best/1$ due to its fast convergence. It produces a new mutant individual $\grave{\mathbf{x}}_i^g$ for $\mathbf{x}_i^g$ by perturbing the best individual $\overset{\bullet}{\mathbf{x}}_g$ ($\overset{\bullet}{i} \neq w_{14} \neq w_{15} \neq i$) searched in current generation g, with the difference of two other randomly chosen individuals in the population, $\mathbf{x}_{w_{14}}^g$ and $\mathbf{x}_{w_{15}}^g$.

θ_1^D denotes a scaling factor that controls the perturbation and improves the convergence. In this chapter, the best individual is denoted by $\overset{\bullet}{i}$, and its solution is denoted by $\overset{\bullet}{\mathbf{x}}_g$. It is described as:

$$\grave{\mathbf{x}}_i^g = \overset{\bullet}{\mathbf{x}}_g + \theta_1^D \left(\mathbf{x}_{w_{14}}^g - \mathbf{x}_{w_{15}}^g\right), \overset{\bullet}{i} \neq w_{14} \neq w_{15} \neq i \tag{7.24}$$

It is worth noting that θ_1^D in equation (7.24) is a constant. It fails to dynamically adjust to the evolution of population because it controls the search process with a constant convergence rate. Thus, this section designs θ_1^D as a decreasing function shown as:

$$\theta_1^D = exp^{-\theta_2^D \frac{g}{\bar{g}}} \tag{7.25}$$

where θ_2^D is a positive constant. Other design of θ_1^D can be found in [227]. θ_1^D decreases as g increases.

Then, as is shown in equation (7.24), the evolution of population can explore the solution space widely at the start, and quickly at the end. Nevertheless, if the population traps into local optima, it cannot escape from them. As is shown in equation (7.24), the best individual, $\overset{\bullet}{\mathbf{x}}_g$, is critically important for a mutation operation. It is the one with the least value of an objective function for a single-objective optimization problem. However, the best individual means a group of Pareto optimal solutions to a multi-objective optimization problem. It is contradictive to guide individuals to a single solution because the goal is to obtain a set of Pareto solutions. Therefore, this chapter proposes an adaptive mutation operation to guide individuals to the Pareto set, and increase the global searching efficiency.

This chapter adopts an external archive (EA) denoted by $\breve{\Omega}$ to keep the Pareto optimal solutions searched so far. The best individual of a current population is chosen from $\breve{\Omega}$. Then, this chapter checks whether individual i is in $\breve{\Omega}$. If so, i is chosen as $\overset{\bullet}{\mathbf{x}}_g$. Otherwise, the elitist individual with the minimum Euclidean distance from individual i is chosen as $\overset{\bullet}{\mathbf{x}}_g$. Let $|\breve{\Omega}|$ denote the maximum size of $\breve{\Omega}$. $\overset{\longleftrightarrow}{i,j}$ denotes the Euclidean distance between individual i and elitist individual j in $\breve{\Omega}$. Based on above discussion, the adaptive mutation is implemented as:

$$\grave{\mathbf{x}}_i^g = \begin{cases} \mathbf{x}_i^g + exp^{-\theta_2^D \frac{g}{\bar{g}}} \left(\mathbf{x}_{w_{14}}^g - \mathbf{x}_{w_{15}}^g\right), & \text{if } \mathbf{x}_i^g \in \breve{\Omega} \\ \overset{\bullet}{\mathbf{x}}_g + exp^{-\theta_2^D \frac{g}{\bar{g}}} \left(\mathbf{x}_{w_{14}}^g - \mathbf{x}_{w_{15}}^g\right), & \text{otherwise} \end{cases} \tag{7.26}$$

$$\overset{\bullet}{i} = \arg_j \underset{j\in\{1,2,\ldots,|\breve{\Omega}|\}}{\text{Min}} \left(\overset{\longleftrightarrow}{i,j}\right) \tag{7.27}$$

$$\overset{\longleftrightarrow}{i,j} = \sqrt{\sum_{d=1}^{\mathbb{N}^D} \left(\mathbf{x}_{i,d}^g - \mathbf{x}_{j,d}^g\right)^2}, j\in\{1,2,\ldots,|\breve{\Omega}|\} \tag{7.28}$$

7.2.3 SA-based Crossover

After an adaptive mutation operation, a crossover operation is implemented to produce a trial individual $\acute{\boldsymbol{X}}_{i,d}^{\eta}$ for individual i. It is shown that the population evolution usually traps into local optima, and it also might cause a premature problem. To solve it, this chapter adopts the Metropolis acceptance criterion of SA [59] to improve the diversity of individuals in the population evolution and avoid its too early convergence.

SA is able to escape from local optima by conditionally accepting some worse individuals, and finally converge to global optima with high probability. Let $\breve{M}$ denote the number of objective functions, and Λ_i^g the difference between $\mathbf{x}_i^g$ and $\grave{\mathbf{x}}_i^g$. Then,

$$\Lambda_i^g = |\widetilde{F_1}(\mathbf{x}_i^g) - \widetilde{F_1}(\grave{\mathbf{x}}_i^g)| + |\widetilde{\Delta_\tau^{13}}(\mathbf{x}_i^g) - \widetilde{\Delta_\tau^{13}}(\grave{\mathbf{x}}_i^g)| \tag{7.29}$$

To improve the population diversity and avoid premature convergence, an SA-based crossover operation is adopted, *i.e.*,

$$\acute{\mathbf{x}}_{i,d}^g = \begin{cases} \mathbf{x}_{i,d}^g, & \text{if } exp^{-\frac{\Lambda_i^g}{\theta_2^g}} > w_{16} \\ \mathbf{x}_{i,d}^g, & \text{otherwise} \end{cases} \tag{7.30}$$

where w_{16} is a random number uniformly distributed in (0,1), and θ_2^g denotes the current temperature in iteration g ($g\in\{1,2,\ldots,\hat{g}\}$).

7.2.4 SA-based Selection

In each generation, after mutation and crossover operations, an SA-selection is implemented on each individual to check whether a newly produced trial individual

$\acute{\mathbf{x}}_i^g$ or $\mathbf{x}_i^g$ is selected as $\mathbf{x}_i^{g+1}$ in the next generation. Specifically, if $\acute{\mathbf{x}}_i^g$ dominates $\mathbf{x}_i^g$, $\acute{\mathbf{x}}_i^g$ is accepted. If $exp^{-\frac{\Lambda_i^g}{\theta_2^g}} > w_{16}$, $\acute{\mathbf{x}}_i^g$ is selected; otherwise, $\mathbf{x}_i^g$ is selected.

$$\mathbf{x}_i^{g+1} = \begin{cases} \acute{\mathbf{x}}_i^g, & \text{if } \acute{\mathbf{x}}_i^g \text{ dominates } \mathbf{x}_i^g \\ \acute{\mathbf{x}}_i^g, & \text{if } exp^{-\frac{\Lambda_i^g}{\theta_2^g}} > w_{16} \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases} \tag{7.31}$$

7.2.5 Adaptive Elitist Archive Update Mechanism

At the beginning of the evolution of the population, the individual size of $\breve{\Omega}$ is zero. Then, a current individual is added into $\breve{\Omega}$ directly. If the current individual is dominated by each elitist individual in $\breve{\Omega}$, it is rejected. Otherwise, if the current individual dominates an elitist individual in $\breve{\Omega}$, it is added into $\breve{\Omega}$ and this dominated elitist individual is removed from $\breve{\Omega}$. Besides, if a current individual and elitist individuals in $\breve{\Omega}$ do not have any dominance relations, it is added into $\breve{\Omega}$ and when the number of elitist individuals in $\breve{\Omega}$ reaches $|\breve{\Omega}|$, the following adaptive elitist archive update mechanism is adopted to update elitist individuals in $\breve{\Omega}$. This mechanism is adopted to keep well-diversified Pareto optimal individuals in each generation, and help the convergence of population. $\widetilde{f}_\iota$ denotes the ιth ($\iota \in \{1, 2, \ldots, \overset{o}{M}\}$) objective function. Let $\hat{\theta}_6^\iota$ and $\breve{\theta}_6^\iota$ denote the maximum and minimum values of the ιth objective function, *i.e.*,

$$\begin{cases} \breve{\theta}_6^\iota = \underset{i \in \{1,2,\ldots,|\breve{\Omega}|\}}{\text{Min}} \left(\widetilde{f}_\iota(\mathbf{x}_i^g) \right), \iota \in \{1, 2, \ldots, \overset{o}{M}\} \\ \hat{\theta}_6^\iota = \underset{i \in \{1,2,\ldots,|\breve{\Omega}|\}}{\text{Max}} \left(\widetilde{f}_\iota(\mathbf{x}_i^g) \right), \iota \in \{1, 2, \ldots, \overset{o}{M}\} \end{cases} \tag{7.32}$$

Let $\theta_{3,\iota}^D$ denote the range of objective function ι, *i.e.*, $\theta_{3,\iota}^D = \hat{\theta}_6^\iota - \breve{\theta}_6^\iota$. $\breve{\theta}_{4,\iota}^D$ and $\hat{\theta}_{4,\iota}^D$ denote lower and higher bounds of the grid region for objective function ι, *i.e.*,

$$\begin{cases} \breve{\theta}_{4,\iota}^D = \breve{\theta}_6^\iota - \frac{1}{2\theta_4} \theta_{3,\iota}^D \\ \hat{\theta}_{4,\iota}^D = \hat{\theta}_6^\iota + \frac{1}{2\theta_4} \theta_{3,\iota}^D \end{cases} \tag{7.33}$$

Algorithm 5 SBDE (Simulated-annealing-based Bi-objective Differential Evolution)

1: Perform the initialization with equation (7.23)
2: $g \leftarrow 1$
3: $\breve{\Omega} \leftarrow \emptyset$
4: **while** $g \leq \hat{g}$ **do**
5: **for** $i \leftarrow 1$ to $|\mathbb{X}|$ **do**
6: Perform the adaptive mutation with equation (7.26)
7: Perform the SA-based crossover with equation (7.30)
8: Perform the SA-based selection with equation (7.31)
9: **end for**
10: Change $\breve{\Omega}$ with the adaptive elitist archive update mechanism
11: $g \leftarrow g+1$
12: $\theta_2^g \leftarrow \theta_2^g * \theta_3$
13: **end while**
14: Select the knee solution:
$\overset{*}{\mathbf{x}} = \arg_{\mathbf{x}} \underset{\boldsymbol{x} \in \breve{\Omega}}{\text{Min}} \| [\frac{\widetilde{F_1}(\boldsymbol{x}) - \widetilde{F_1}^{min}}{\widetilde{F_1}^{max} - \widetilde{F_1}^{min}}, \frac{\widetilde{\Delta_\tau^{13}}(\boldsymbol{x}) - \widetilde{\Delta_\tau^{13}}^{min}}{\widetilde{\Delta_\tau^{13}}^{max} - \widetilde{\Delta_\tau^{13}}^{min}}] - [1, 0] \|_1$
15: Output $\overset{*}{\mathbf{x}}$ and its two objectives $[\widetilde{F_1}(\overset{*}{\mathbf{x}}), \widetilde{\Delta_\tau^{13}}(\overset{*}{\mathbf{x}})]$

The objective space specified by individuals in current $\breve{\Omega}$ is separated into θ_4^2 hypercubes of equal size, and θ_4 denotes the grid number in any dimension of the space. The number of elitist individuals in each hypercube is calculated, and the hypercube with the most individuals is recored as the most congested one. Then, when the number of elitist individuals in current $\breve{\Omega}$ reaches $|\breve{\Omega}|$, the following adaptive elitist archive update mechanism is invoked.

1) If an individual is within its current objective space, it is put into $\breve{\Omega}$ and added to its corresponding hypercube. Besides, an elitist individual is removed randomly from the most congested grid.

2) If an individual is outside the current objective space, it is also put into $\breve{\Omega}$. The new objective space is then determined, and hypercubes are separated again. The number of elitist individuals in each hypercube is also obtained again. Then, an elitist individual is removed randomly from the most congested grid.

The extreme elitist individuals are reserved and not removed from $\breve{\Omega}$. Then, the size of $\breve{\Omega}$ is fixed and the individual diversity of $\breve{\Omega}$ is increased. Algorithm 5 presents the pseudo codes of SBDE. Line 1 initializes the population with equation (7.23).

The EA, $\breve{\Omega}$, is initialized with $\emptyset$ in Line 3. In each generation g, Line 6 performs the adaptive mutation for each individual i with equation (7.26). Line 7 performs the

SA-based crossover for each individual i with equation (7.30). Then, Line 8 performs the SA-based selection for each individual i with equation (7.31). Line 10 changes $\breve{\Omega}$ with the adaptive elitist archive update mechanism. Let θ_3 denote the cooling rate of temperature. Line 12 decreases the temperature (θ_2^g) in generation g by θ_3. Line 14 selects the knee solution $\overset{*}{\mathbf{x}}$. Finally, Line 15 outputs the knee solution $\overset{*}{\mathbf{x}}$ and its two objectives $[\widetilde{F_1}(\overset{*}{\mathbf{x}}), \widetilde{\Delta_\tau^{13}}(\overset{*}{\mathbf{x}})]$. Here, $\widetilde{F_1}^{min}$ and $\widetilde{F_1}^{max}$ denote the minimum and maximum values of $\widetilde{F_1}(\boldsymbol{x})$ ($\boldsymbol{x}\in\breve{\Omega}$). $\widetilde{\Delta_\tau^{13}}^{min}$ and $\widetilde{\Delta_\tau^{13}}^{max}$ denote the minimum and maximum values of $\widetilde{\Delta_\tau^{13}}(\boldsymbol{x})$ ($\boldsymbol{x}\in\breve{\Omega}$).

7.3 Performance Evaluation

This section evaluates PQTS by using realistic data including task arriving rates of three applications sampled every 5 minutes in Google cluster[1], prices of power grid[2], bandwidth prices of ISPs, solar irridiance[3] and wind speed[4] for 24 hours on May 10, 2011 [24].

7.3.1 Parameter Setting

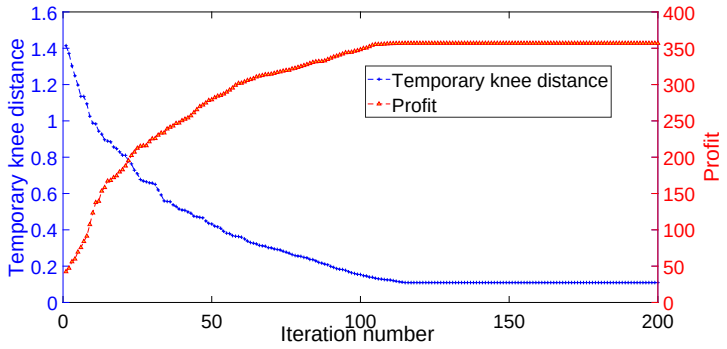


Figure 7.2 Convergence analysis of SBDE.

This chapter considers three applications, three ISPs and three CDCs, *i.e.* $\mathcal{N}^A$=3, $\mathcal{N}^K$=3 and $\mathcal{N}^C$=3. Their parameters including $\overset{\bullet}{\mathbb{N}}_{c,n}$, $\check{\Phi}_n^c$, $\hat{\Phi}_n^c$, $\hat{Q}_n^c$ and $\hat{N}_{c,n}$ are

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).
[2]http://www.nyiso.com/public/index.jsp (accessed on May 10, 2019).
[3]http://www.nrel.gov/midc/srrl_bms/ (accessed on May 10, 2019).
[4]http://www.nrel.gov/midc/nwtc_m2/ (accessed on May 10, 2019).

set according to Section 3.4 in Chapter 3. According to the work in [18], $\hat{B}_1$=4×10^6 (Mbps), $\hat{B}_2$=5×10^6 (Mbps) and $\hat{B}_3$=6×10^6 (Mbps). Besides, ζ_1=8 (Mb), ζ_2=5 (Mb), ζ_3=2 (Mb), $\hat{T}_1$=0.15 (seconds), $\hat{T}_2$=0.2 (seconds) and $\hat{T}_3$=0.25 (seconds). According to the work in [236], the parameters of SBDE are set as follows. $|\mathbb{X}|$=200, $\hat{g}$=200, $|\breve{\Omega}|$=100, θ_2^0=10^{12}, θ_3=0.975, θ_2^D=0.2, θ_4=80 and $\overset{\infty}{\mathcal{N}}$=10^{20}.

7.3.2 Experimental Results

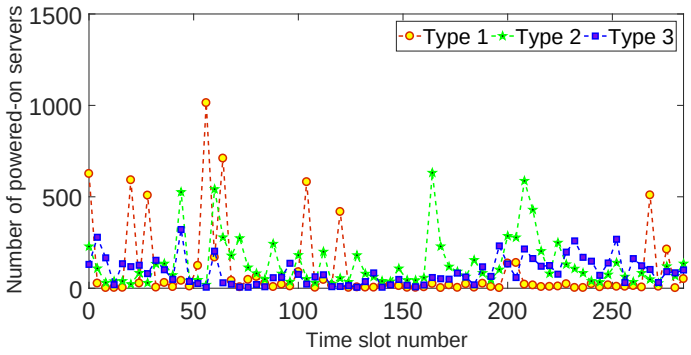


Figure 7.3 Number of switched-on servers in CDC 1.

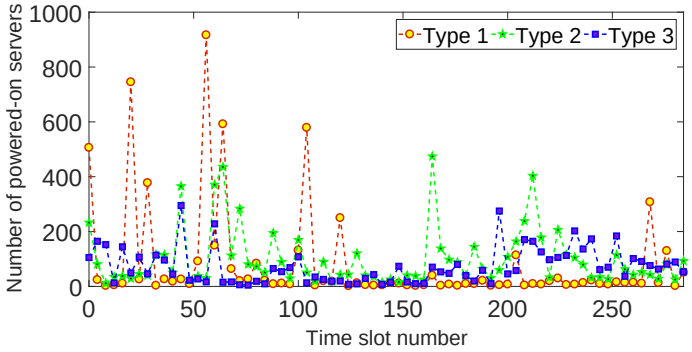


Figure 7.4 Number of switched-on servers in CDC 2.

Figure 7.2 illustrates the convergence analysis in terms of profit and temporary knee distance of SBDE in time slot 100. Here, the temporary knee distance in each iteration is calculated as follows. In Algorithm 5, each iteration runs Lines 5–12. $\breve{\Omega}$ is updated with an adaptive elitist archive update mechanism. Therefore, in each iteration, a temporary knee solution with the minimum knee distance is selected according to the rule of knee solution selection in Line 14. The temporary knee

Table 7.1 Convergence Time with Different Scales of Tasks

Scale	$\frac{1}{4}$	$\frac{1}{2}$	1	2	4
Convergence time (s)	2.21	3.99	6.77	13.13	25.08

distance in each iteration is obtained accordingly, and shown in Figure 7.2. It is observed that it reduces as the number of iterations increases. Besides, it is also shown that the profit increases with the evolution of SBDE. Figure 7.2 illustrates that the stable values are approached and demonstrates the convergence of SBDE.

Table 7.1 shows the convergence time with different scales of tasks. For example, if scale is 2, the task arriving rate of application n in time slot τ is doubled and it becomes $2\lambda_\tau^n$. Similarly, if scale is $\frac{1}{2}$, it becomes $\frac{1}{2}\lambda_\tau^n$. Thus, it is shown that the convergence time increases with the increase of the scales of tasks. Note that similar to the work in [76], the length of each time slot is 5 minutes, and tasks are scheduled based on the knee solution $\overset{*}{\mathbf{x}}$ obtained by SBDE from one time slot to another. Thus, compared to length of each time slot, the convergence time is negligible and does not affect the scheduling of tasks.

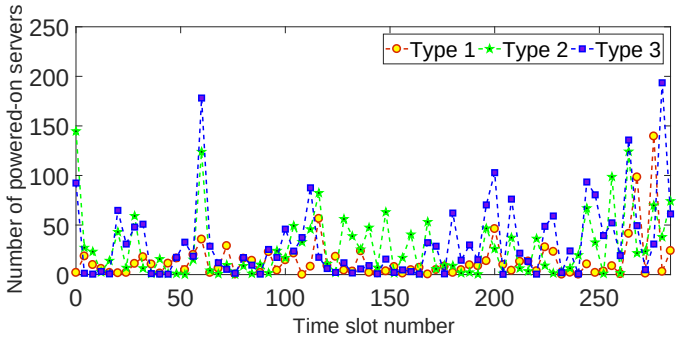


Figure 7.5 Number of switched-on servers in CDC 3.

Figures 7.3–7.5 illustrate the number of switched-on servers in three CDCs, respectively. It is shown that the number of switched-on servers for each application in three CDCs is less than its corresponding limit. Besides, it is observed that the number of switched-on servers for each application in three CDCs varies. For example, the number of switched-on servers for application 2 in CDC 3, is much smaller than

those in CDCs 1 and 2. The reason is that prices of power grid at CDCs 1 and 2 are less than that at CDC 3 in each time slot, and more tasks are therefore scheduled to CDCs 1 and 2. Accordingly, this yields a lower number of switched-on servers in CDC 3.

7.3.3 Comparison Results

To demonstrate the performance of SBDE, this chapter first compares it with several state-of-the-art multiobjective evolutionary algorithms including the improved fast and elitist Non-dominated Sorting Genetic Algorithm (NSGA2) [239], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [240] and MODE [236]. It is worth noting that each algorithm obtains its own Pareto front, which consists of a set of different solutions. Then, the aforementioned minimum Manhattan distance method is adopted to select a knee solution from the Pareto front of each algorithm. Four knee solutions are further compared with each other in terms of profit and average task loss possibility.

Table 7.2 Comparison Among Optimization Algorithms

Algorithms	Knee solution	
	Profit ($)	Average task loss possibility
NSGA2 [239]	3316.2	0.004
SPEA2 [240]	3194.9	0.010
MODE [236]	2830.9	0.022
SBDE	3437.5	0.0025

Table 7.2 shows that SBDE performs better than its three peers in terms of both profit and average task loss possibility. Compared with SBDE, NSGA2 has 3.53% reduction in profit and 37.50% increase in average task loss possibility; SPEA2 has 7.06% reduction in profit and 75.00% increase in average task loss possibility; and MODE has 17.65% reduction in profit and 88.64% increase in average task loss possibility. Furthermore, the convergence time of the Pareto optimal front obtained by SBDE is 6.77 seconds, which is negligible because the length of each time slot

is 5 minutes. The results are explained as follows. NSGA2, SPEA2 and MODE all converge to worse Pareto fronts with low diversity of populations. Different from NSGA2, SPEA2 and MODE, SBDE adopts adaptive mutation, SA-based crossover, SA-based selection, adaptive elitist archive update mechanism and the minimum Manhattan distance to improve the diversity of solutions, convergence speed and accuracy. Thus, the Pareto optimal front obtained by SBDE is distributed more evenly and diversely than other three algorithms. This means that SBDE provides more comprehensive candidate solutions. Consequently, the profit of SBDE is the largest while its average task loss possibility is the lowest among four algorithms.

To demonstrate the performance of PQTS, it is further compared with three typical task scheduling approaches [241, 242] in terms of the profit and the average task loss possibility.

1) M1 [241] is electricity-cost-aware distributed task scheduling.
2) M2 [87] is profit maximization approach (PMA).
3) M3 [242] is green task balancing (GTB).

Figures 7.6 and 7.7 show the comparison of profit and average task loss possibility of PQTS and M1–M3. The average task loss possibility of PQTS is the lowest and its profit is the highest compared with M1–M3. Its profit is 8.40% higher and its average task loss possibility is 93.47% lower than M1's; 11.72% higher and its average task loss possibility is 93.50% lower than M2's; and 20.66% higher and its average task loss possibility is 94.75% lower than M3's. The reasons are explained as follows. The profit of M3 is the least because it aims to achieve the energy cost minimization rather than the profit maximization. In M3, the number of switched-on servers is minimized provided that it is sufficient to execute all tasks. Thus, M3 achieves larger average task loss possibility and smaller profit. Similarly, M1 aims to minimize the energy cost of distributed data centers with intelligent load balancing. M2 aims to maximize the profit of a green data center with convex optimization for

two cases, with and without behind-the-meter green generators. However, they both do not consider the bandwidth cost and capacity limits of ISPs, and the availability of wind and solar energy in CDCs. Consequently, the profits of M1 and M2 are lower than those of PQTS. M1 and M2 both model QoS as a constraint in its optimization problem, and therefore, the minimum QoS satisfaction is achieved. Different from them, PQTS jointly maximizes the profit of CDC providers, and minimizes the average task loss possibility of all applications. Therefore, it realizes higher profit and better QoS for CDC providers than M1–M3.

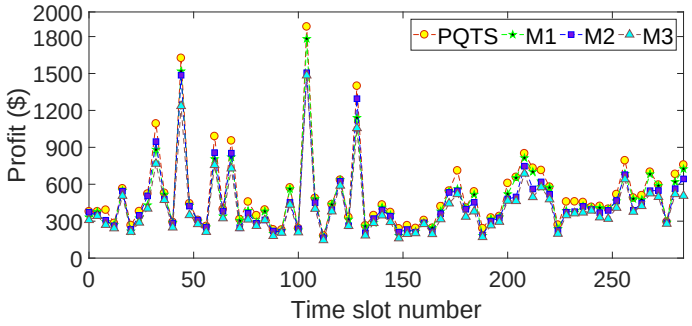


Figure 7.6 Profit comparison of PQTS and M1–M3.

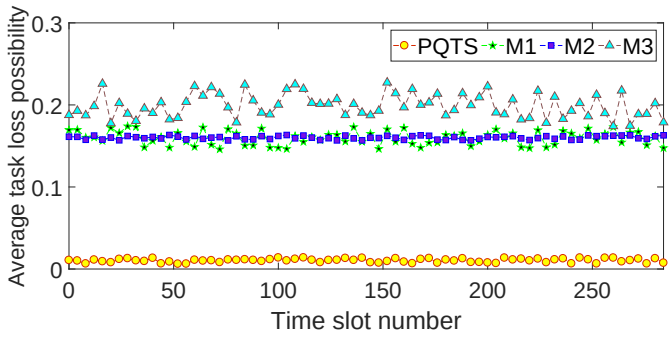


Figure 7.7 Average task loss possibility comparison of PQTS and M1–M3.

7.4 Summary

The number of tasks of global users significantly increases the energy cost of cloud data centers (CDCs). Meanwhile, the dramatic growth of tasks also increases the revenue of CDC providers, which is closely related to tasks' quality of service (QoS). Besides, many factors in different CDCs, *e.g.*, prices of power grid, wind and solar

energy, exhibit the spatial variations. Consequently, it becomes a big challenge to jointly maximize the profit of CDC providers, and minimize the average task loss possibility of all applications. This chapter proposes a Profit and QoS-optimized Task Scheduling (PQTS) method to achieve a beneficial tradeoff between these two objectives for both users and CDCs. Specifically, this chapter formulates a bi-objective optimization problem and solves it with a simulated-annealing-based bi-objective differential evolution algorithm. The minimum Manhattan distance is used to specify a knee solution that determines proper task split among ISPs and task service rates at CDCs in each time slot. Real-life data-based simulations reveal that the proposed method achieves significantly higher profit and lower average task loss possibility of all applications than three existing task scheduling algorithms.

CHAPTER 8

ENERGY-CONSUMPTION AND PERFORMANCE-OPTIMIZED TASK SCHEDULING IN DISTRIBUTED CLOUD DATA CENTERS

This chapter presents the details of the energy-consumption and performance-optimized task scheduling method, and it is organized as follows. Section 8.1 gives the formulation of a bi-objective constrained optimization problem. Section 8.2 gives the proposed Simulated-annealing-based Adaptive Differential Evolution (SADE). According to SADE, a close-to-optimal task scheduling strategy is given to trade off the task response time of all tasks in CDCs, and the energy cost of a CDC provider. Section 8.3 evaluates the proposed method by using realistic data. Section 8.5 concludes this chapter.

8.1 Problem Formulation

This section gives the formulation of a bi-objective constrained optimization problem. Figure 8.1 shows the architecture of CDCs. It is assumed that there are $\mathcal{N}^C$ back-end data centers in CDCs. Users send their tasks through front-end pervasive devices, *e.g.*, laptops, smart phones and computers, to CDCs [243]. Users' arriving tasks are scheduled in a First-Come-First-Serve (FCFS) way. *Task Scheduler* executes SADE periodically and schedules tasks in a queue to jointly minimize both the energy cost and the task response time of all tasks by intelligently executing all arriving tasks among CDCs.

8.1.1 Task Response Time Model

This section first gives the model of the task response time of all tasks in CDCs. This section considers $\mathcal{N}^C$ heterogeneous CDCs to simultaneously execute tasks. In CDC c ($1 \leq c \leq \mathcal{N}^C$), there are a group of $\hat{N}_c$ heterogeneous servers in total. Here, this chapter adopts a $G/G/1$ queuing model that is the most general queuing model to analyze the

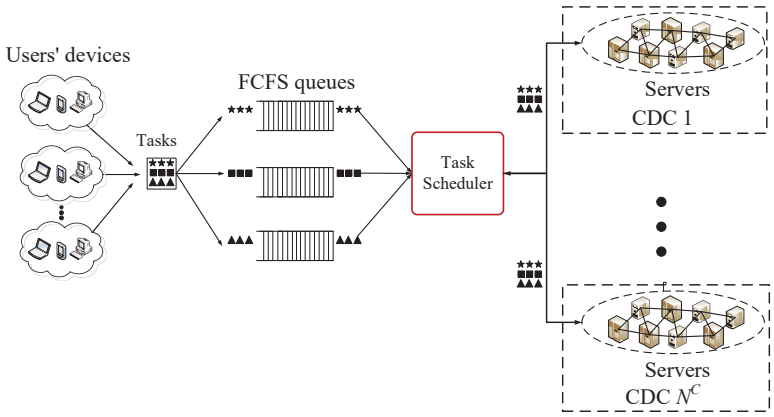


Figure 8.1 Architecture of CDCs.

performance of each arbitrary server in a CDC. Specifically, each switched-on server is analyzed as a $G/G/1$ queuing system.

Let $\overset{o}{N}_{\tau,c}$ denote the number of the switched-on servers in CDC c in time slot τ. It is assumed in this chapter that the task execution and interarrival times may have arbitrary distributions of probability in server q ($1 \leq q \leq \overset{o}{N}_{\tau,c}$). $\overset{o}{N}_{\tau,c}$ must be less than or equal to $\hat{N}_c$, *i.e.*,

$$0 \leq \overset{o}{N}_{\tau,c} \leq \hat{N}_c, \ \ \overset{o}{N}_{\tau,c} \in N^+ \tag{8.1}$$

Let $\hat{\sigma}_{\tau,c,q}$ denote the interarrival time for server q in CDC c. $\hat{\sigma}_{\tau,c,q}$ can be any random variable with the variance $\hat{\sigma}_{\tau,c,q}$ and the mean $\overline{\mathbb{T}}_{\tau}^{c,q}$, which are obtained from analyzing tasks in the realistic Google cluster trace. Let $\tilde{\lambda}_{\tau}^{c}$ denote the task arriving rate of CDC c in time slot τ. Thus, the task arriving rate at server q in CDC c in time slot τ is obtained as $\lambda_{\tau}^{c,q} = \frac{1}{\hat{\sigma}_{\tau,c,q}}$, which is the number of tasks per second in time slot τ. Let $\mathbb{r}_{c}^{q}$ denote the random execution requirement of each task submitted to server q in CDC c. $\mathbb{r}_{c}^{q}$ can have any arbitrary probability distribution with the variance $\breve{\sigma}_{c,q}$ and the mean $\bar{r}_{c}^{q}$. Let $\varrho_{\tau}^{c,q}$ denote the running speed of server q in CDC c in time slot τ. Let $\hat{\varrho}_c$ denote the maximum running speed of each server in CDC c. Thus, $\varrho_{\tau}^{c,q}$ must be less than or equal to $\hat{\varrho}_c$, *i.e.*,

$$0 \leq \varrho_{\tau}^{c,q} \leq \hat{\varrho}_c \tag{8.2}$$

In addition, the running speed of server q in CDC c in time slot τ must be large enough to process its arriving tasks. Specifically, $\varrho_{\tau}^{c,q}$ must be larger than or equal to $\frac{\bar{r}_c^q}{\bar{\mathbb{T}}_{\tau}^{c,q}}$, *i.e.*,

$$\varrho_{\tau}^{c,q} > \frac{\bar{r}_c^q}{\bar{\mathbb{T}}_{\tau}^{c,q}} \tag{8.3}$$

The running time of each task on server q in CDC c in time slot τ is $\mathbb{\Omega}_{\tau}^{c,q} = \mathbb{r}_c^q / \varrho_{\tau}^{c,q}$, which is measured in second. Its mean, variance and variation coefficient are $\bar{\mathbb{t}}_{\tau}^{c,q}$, $\tilde{\sigma}_{\tau,c,q}$ and $\mathbb{\Omega}_{\tau}^{c,q}$, respectively.

$$\bar{\mathbb{t}}_{\tau}^{c,q} = \frac{\bar{r}_c^q}{\varrho_{\tau}^{c,q}} \tag{8.4}$$

$$\tilde{\sigma}_{\tau,c,q} = \frac{\breve{\sigma}_{c,q}}{\left(\varrho_{\tau}^{c,q}\right)^2} \tag{8.5}$$

$$\mathbb{\Omega}_{\tau}^{c,q} = \frac{\tilde{\sigma}_{\tau,c,q}}{\bar{\mathbb{t}}_{\tau}^{c,q}} \tag{8.6}$$

Let λ_{τ} denote the total task arriving rate in time slot τ. $\tilde{\lambda}_{\tau}^{c}$ denotes the task arriving rate of CDC c in time slot τ. Then, λ_{τ} is the sum of task arriving rates of all CDCs in τ, *i.e.*,

$$\lambda_{\tau} = \sum_{c=1}^{\mathcal{N}^C} \tilde{\lambda}_{\tau}^{c} \tag{8.7}$$

In this way, the average response time of tasks in CDC c with $\overset{o}{N}_{\tau,c}$ servers is obtained as follows.

$$T_{\tau}^{c} = \frac{1}{\tilde{\lambda}_{\tau}^{c}} \sum_{i=1}^{\overset{o}{N}_{\tau,c}} \lambda_{\tau}^{c,q} \left(\frac{\bar{r}_c^q}{\varrho_{\tau}^{c,q}} + \left((\bar{r}_c^q)^2 + \breve{\sigma}_{c,q} \right) \right) \Delta_{\tau,c,q}^{14} \tag{8.8}$$

$$\Delta_{\tau,c,q}^{14} = \frac{\hat{\sigma}_{\tau,c,q} (\varrho_{\tau}^{c,q})^2 + \breve{\sigma}_{c,q}}{2\varrho_{\tau}^{c,q} (\bar{\mathbb{T}}_{\tau}^{c,q} \varrho_{\tau}^{c,q} - \bar{r}_c^q)((\bar{\mathbb{T}}_{\tau}^{c,q})^2 (\varrho_{\tau}^{c,q})^2 + \breve{\sigma}_{c,q})}$$

Then, this chapter adopts the maximum value of the average response time of tasks in multiple CDCs as the task response time of all tasks in CDCs, which is

denoted by T_τ. Therefore, T_τ is obtained as follows.

$$T_\tau = \underset{c}{\mathbf{Max}} \ \{T_\tau^c\} \tag{8.9}$$

For clarity, the derivation of (8.8) is given in the Appendix.

8.1.2 Energy Cost Model

Then, this chapter further describes the model of energy cost of CDCs. It is shown that in a well-designed circuit, the dominant component in CDCs is dynamic power consumption (measured in Watt). Let $\mathbb{P}_\tau^{c,q}$ denote the amount of power consumption of each server q in CDC c in time slot τ. $\mathbb{U}_\tau^{c,q}$ denotes the supply voltage of each server q in CDC c and $\chi_\tau^{c,q}$ denotes the clock frequency of each server q in CDC c. In the perfect case, the clock frequency and the supply voltage are related to each other in such a way that $\mathbb{U}_\tau^{c,q} \propto (\chi_\tau^{c,q})^{\overset{0}{\gamma}_{3,c}}$ for some constant $\overset{0}{\gamma}_{3,c} > 0$ [244]. The processor running speed $\varrho_\tau^{c,q}$ is linearly proportional to $\chi_\tau^{c,q}$, *i.e.*, $\varrho_\tau^{c,q} \propto \chi_\tau^{c,q}$. Therefore, similar to the work in [36], $\mathbb{U}_\tau^{c,q} = \overset{0}{\gamma}_{4,c} (\chi_\tau^{c,q})^{\overset{0}{\gamma}_{3,c}}$ and $\varrho_\tau^{c,q} = \overset{0}{\gamma}_{5,c} \chi_\tau^{c,q}$. Here, $\overset{0}{\gamma}_{4,c}$ and $\overset{0}{\gamma}_{5,c}$ are two constants for CDC c. Therefore, the following equation for obtaining the amount of power consumption is given as follows.

$$\begin{aligned}
\mathbb{P}_\tau^{c,q} &= \vartheta_c \omega_c \left(\mathbb{U}_\tau^{c,q}\right)^2 \chi_\tau^{c,q} \\
&= \vartheta_c (\overset{0}{\gamma}_{4,c})^2 \omega_c \left(\chi_\tau^{c,q}\right)^{2\overset{0}{\gamma}_{3,c}+1} \\
&= \vartheta_c (\overset{0}{\gamma}_{4,c})^2 \omega_c \frac{\left(\varrho_\tau^{c,q}\right)^{2\overset{0}{\gamma}_{3,c}+1}}{\left(\overset{0}{\gamma}_{5,c}\right)^{2\overset{0}{\gamma}_{7,c,n}+1}} \\
&= \beth_c \left(\varrho_\tau^{c,q}\right)^{\Psi_c}
\end{aligned} \tag{8.10}$$

where ϑ_c denotes the activity factor in CDC c, ω_c denotes the loading capacitance in CDC c. Besides, $\beth_c = \vartheta_c (\overset{0}{\gamma}_{4,c})^2 \omega_c / (\overset{0}{\gamma}_{5,c})^{2\overset{0}{\gamma}_{7,c,n}+1}$ and $\Psi_c = 2\overset{0}{\gamma}_{7,c,n}+1$.

Similar to the work in [22, 113, 245], it is assumed that the servers in each CDC are homogeneous while those in different CDCs are heterogeneous. Each server

q with speed $\varrho_\tau^{c,q}$ consumes the power of $\mathbb{P}_\tau^{c,q}$. It is worth noting that each server has to consume a certain amount of base power when it is idle due to the dissipation of short-circuit and static power, and other wasted and leakage power [79]. Let $\check{\Phi}_q^c$ denote the amount of power consumed by each idle server q in CDC c. In addition, let L denote the length of each time slot. Let P_τ^c denote the total power consumption of CDC c in time slot τ. Similar to the work in [24, 246], P_τ^c is obtained as follows.

$$P_\tau^c = \sum_{q=1}^{\overset{o}{N}_{\tau,c}} \left(\mathbb{P}_\tau^{c,q} + \check{\Phi}_q^c \right) = \sum_{q=1}^{\overset{o}{N}_{\tau,c}} \left(\urcorner_c \left(\varrho_\tau^{c,q} \right)^{\Psi_c} + \check{\Phi}_q^c \right) \tag{8.11}$$

Then, the total energy consumption of CDC c in time slot τ is $P_\tau^c L$, and it cannot exceed $\hat{\mathbb{E}}_c$, which denotes the maximum available energy in CDC c, *i.e.*,

$$P_\tau^c L < \hat{\mathbb{E}}_c \tag{8.12}$$

Let p_τ^c denote the electricity price of CDC c in time slot τ. It is assumed that this chapter neglects the regulation influence of CDCs on electricity prices. Thus, the energy cost of CDC c is denoted by $f_{22}^{c,*}$, which is obtained as follows.

$$f_{22}^{c,*} = P_\tau^c L p_\tau^c = \sum_{q=1}^{\overset{o}{N}_{\tau,c}} \left(\urcorner_c \left(\varrho_\tau^{c,q} \right)^{\Psi_c} + \check{\Phi}_q^c \right) L p_\tau^c \tag{8.13}$$

Then, the energy cost of the CDC provider is denoted by f_{22}, which is obtained as follows.

$$f_{22} = \sum_{c=1}^{\mathcal{N}^C} f_{22}^{c,*} = \sum_{c=1}^{\mathcal{N}^C} P_\tau^c L p_\tau^c = \sum_{c=1}^{\mathcal{N}^C} \left(\sum_{q=1}^{\overset{o}{N}_{\tau,c}} \left(\urcorner_c \left(\varrho_\tau^{c,q} \right)^{\Psi_c} + \check{\Phi}_q^c \right) L p_\tau^c \right) \tag{8.14}$$

8.1.3 Bi-objective Constrained Optimization Problem

The first objective is to minimize T_τ, *i.e.*,

$$\underset{\tilde{\lambda}_\tau^c, \overset{o}{N}_{\tau,c}, \varrho_\tau^{c,q}}{\mathbf{Min}} \quad \{T_\tau\} \tag{8.15}$$

The second objective is to minimize f_{22}, *i.e.*,

$$\underset{\tilde{\lambda}_{\tau}^{c}, \overset{o}{N}_{\tau,c}, \varrho_{\tau}^{c,q}}{\textbf{Min}} \quad \{f_{22}\} \tag{8.16}$$

Then, according to constraints (8.1), (8.2), (8.3), (8.7) and (8.12), this chapter formulates a bi-objective optimization problem as:

$$\underset{\tilde{\lambda}_{\tau}^{c}, \overset{o}{N}_{\tau,c}, \varrho_{\tau}^{c,q}}{\textbf{Min}} \quad \{T_{\tau}\}$$

$$\underset{\tilde{\lambda}_{\tau}^{c}, \overset{o}{N}_{\tau,c}, \varrho_{\tau}^{c,q}}{\textbf{Min}} \quad \{f_{22}\}$$

subject to

$$\overset{o}{N}_{\tau,c} \leq \hat{N}_{c} \tag{8.17}$$

$$\varrho_{\tau}^{c,q} \leq \hat{\varrho}_{c} \tag{8.18}$$

$$\lambda_{\tau} = \sum_{c=1}^{\mathcal{N}^{C}} \tilde{\lambda}_{\tau}^{c} \tag{8.19}$$

$$P_{\tau}^{c} L < \hat{\mathbb{E}}_{c} \tag{8.20}$$

$$\varrho_{\tau}^{c,q} > \frac{\bar{r}_{c}^{q}}{\bar{\mathbb{T}}_{\tau}^{c,q}} \tag{8.21}$$

$$\overset{o}{N}_{\tau,c} \geq 0, \varrho_{\tau}^{c,q} \geq 0, \tilde{\lambda}_{\tau}^{c} \geq 0 \tag{8.22}$$

$$\tilde{\lambda}_{\tau}^{c} = 0, \text{ if } \overset{o}{N}_{\tau,c} = 0 \tag{8.23}$$

$$\tilde{\lambda}_{\tau}^{c} > 0, \text{ if } \overset{o}{N}_{\tau,c} > 0 \tag{8.24}$$

Constraint (8.22) specifies ranges of decision variables including $\tilde{\lambda}_{\tau}^{c}$, $\overset{o}{N}_{\tau,c}$ and $\varrho_{\tau}^{c,q}$. In addition, constraints (8.23) and (8.24) show that if $\overset{o}{N}_{\tau,c}=0$, $\tilde{\lambda}_{\tau}^{c}$ must be 0 because there are no corresponding switched-on servers in CDC c to execute tasks in time slot τ; otherwise, if $\overset{o}{N}_{\tau,c}>0$, $\tilde{\lambda}_{\tau}^{c}$ must be greater than 0 because there are available corresponding switched-on servers in CDC c to execute tasks in time slot τ. Then, the algorithm proposed to solve this bi-objective constrained optimization problem is described next.

8.2 Simulated-annealing-based Adaptive Differential Evolution

It is worth noting that T_τ and f_{22} in a bi-objective constrained optimization problem are nonlinear with respect to $\tilde{\lambda}_\tau^c$, $\overset{o}{N}_{\tau,c}$ and $\varrho_\tau^{c,q}$. Thus, it is a constrained bi-objective nonlinear optimization problem. To well solve it, this chapter adopts a penalty function approach to convert it into an unconstrained optimization problem, *i.e.*,

$$\begin{aligned} &\underset{\tilde{\lambda}_\tau^c, \overset{o}{N}_{\tau,c}, \varrho_\tau^{c,q}}{\textbf{Min}} \left\{ \widetilde{T}_\tau = \overset{\infty}{\mathcal{N}} \mho + T_\tau \right\} \\ &\underset{\tilde{\lambda}_\tau^c, \overset{o}{N}_{\tau,c}, \varrho_\tau^{c,q}}{\textbf{Min}} \left\{ \widetilde{f_{22}} = \overset{\infty}{\mathcal{N}} \mho + f_{22} \right\} \end{aligned} \tag{8.25}$$

where $\widetilde{T}_\tau$ and $\widetilde{f_{22}}$ are two augmented objective functions, $\overset{\infty}{\mathcal{N}}$ is a positive constant, and $\mho$ is the penalty of all constraints. $\boldsymbol{x}$ is a decision variable vector including $\tilde{\lambda}_\tau^c$, $\overset{o}{N}_{\tau,c}$ and $\varrho_\tau^{c,q}$. $\mho$ is obtained with equation (3.21) in Chapter 3.

There are several traditional approaches to transform two objectives into one objective with linear weights, *e.g.*, weighted sum approach, to solve it. However, such approach has difficulty in determining weights for two objectives. In addition, it only gives a single candidate solution and requires multiple executions to produce Pareto-optimal candidates. It is more beneficial to give them in a single execution. Thus, several multi-objective evolutionary algorithms, *e.g.*, Strength Pareto Evolutionary Algorithm (SPEA) [247], ϵ-constraint approach [248], multi-objective particle swarm optimization (MOPSO) [249] and the non-dominated sorting genetic algorithm 2 (NSGA2) [250], are available to jointly optimize multiple objectives and give Pareto-optimal candidates for decision makers. Each algorithm owns its advantages and disadvantages. For example, in the ϵ-constraint approach, the ϵ vector is difficult to set as it needs to be in valid ranges of objective functions though it is applicable to different optimization problems. In addition, the mutation and crossover operations in NSGA2 are aimless and random, and can have difficulties in producing high-quality offspring.

Among them, differential evolution (DE) [251, 252] is a typical population-based algorithm for multi-objective optimization problems. Compared to other similar algorithms, DE's implementation is much easier and its convergence process is faster [253]. Thus, multi-objective DE (MODE) is widely applied to solve multi-objective optimization problems. MODE only has a few parameters and efficient population update. Therefore, it is widely used in many areas, *e.g.*, facial expression recognition and power flow optimization [254]. Nevertheless, MODE also suffers from a premature convergence problem.

This chapter proposes a Simulated-annealing-based Adaptive Differential Evolution (SADE) algorithm to solve the unconstrained optimization problem. SADE includes SA-based crossover and selection, and adaptive mutation to increase the convergence accuracy and speed. Besides, an entropy-based crowding distance method is adopted to produce a high-diversity Pareto-optimal front. Then, a final solution called knee is chosen from the front with the minimum Manhattan distance, and it represents the best tradeoff between two objectives.

8.2.1 Individual Encoding

Each individual i includes $\tilde{\lambda}_{\tau}^{c}$, $\overset{o}{N}_{\tau,c}$ and $\varrho_{\tau}^{c,q}$, and is encoded as:

$$\mathbf{x}_i = \left[\tilde{\lambda}_{\tau}^{1}, \cdots, \tilde{\lambda}_{\tau}^{\mathcal{N}^C}, \overset{o}{N}_{\tau,1}, \cdots, \overset{o}{N}_{\tau,\mathcal{N}^C}, \varrho_{\tau}^{1,q}, \cdots, \varrho_{\tau}^{\mathcal{N}^C,q}\right] \tag{8.26}$$

8.2.2 Population Initialization

Let $|\mathbb{X}|$ denote the population size. Let $\overset{0}{\mathbf{x}}_{i,d}$ denote the initial decision variable d of individual $i \in \{1, 2, \ldots, |\mathbb{X}|\}$, *i.e.*,

$$\overset{0}{\mathbf{x}}_{i,d} = \check{\boldsymbol{\theta}}_5^d + w_{19,i} * \left(\hat{\boldsymbol{\theta}}_5^d - \check{\boldsymbol{\theta}}_5^d\right) \tag{8.27}$$

where $\check{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\theta}}_5^d$ are lower and upper limits of decision variable d, and $w_{19,i}$ is a number for individual i randomly obtained in (0,1).

8.2.3 Adaptive Mutation

The number of generations is denoted by $\hat{g}$. Here, the best individual is denoted by $\overset{\bullet}{i}$. A new individual $\grave{\mathbf{x}}_i^g$ is produced for $\mathbf{x}_i^g$ as follows.

$$\grave{\mathbf{x}}_i^g = \overset{\bullet}{\mathbf{x}}_g + exp^{-0.2\frac{g}{\hat{g}}}\left(\mathbf{x}_{w_{20}}^g - \mathbf{x}_{w_{21}}^g\right), \overset{\bullet}{i} \neq w_{20} \neq w_{21} \neq i \tag{8.28}$$

where $\overset{\bullet}{\mathbf{x}}_g$ ($\overset{\bullet}{i} \neq w_{20} \neq w_{21} \neq i$) is the best individual in generation $g \in \{1, 2, \ldots, \hat{g}\}$, and $\mathbf{x}_{w_{20}}^g$ and $\mathbf{x}_{w_{21}}^g$ are two random individuals. It is worth noting that w_{20} is different from w_{21}, and both of them are different from $\overset{\bullet}{i}$.

Furthermore, the adaptive mutation is designed as follows.

$$\grave{\mathbf{x}}_i^g = \begin{cases} \mathbf{x}_i^g + exp^{-0.2\frac{g}{\hat{g}}}\left(\mathbf{x}_{w_{20}}^g - \mathbf{x}_{w_{21}}^g\right), & \text{if } \mathbf{x}_i^g \in \breve{\Omega} \\ \overset{\bullet}{\mathbf{x}}_g + exp^{-0.2\frac{g}{\hat{g}}}\left(\mathbf{x}_{w_{20}}^g - \mathbf{x}_{w_{21}}^g\right), & \text{otherwise} \end{cases} \tag{8.29}$$

$$\overset{\bullet}{i} = \arg_j \underset{j \in \{1,2,\ldots,|\breve{\Omega}|\}}{\text{Min}} \left(\overset{\longleftrightarrow}{i, j}\right) \tag{8.30}$$

$$\overset{\longleftrightarrow}{i, j} = \sqrt{\sum_{d=1}^{\mathbb{N}^D} \left(\mathbf{x}_{i,d}^g - \mathbf{x}_{j,d}^g\right)^2}, j \in \{1, 2, \ldots, |\breve{\Omega}|\} \tag{8.31}$$

where $\breve{\Omega}$ denotes an external archive (EA) to store the Pareto-optimal candidates, $|\breve{\Omega}|$ denotes the number of individuals in $\breve{\Omega}$, $\mathbb{N}^D$ denotes the number of decision variables, and $\overset{\longleftrightarrow}{i, j}$ denotes the distance between i and $j \in \{1, 2, \ldots, |\breve{\Omega}|\}$.

8.2.4 SA-based Crossover

This chapter uses the Metropolis acceptance rule of SA [255] to find global optima by conditionally choosing worse candidates. The SA-based crossover is conducted as:

$$\acute{\mathbf{x}}_{i,d}^g = \begin{cases} \mathbf{x}_{i,d}^g, & \text{if } exp^{-\frac{\Lambda_i^g}{\theta_2^g}} > w_{22} \\ \mathbf{x}_{i,d}^g, & \text{otherwise} \end{cases} \tag{8.32}$$

$$\Lambda_i^g = \sum_{\iota=1}^{\overset{o}{M}} |\widetilde{f}_\iota(\mathbf{x}_i^g) - \widetilde{f}_\iota(\grave{\mathbf{x}}_i^g)| \tag{8.33}$$

where w_{22} denotes a random number in (0,1), θ_2^g is the temperature in iteration g, $\acute{\mathbf{x}}_i^g$ denotes a new candidate for i, $\overset{o}{M}$ denotes the number of objective functions and Λ_i^g is the difference between $\grave{\mathbf{x}}_i^g$ and $\mathbf{x}_i^g$.

8.2.5 SA-based Selection

Then, SA-selection is conducted on each candidate to update $\mathbf{x}_i^{g+1}$, *i.e.*,

$$\mathbf{x}_i^{g+1} = \begin{cases} \acute{\mathbf{x}}_i^g, & \text{if } \acute{\mathbf{x}}_i^g \text{ dominates } \mathbf{x}_i^g \\ \acute{\mathbf{x}}_i^g, & \text{if } exp^{-\frac{\Lambda_i^g}{\theta_2^g}} > w_{22} \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases} \tag{8.34}$$

8.2.6 Entropy-based Crowding Distance Method

To keep elite candidates obtained by SADE, EA is adopted to keep them for the convergence purpose. When the number of elite candidates in EA exceeds its limit, this chapter adopts an entropy-based crowding distance method [256] to realize the truncation of EA for obtaining an evenly distributed Pareto front. The entropy is a concept to properly evaluate the diversity distribution characteristics of the Pareto front. Here, this chapter adopts the lower adjacent distance to calculate the entropy of distribution, which is obtained as follows:

$$\theta_{1,i}^A = -\theta_{2,i}^A log(\theta_{2,i}^A) \tag{8.35}$$

$$\theta_{2,i}^A = \frac{\theta_{3,i}^A}{\sum_{i=1}^{|\breve{\Omega}|-1} \theta_{3,i}^A} \tag{8.36}$$

$$\theta_{3,i}^A = \sum_{\iota=1}^{\overset{o}{M}} |\widetilde{f}_\iota(\mathbf{x}_i) - \widetilde{f}_\iota(\theta_{4,i,\iota}^A)| \tag{8.37}$$

where $\theta_{1,i}^A$ denotes the entropy of candidate i in EA, $\theta_{2,i}^A$ denotes the sparsity degree of candidate i in EA, $\theta_{3,i}^A$ denotes the distance of candidate i to its lower adjacent

candidate in EA, $\widetilde{f_\iota}$ denotes the ιth ($\iota \in \{1, 2, \ldots, \overset{o}{M}\}$) augmented objective function. and $\theta^A_{4,i,\iota}$ denotes the lower adjacent candidate of candidate i in EA along objective ι.

Then, let θ^A_5 ($\theta^A_5 \in [0,1]$) denote the diversity of the Pareto-optimal front, and it is calculated by summing up the entropy of all candidates in EA, *i.e.*,

$$\theta^A_5 = \frac{1}{log_2(|\breve{\Omega}|-1)} \sum_{i=1}^{|\breve{\Omega}|-1} E_i \tag{8.38}$$

Note that the Pareto-optimal front is evenly distributed if the entropy of each candidate in EA is 1. Therefore, the metric of diversity can keep elite candidates in EA. If the number of elitist candidates in $\breve{\Omega}$ is $|\breve{\Omega}|$ and a new candidate is not dominated by any candidate in EA, it is first put into $\breve{\Omega}$ and the current size of EA is $|\breve{\Omega}|+1$. Then, the entropy of each candidate in EA is calculated again. The candidate with the smallest entropy is removed from EA, and there are still $|\breve{\Omega}|$ elitist candidates in EA. Algorithm 6 shows details of SADE. Line 1 conducts initialization with (8.27). Line 3 initializes $\breve{\Omega}$ with $\emptyset$. Line 6 conducts adaptive mutation with (8.29). Line 7 conducts SA-based crossover and selection with (8.32) and (8.34). Line 9 conducts the entropy-based crowding distance method. Line 11 reduces θ^g_2 by the temperature cooling rate denoted by θ_3. Lines 13 and 14 determine and output knee $\overset{*}{\mathbf{x}}$. Here, $\widetilde{T_\tau}^{min}$ and $\widetilde{T_\tau}^{max}$ denote the minimum and maximum values of $\widetilde{T_\tau}(\boldsymbol{x})$ ($\boldsymbol{x} \in \breve{\Omega}$). $\widetilde{f_{22}}^{min}$ and $\widetilde{f_{22}}^{max}$ denote the minimum and maximum values of $\widetilde{f_{22}}(\boldsymbol{x})$ ($\boldsymbol{x} \in \breve{\Omega}$).

8.3 Performance Evaluation

This chapter uses real-life arriving tasks in Google cluster trace[1] to evaluate SADE. Figure 8.2 illustrates arriving rates of tasks of three applications. Figure 8.3 illustrates real-life electricity prices in three CDCs[2].

The length of each time slot is 300 seconds, *i.e.*, L= 300 seconds. According to the work in [24], the parameter setting for three CDCs is given in Table 8.1. Figure

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).
[2]http://www.energyonline.com/Data/ (accessed on May 6, 2019).

Algorithm 6 SADE (Simulated-annealing-based Adaptive Differential Evolution)

Conduct initialization with (8.27)
$g \leftarrow 1$
$\breve{\Omega} \leftarrow \emptyset$
while $g \leq \hat{g}$ **do**
  for $i \leftarrow 1$ to $|\mathbb{X}|$ **do**
    Conduct adaptive mutation with (8.29)
    Conduct SA-based crossover and selection with (8.32) and (8.34)
  end for
  Conduct the entropy-based crowding distance method
  $g \leftarrow g+1$
  $\theta_2^g \leftarrow \theta_2^g * \theta_3$
end while
Determine knee:
$\overset{*}{\mathbf{x}} = \arg_{\boldsymbol{x}} \underset{\boldsymbol{x} \in \breve{\Omega}}{\text{Min}} \| [\frac{\widetilde{T_\tau}(\boldsymbol{x}) - \widetilde{T_\tau}^{min}}{\widetilde{T_\tau}^{max} - \widetilde{T_\tau}^{min}}, \frac{\widetilde{f_{22}}(\boldsymbol{x}) - \widetilde{f_{22}}^{min}}{\widetilde{f_{22}}^{max} - \widetilde{f_{22}}^{min}}] - [1, 0] \|_1$
Output $\overset{*}{\mathbf{x}}$ and its two objectives $[\widetilde{T_\tau}(\overset{*}{\mathbf{x}}), \widetilde{f_{22}}(\overset{*}{\mathbf{x}})]$

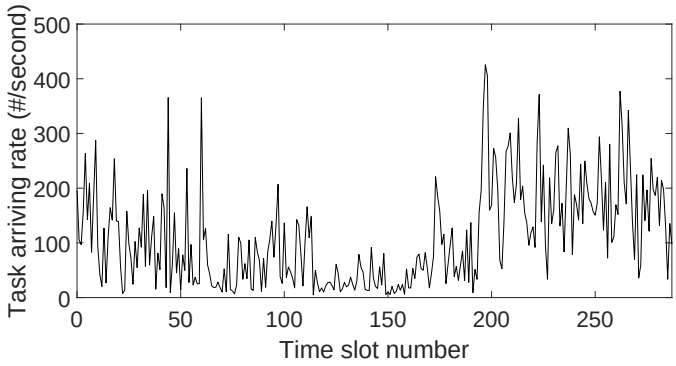


Figure 8.2 Arriving rates of tasks

.

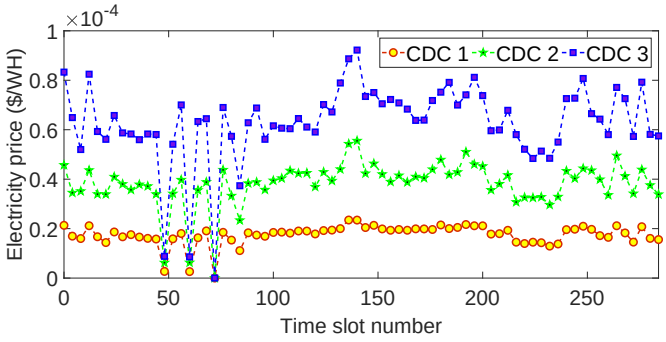


Figure 8.3 Electricity prices in three CDCs.

8.4 illustrates the arriving rates of tasks allocated to three CDCs, respectively. It is observed that the number of tasks allocated to CDC 1 is the largest and that allocated to CDC 3 is the smallest among three CDCs.

Table 8.1 Energy Parameter Setting

	$\urcorner_c$	Ψ_c	$\check{\Phi}_q^c$(W)	$\hat{N}_c$	$\hat{\varrho}_c$(tasks/second)	$\hat{\mathbb{E}}_c$ (WH)
c=1	12.5	1.5	400	140	3.0×10^4	2×10^{27}
c=2	9.4	2.0	500	150	3.1×10^4	2.5×10^{27}
c=3	6.4	2.5	500	160	3.2×10^4	1.25×10^{27}

Figure 8.5 illustrates the number of active servers in three CDCs, respectively. It is observed that they are all less than their corresponding limits. In addition, the number of active servers in CDC 1 is the largest and that in CDC 3 is the smallest among three CDCs. The reason is that CDC 1's electricity price is the lowest and that of CDC 3 is the highest in each time slot. Consequently, more tasks are allocated to CDC 1 and more servers are active than those of CDCs 2 and 3.

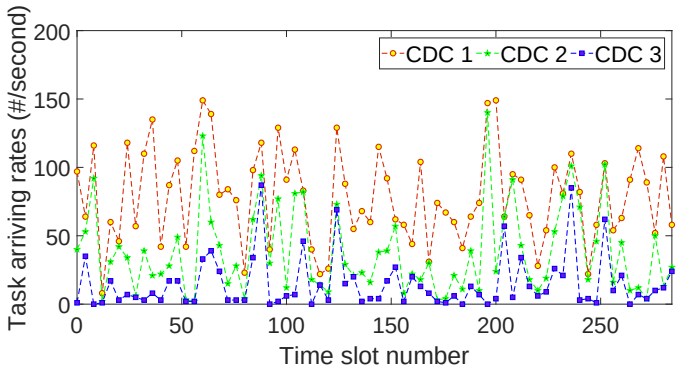


Figure 8.4 Arriving rates of tasks allocated to three CDCs.

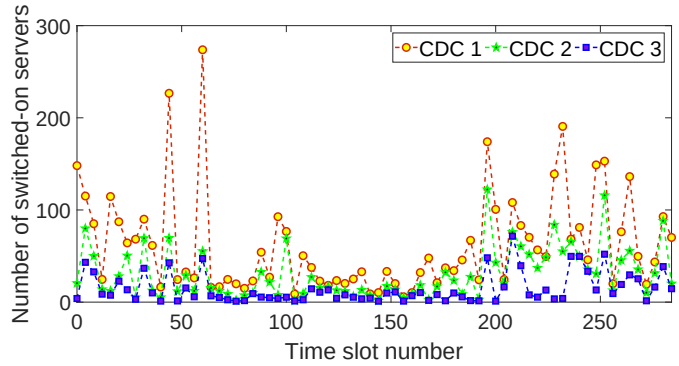


Figure 8.5 Number of active servers in three CDCs.

Figure 8.6 illustrates the convergence result in terms of the task response time and the energy cost of CDCs with SADE in time slot 1. It is observed that they

both decrease with iterations, and converge to stable values at the end of iterations of SADE. This is because the entropy-based crowding distance method increases the candidate diversity in EA, and finally produces a knee solution.

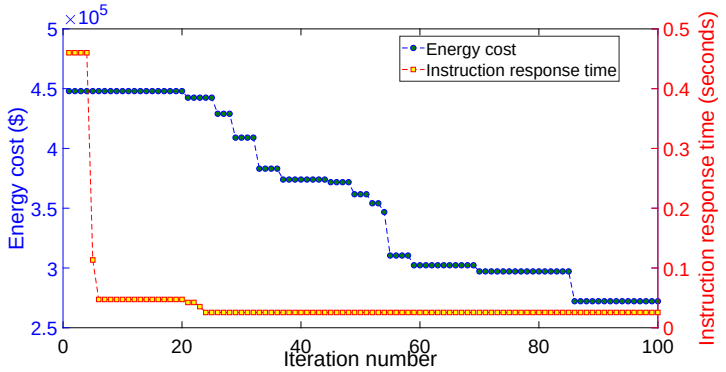


Figure 8.6 Convergence analysis of SADE in time slot 1.

To demonstrate the effectiveness of the entropy-based crowding distance method, this chapter further compares it with the standard EA update [255] and adaptive EA update [257] of MODE.

1) Standard EA update [255]: the truncation operation of EA is conducted as follows. If a new candidate dominates some individuals in EA, it replaces a candidate randomly selected in the most crowded grid.

2) Adaptive EA update [257]: if the number of candidates in EA reaches its limit, a new candidate is handled as follows. If it is within the objective space, it is put into a corresponding hypercube, and a random candidate in the most congested grid is removed. If it is outside the objective space, it is first put into EA. The new objective space is updated and a random candidate in the most congested grid is removed.

Figure 8.7 shows the Pareto-optimal front comparison of the entropy-based crowding distance, adaptive EA update and standard one, respectively. Here, knees 1, 2 and 3 are the knee solutions obtained by them. It is observed that candidates in the Pareto-optimal front of the entropy-based crowding distance are distributed more diversely and evenly than its two peers. Its knee outperforms those with its peers. It demonstrates that the entropy-based crowding distance method cannot only obtain extreme values of the Pareto-optimal front but also well compromised candidates, and it provides a reliable way to solve the problem.

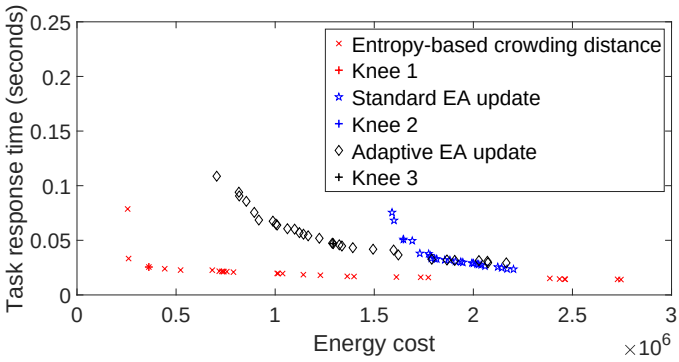


Figure 8.7 Pareto-optimal front comparison.

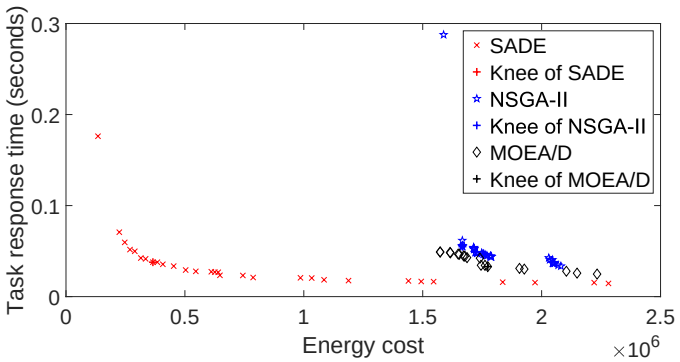


Figure 8.8 Comparison of SADE, NSGA2, and MOEA/D.

To prove the effectiveness of SADE, this chapter compares it with two typical bi-objective optimization algorithms, *i.e.*, Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [258] and Nondominated Sorting Genetic Algorithm 2 (NSGA2) [250]. Figure 8.8 shows the comparison of SADE, NSGA2 and MOEA/D in terms of task response time and energy cost of CDCs, respectively in time slot 1. It is observed that SADE's Pareto-optimal front is distributed more diversely than those of NSGA2 and MOEA/D, respectively. In Figure 8.8, EAs of SADE, NSGA2 and MOEA/D all have 30 non-dominated candidates. The final knee of SADE outperforms those of NSGA2, and MOEA/D with respect to both energy cost and task response time.

In addition, extreme points on both upper and lower limits are obtained by SADE in its Pareto-optimal front. Compared with that of SADE, NSGA2's knee is 86.05% increase in the energy cost and 65.54% increase in the task response time; and

MOEA/D's knee is 49.99% increase in the energy cost and 41.78% increase in the task response time, respectively. The reasons are described as follows. SADE improves the evenness and diversity of the Pareto-optimal front by integrating adaptive mutation, SA-based selection and crossover, and the entropy-based crowding distance method. Thus, its knee is better than those of NSGA2 and MOEA/D.

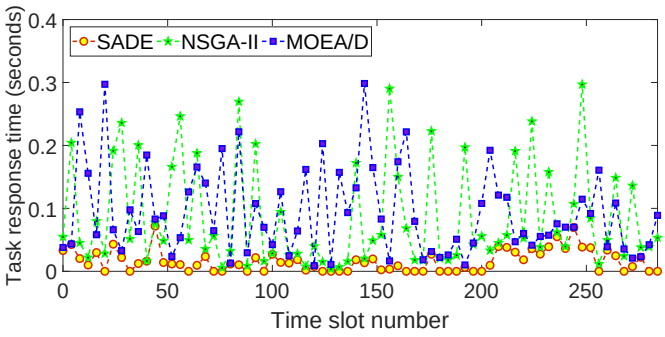


Figure 8.9 Comparison of task response time.

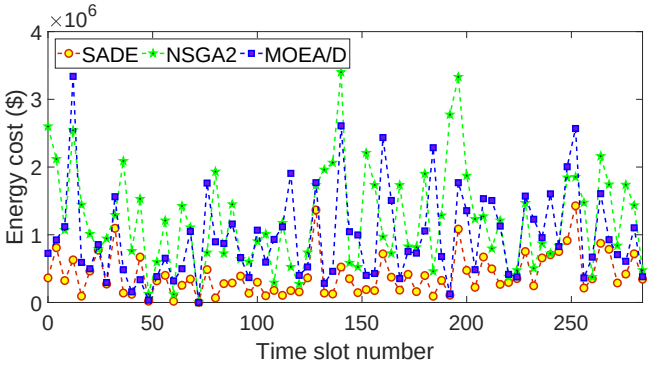


Figure 8.10 Comparison of energy cost.

Figures 8.9 and 8.10 illustrate the comparison of the task response time and the energy cost of SADE, NSGA2, and MOEA/D in each time slot. It is clearly shown that both of SADE are both smaller than those of NSGA2 and MOEA/D in each time slot, respectively. In Figure 8.9, compared with NSGA2 and MOEA/D, SADE reduces the task response time by 58.37% and 63.54% on average, respectively.

In Figure 8.10, compared with NSGA2 and MOEA/D, SADE reduces the energy cost by 61.69% and 53.43% on average, respectively. This is because the Pareto-optimal fronts of NSGA2 and MOEA/D are dominated by that of SADE.

SADE's solutions are more diversely distributed than those of NSGA2 and MOEA/D. Consequently, it proves that SADE finds both well traded-off candidate, and extreme ones in its Pareto-optimal front.

8.4 Appendix

According to the $G/G/1$ queuing theory, the average waiting time of tasks in server q in CDC c, $\mathbb{W}_{\tau}^{c,q}$, is approximately obtained as follows.

$$\mathbb{W}_{\tau}^{c,q}=\frac{1+(\mathbb{\Omega}_{\tau}^{c,q})^2}{(\frac{1}{\rho_{\tau}^{c,q}})^2+(\mathbb{\Omega}_{\tau}^{c,q})^2}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2\bar{\mathbb{T}}_{\tau}^{c,q}(1-\rho_{\tau}^{c,q})} \tag{8.39}$$

$\mathbb{t}_{\tau}^{c,q}$ denotes the random execution time of a server q in CDC c, and $\mathbb{t}_{\tau}^{c,q}=\frac{\mathbb{r}_c^q}{\varrho_{\tau}^{c,q}}$. $\rho_{\tau}^{c,q}$ denotes the utilization of server q in CDC c. Since $\rho_{\tau}^{c,q}<1$, $0\leq\bar{r}_c^q<\bar{\mathbb{T}}_{\tau}^{c,q}\varrho_{\tau}^{c,q}$ is obtained accordingly.

According to (8.39), the following equation is obtained.

$$\begin{aligned}\mathbb{W}_{\tau}^{c,q}&=\frac{1+(\mathbb{\Omega}_{\tau}^{c,q})^2}{(\frac{1}{\rho_{\tau}^{c,q}})^2+(\mathbb{\Omega}_{\tau}^{c,q})^2}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2\bar{\mathbb{T}}_{\tau}^{c,q}(1-\rho_{\tau}^{c,q})}\\&=\frac{1+\frac{\tilde{\sigma}_{\tau,c,q}}{(\bar{\mathbb{t}}_{\tau}^{c,q})^2}}{(\frac{\bar{\mathbb{T}}_{\tau}^{c,q}}{\bar{\mathbb{t}}_{\tau}^{c,q}})^2+\frac{\tilde{\sigma}_{\tau,c,q}}{(\bar{\mathbb{t}}_{\tau}^{c,q})^2}}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2\bar{\mathbb{T}}_{\tau}^{c,q}(1-\rho_{\tau}^{c,q})}\\&=\frac{(\bar{\mathbb{t}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}{(\bar{\mathbb{T}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2\bar{\mathbb{T}}_{\tau}^{c,q}(1-\rho_{\tau}^{c,q})}\end{aligned} \tag{8.40}$$

Then, the average response time of tasks in server q in CDC c in time slot τ is denoted by $\mathbb{T}_{\tau}^{c,q}$, and it is obtained as:

$$\begin{aligned}\mathbb{T}_{\tau}^{c,q}&=\bar{\mathbb{t}}_{\tau}^{c,q}+\mathbb{W}_{\tau}^{c,q}\\&=\bar{\mathbb{t}}_{\tau}^{c,q}+\frac{(\bar{\mathbb{t}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}{(\bar{\mathbb{T}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2\bar{\mathbb{T}}_{\tau}^{c,q}(1-\rho_{\tau}^{c,q})}\\&=\bar{\mathbb{t}}_{\tau}^{c,q}+\frac{(\bar{\mathbb{t}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}{(\bar{\mathbb{T}}_{\tau}^{c,q})^2+\tilde{\sigma}_{\tau,c,q}}\frac{\hat{\sigma}_{\tau,c,q}+\tilde{\sigma}_{\tau,c,q}}{2(\bar{\mathbb{T}}_{\tau}^{c,q}-\bar{\mathbb{t}}_{\tau}^{c,q})}\\&=\frac{\bar{r}_c^q}{\varrho_{\tau}^{c,q}}+\frac{\frac{(\bar{r}_c^q)^2}{\varrho_{\tau}^{c,q}}+\frac{\breve{\sigma}_{c,q}}{(\varrho_{\tau}^{c,q})^2}}{(\bar{\mathbb{T}}_{\tau}^{c,q})^2+\frac{\breve{\sigma}_{c,q}}{(\varrho_{\tau}^{c,q})^2}}\frac{\hat{\sigma}_{\tau,c,q}+\frac{\breve{\sigma}_{c,q}}{(\varrho_{\tau}^{c,q})^2}}{2(\bar{\mathbb{T}}_{\tau}^{c,q}-\frac{\bar{r}_c^q}{\varrho_{\tau}^{c,q}})}\end{aligned}$$

$$=\frac{\bar{r}_c^q}{\varrho_\tau^{c,q}}+\frac{(\bar{r}_c^q)^2+\breve{\sigma}_{c,q}}{(\bar{\mathbb{T}}_\tau^{c,q})^2(\varrho_\tau^{c,q})^2+\breve{\sigma}_{c,q}}\frac{\hat{\sigma}_{\tau,c,q}(\varrho_\tau^{c,q})^2+\breve{\sigma}_{c,q}}{2\varrho_\tau^{c,q}(\bar{\mathbb{T}}_\tau^{c,q}\varrho_\tau^{c,q}-\bar{r}_c^q)}$$
$$=\frac{\bar{r}_c^q}{\varrho_\tau^{c,q}}+((\bar{r}_c^q)^2+\breve{\sigma}_{c,q})\frac{\hat{\sigma}_{\tau,c,q}(\varrho_\tau^{c,q})^2+\breve{\sigma}_{c,q}}{2\varrho_\tau^{c,q}(\bar{\mathbb{T}}_\tau^{c,q}\varrho_\tau^{c,q}-\bar{r}_c^q)((\bar{\mathbb{T}}_\tau^{c,q})^2(\varrho_\tau^{c,q})^2+\breve{\sigma}_{c,q})}$$

Then, the average response time of tasks in CDC c with $\overset{o}{N}_{\tau,c}$ servers is:

$$T_\tau^c=\frac{1}{\tilde{\lambda}_\tau^c}\sum_{q=1}^{\overset{o}{N}_{\tau,c}}\lambda_\tau^{c,q}\hat{\sigma}_{\tau,c,q} \tag{8.41}$$

It is worth noting that equation (8.8) is equivalent to equation (8.41). Then, equation (8.8) is derived accordingly.

8.5 Summary

A growing number of large-scale companies own distributed cloud data centers (CDCs) around the world to provide services to their global users with resource sharing. Each application is deployed in multiple geographically CDCs for low energy cost and fast response. Current CDCs face a big challenge of how to jointly decrease the energy cost of CDCs and improve Quality of Service (QoS). In this chapter, the joint optimization of energy cost and QoS is formulated as a bi-objective constrained optimization problem and solved by a Simulated-annealing-based Adaptive Differential Evolution (SADE) algorithm to obtain a close-to-Pareto-optimal set. In this way, this chapter properly allocates arriving tasks among CDCs, and changes task service rates of each CDC in each time slot. Real-life data-based results prove that SADE reduces energy cost and response time of tasks compared with several scheduling peers.

CHAPTER 9

REVENUE AND ENERGY COST-OPTIMIZED BI-OBJECTIVE TASK SCHEDULING FOR GREEN CLOUD DATA CENTERS

This chapter presents the details of a proposed revenue and energy cost-optimized bi-objective task scheduling method, and it is organized as follows. Section 9.1 presents a system model. Section 9.2 formulates the problem. Section 9.3 proposes the details of an Improved Multi-objective Evolutionary Algorithm based on Decomposition (IMEAD). Section 9.4 presents its performance evaluation results. Section 9.5 concludes the chapter.

9.1 System Model

This section introduces the proposed system model. Figure 9.1 illustrates the system architecture of green CDCs. For performance concerns, CDC providers typically manage several CDCs distributed in multiple geographical places and provide different applications to users around the world. Similar to the work in [80], it is assumed that programs and data for all applications are consistent and the same in all CDCs. Users send their tasks to CDCs with different types of devices, *e.g.*, smart phones, laptops and computers. Then, tasks are scheduled and executed with a First-Come-First-Serve (FCFS) policy [259].

The information of task queues is periodically transmitted to a centralized *Task Scheduler*. Each CDC is switched by three energy sources including power grid, wind energy and solar energy. Energy information includes electricity prices, wind speed and solar irradiance, which are supplied to *Task Scheduler*. Based on such information, *Task Scheduler* executes IMEAD to trade off the revenue maximization of CDC providers, and its energy cost minimization by smartly executing arriving tasks among CDCs. It runs IMEAD that takes the advantages of various spatial differences in CDCs while meeting delay constraints of tasks of applications.

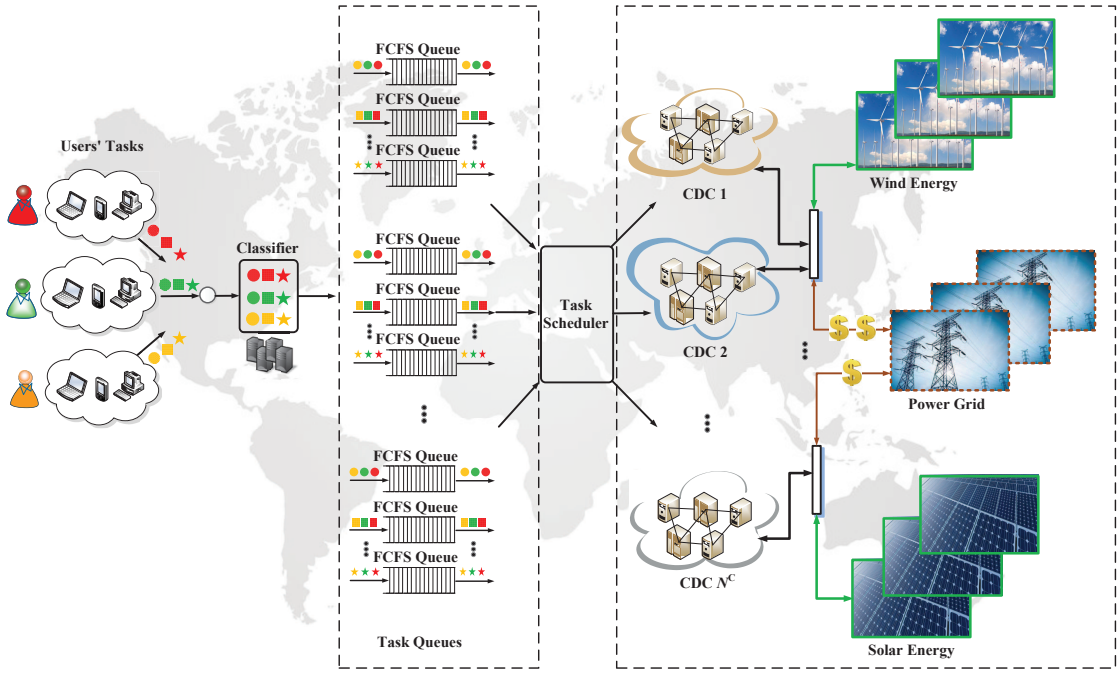


Figure 9.1 System architecture.

The arriving rates of tasks for different applications can vary from time to time among CDCs. To improve CDC performance and maximize its profit, this chapter first develops a system model. The objective is to maximize the revenue and minimize the cost generated by different types of applications under a time-varying workload by dynamically tuning the following parameters at the beginning of each control time slot. Parameters are summarized in Table 9.1.

9.1.1 Task Arriving and Service Rates

Let $\lambda_\tau^{c,n}$ denote the task arriving rate of application n scheduled to CDC c in time slot τ, and $\mathcal{N}^C$ denotes the number of CDCs. Then the task arriving rate of application n to all CDCs is:

$$\lambda_\tau^n = \sum_{c=1}^{\mathcal{N}^C} \lambda_\tau^{c,n} \tag{9.1}$$

Let $\mu_\tau^{c,n}$ denote the rate at which tasks are removed from the queue of application n and handled by CDC c. The service rate depends on the number of servers that are switched-on. $\overset{o}{N}_{\tau,c,n}$ denotes the number of switched-on servers of application n in

Table 9.1 Problem Parameters

Notation	Definition
$\mathcal{N}^A$	Number of applications
$\mathcal{N}^C$	Number of CDCs
$\overset{\bullet}{\mathbb{N}}_n$	Number of tasks processed by each server for application n in each minute
$\overset{o}{N}_{\tau,c,n}$	Number of switched-on servers for application n in CDC c in time slot τ
$\hat{N}_c$	Total number of servers in CDC c
$\lambda_\tau^{c,n}$	Task arriving rate of application n in CDC c in time slot τ
$\hat{\mathbb{Q}}_n$	Maximum number of tasks processed by each server for application n
L	Length of each time slot τ
$\check{\Phi}$	Idle power of a single server
$\hat{\Phi}$	Peak power when a server is handling a task
α	Power usage effectiveness in a CDC
p_τ^c	Price of electricity in CDC c in time slot τ
$\overset{+}{E}_{\tau,c}$	Amount of green energy in CDC c in time slot τ
ψ_{1c}	Conversion rate of solar irradiance to electricity in CDC c
ψ_{2c}	Active irradiance area of solar panels in CDC c
$\psi_{\tau,3c}$	Solar irradiance in CDC c in time slot τ
ϕ_{1c}	Conversion rate of wind energy to electricity in CDC c
ϕ_{2c}	On-site air density in CDC c
ϕ_{3c}	Rotor area of wind turbines in CDC c
$\phi_{\tau,4c}$	Wind speed in CDC c in time slot τ

CDC c in time slot τ. Each server of application n can handle $\overset{\bullet}{\mathbb{N}}_n$ tasks per minute. Therefore, $\mu_\tau^{c,n}$ is obtained as:

$$\mu_\tau^{c,n} = \overset{\bullet}{\mathbb{N}}_n \overset{o}{N}_{\tau,c,n} \tag{9.2}$$

subject to

$$\sum_{n=1}^{\mathcal{N}^A} \overset{o}{N}_{\tau,c,n} \leq \hat{N}_c \tag{9.3}$$

$$\overset{o}{N}_{\tau,c,n} \geq 0 \tag{9.4}$$

$$\mu_\tau^{c,n} \geq 0 \tag{9.5}$$

$$1 \leq n \leq \mathcal{N}^A, 1 \leq c \leq \mathcal{N}^C \tag{9.6}$$

where $\mathcal{N}^A$ denotes the number of applications, and $\hat{N}_c$ denotes the total number of available servers in CDC c.

As the number of switched-on servers and accordingly the service rate increase, more tasks can be handled before a deadline in service level agreements (SLAs), which in turn increases the payments that a CDC receives. On the other hand, it also increases CDCs' power consumption and accordingly their energy cost. Therefore, there is a tradeoff when determining the service rate of CDCs. In time slot τ, to guarantee the stability of the task queue of application n in CDC c, $\lambda_{\tau}^{c,n}$ must be less than $\mu_{\tau}^{c,n}$. Therefore,

$$\lambda_{\tau}^{c,n} < \mu_{\tau}^{c,n} \tag{9.7}$$

9.1.2 Service Level Agreements

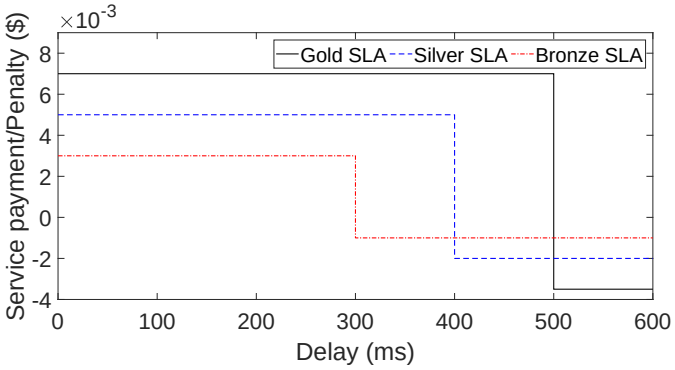


Figure 9.2 SLAs for CDCs.

CDCs cannot process all arriving tasks immediately after they arrive. The reason is that CDCs only have limited computation capacities, and the practical tasks have the stochastic nature. Therefore, all kinds of arriving tasks are placed in different queues until they are handled by any available server. To satisfy the SLA requirements, the queuing delay or waiting time for different types of arriving tasks should be limited within a certain range. The exact SLA depends on types of applications offered by CDCs, *e.g.*, e-commerce and video streaming, and cloud-based computational tasks. Similar to the work in [117], Figure 9.2 shows three typical SLAs for CDCs. In Figure 9.2, each SLA is identified by three non-negative parameters including $\hat{T}_n$, ∇_n and ϵ_n.

$\hat{T}_n$ denotes the maximum queuing delay that a task can tolerate. ∇_n denotes the revenue that CDCs receive when it handles a single task before deadline $\hat{T}_n$. ϵ_n denotes the penalty that CDCs have to pay to its users each time slot τ it cannot handle a task before deadline $\hat{T}_n$. For example, in the Gold SLA, $\hat{T}_1$= 500 ms, ∇_1=7×10^{-3} dollars and ϵ_1=3.5×10^{-3} dollars. In the Silver SLA, $\hat{T}_2$= 400 ms, ∇_2=5×10^{-3} dollars and ϵ_2=2×10^{-3} dollars. In the Bronze SLA, $\hat{T}_3$= 300 ms, ∇_3=3×10^{-3} dollars and ϵ_3=1×10^{-3} dollars.

9.1.3 Energy Consumption

The total amount of energy consumption in a CDC is obtained by adding the total energy consumption at the servers to the total energy consumption at the cooling and lighting facilities. For a CDC, power usage effectiveness (PUE) is defined as the ratio of the CDC's total energy consumption to the CDC's energy consumption at the servers. It is considered as a measure for CDC's energy efficiency. The lower, the more efficient. Currently, the PUE value for most enterprise data centers is 2.0 or more. A few state-of-the art facilities have reached a PUE of 1.2 [24].

To reduce CDC's energy consumption, the number of switched-on servers should be dynamically adjusted according to the rate of received tasks. This chapter focuses on critical machine-level energy consumption, and uses CPU utilization as the main signal of machine-level activities. Therefore, the energy consumption for a CDC is modeled such that it is proportional and roughly linear to its utilization. Multiple studies have shown that CPU utilization is indeed a good estimator for power usage [260]. The machine-level total energy consumption associated with the CDC is obtained as [261]:

$$E_{\tau}^{c}= \left(\overset{o}{N}_{\tau,c,n}(\check{\Phi}+(\alpha-1)\hat{\Phi})+\overset{o}{N}_{\tau,c,n}(\hat{\Phi}-\check{\Phi})u+\overset{0}{\gamma}_6 \right) L \tag{9.8}$$

where $u \in [0,1]$ denotes the CPU utilization of servers. $\overset{0}{\gamma}_6$ denotes an empirically derived correction constant. Note that in this chapter, $\overset{0}{\gamma}_6$ is set to zero, which means

the empirically derived correction factor is ignored. $\hat{\Phi}$ denotes the peak power when a server is handling a task. $\check{\Phi}$ is the idle power of an idle server. $\frac{\hat{\Phi}}{\check{\Phi}}$ denotes the power elasticity of servers. Higher elasticity means less energy consumption when the server is idle and not handling any task. α denotes the PUE of a CDC. From (9.8), the energy consumption at a CDC increases as more active servers at higher utilization.

9.1.4 Renewable Energy Generation

In the system model, the price of electricity is denoted by p_{τ}^{c}. To reduce cost of electricity, CDCs may be equipped with behind-the-meter renewable generators, *e.g.*, wind turbine and solar panel, in addition to being connected to the power grid. Let $\overset{+}{E}_{\tau,c}$ denote the renewable energy generated by renewable energy generators in CDC c in time slot τ. The renewable energy $\overset{+}{E}_{\tau,c}$ includes wind and solar energy.

1) Solar Energy

Let $\overset{\circ}{E}_{\tau,c}$ denote the solar energy consumed by the execution of tasks of all applications in CDC c in time slot τ. Following the work in [191], $\overset{\circ}{E}_{\tau,c}$ is obtained as:

$$\overset{\circ}{E}_{\tau,c}=\psi_{1c}\psi_{2c}I_c(\tau)T \tag{9.9}$$

where ψ_{1c} denotes the conversion rate of solar irradiance to electricity in CDC c, ψ_{2c} denotes the active irradiance area of solar panels, and $\psi_{\tau,3c}$ denotes the solar irradiance in CDC c in time slot τ.

2) Wind Energy

Let $\widetilde{E}_{\tau,c}$ denote the wind energy consumed by the execution of tasks of all applications in CDC c in time slot τ. Following the work in [191], $\widetilde{E}_{\tau,c}$ is obtained as:

$$\widetilde{E}_{\tau,c}=\frac{1}{2}\phi_{1c}\phi_{2c}\phi_{3c}(\phi_{\tau,4c})^3T \tag{9.10}$$

where ϕ_{1c} denotes the conversion rate of wind to electricity in CDC c, ϕ_{2c} denotes the on-site air density in CDC c, ϕ_{3c} denotes the rotor area of wind turbines in CDC c, and $\phi_{\tau,4c}$ denotes the wind speed in CDC c in time slot τ.

In this chapter, the amount of power grid energy consumed by CDC c is calculated as $E^c_\tau - \overset{+}{E}_{\tau,c}$. If the energy consumption of CDC c is more than local renewable power generation, *i.e.*, $E^c_\tau > \overset{+}{E}_{\tau,c}$, $E^c_\tau - \overset{+}{E}_{\tau,c}$ is positive and the power flow is in the direction from the power grid to CDC c. If $E^c_\tau = \overset{+}{E}_{\tau,c}$, CDC c operates as a zero-net energy facility [262]. If $E^c_\tau < \overset{+}{E}_{\tau,c}$, $E^c_\tau - \overset{+}{E}_{\tau,c}$ is negative and the power flow is in the direction from CDC c to the power grid. It is worth noting that, in the system model, it is assumed that the CDC provider does not receive compensation for the injected power, while CDCs are allowed to inject their excessive renewable energy into the power grid.

9.2 Problem Formulation

The task arriving rate at a CDC varies over time. To improve CDCs' performance, the number of switched-on servers, $\overset{o}{N}_{\tau,c,n}$, should be adjusted according to the rate of arriving tasks. When tasks are received at higher rates, more servers need to be switched-on. A proportional-to-demand portion of servers are switched-on by monitoring task arriving rates. This results in reducing CDCs' energy consumption. Because there is the tear-and-wear cost of switching servers on and off, and the delay in changing the status of a server cannot be changed instantly, it is rather desired to be updated by every few minutes. Therefore, this chapter divides the running time of a CDC into many time slots with the same length of L, *e.g.*, $L = 5$ minutes. Note that the number of switched-on servers is updated only at the start of each time slot.

9.2.1 Revenue Modeling

Let f_1 denote the total revenue of CDCs in each time slot, and it is calculated as:

$$f_1 = \sum_{n=1}^{\mathcal{N}^A} \sum_{c=1}^{\mathcal{N}^C} \{[(1 - \delta^{c,n}_\tau)\nabla_n - \delta^{c,n}_\tau \epsilon_n]\lambda^{c,n}_\tau L\} \tag{9.11}$$

where $(1 - \delta^{c,n}_\tau)\nabla_n \lambda^{c,n}_\tau L$ denotes the total revenue received by CDCs for tasks of application n that are handled before their SLA deadline. $\delta^{c,n}_\tau \epsilon_n \lambda^{c,n}_\tau L$ denotes the

total penalty paid by users for performing tasks of application n that are not handled before their SLA deadline.

9.2.2 Cost Modeling

The machine-level energy consumption of a CDC is usually determined by unit-time power usage E_τ. Thus, according to the work in [24], the total energy consumption of switched-on servers in CDC c is obtained as:

$$E_\tau^c = \sum_{n=1}^{\mathcal{N}^A} \left[\frac{\left(\check{\Phi} - +(\alpha-1)\hat{\Phi}\right) \mu_\tau^{c,n} + \left(\hat{\Phi} - \check{\Phi}\right) \lambda_\tau^{c,n}(1-\delta_\tau^{c,n})}{\overset{\bullet}{\mathbb{N}}_n} \right] \tag{9.12}$$

Let p_τ^c denote the electricity price of CDC c in time slot τ. Let f_{22} denote the net energy consumption cost in each time slot $_\tau$. It is obtained as:

$$f_{22} = \sum_{c=1}^{\mathcal{N}^C} \{ Lp_\tau^c \left[E_\tau^c - \overset{+}{E}_{\tau,c} \right]^+ \} \tag{9.13}$$

$$= \sum_{c=1}^{\mathcal{N}^C} \{ Lp_\tau^c \max(E_\tau^c - \overset{+}{E}_{\tau,c}, 0) \} \tag{9.14}$$

In this study, it is assumed that PUE in CDCs is 1.2 by following the work in [24]. It is worth noting that this chapter focuses on energy management of CDCs, and the cost model only indicates the cost of electricity. Other cost items can be also included similarly in the model.

9.2.3 Task Loss Probability

$\delta_\tau^{c,n}$ denotes the loss probability of tasks of application n in CDC c in τ. Then,

$$\delta_\tau^{c,n} = \begin{cases} 1, & \mu_\tau^{c,n} = 0 \\ \frac{1}{\hat{\mathbb{Q}}_n+1}, & \lambda_\tau^{c,n} = \mu_\tau^{c,n} \neq 0 \\ \frac{\left(1-\left(\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}\right)\right)\left(\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}\right)^{\hat{\mathbb{Q}}_n}}{\left(1-\left(\frac{\lambda_\tau^{c,n}}{\mu_\tau^{c,n}}\right)\right)^{\hat{\mathbb{Q}}_n+1}}, & \lambda_\tau^{c,n} < \mu_\tau^{c,n} \end{cases} \tag{9.15}$$

subject to

$$\lambda_{\tau}^{c,n} - \frac{1}{\hat{T}_n} \leq \mu_{\tau}^{c,n} \tag{9.16}$$

where $\lambda_{\tau}^{c,n}$ denotes the task arriving rate of application n in CDC c in τ. $\hat{\mathbb{Q}}_n$ denotes the maximum number of tasks processed by each server for application n.

Then, this chapter makes further discussion on the assumptions of the proposed IMEAD. It is assumed that the future information (*e.g.*, wind speed, task arriving rate, solar irradiance and electricity prices) is available in advance. In real CDCs, the prediction of such information usually needs some time to obtain the prediction model by using historical data. However, the recently emerging big data techniques (*e.g.*, deep learning-based prediction [263]), and the increasingly deployed high-performance servers in CDCs [264] will make it possible that the prediction time is negligible.

9.3 Improved Multi-objective Evolutionary Algorithm based on Decomposition

This section presents the proposed IMEAD to jointly optimize the revenue and cost of CDC providers. By assigning different types of tasks to multiple CDCs, the revenue and energy cost of CDC providers are balanced by finding a high-quality Pareto front [265]–[268] . This chapter considers the following two objectives:

$$\mathbf{Max}_{\boldsymbol{x}} \ \{f_1\}$$

$$\mathbf{Min}_{\boldsymbol{x}} \ \{f_{22}\}$$

subject to (9.1), (9.3)–(9.6), (9.7) and (9.16).

Here, $\boldsymbol{x}$ denotes the vector of all decision variables, *i.e.*, $\lambda_{\tau}^{c,n}$ and $\mu_{\tau}^{c,n}$ (n=1, 2, $\cdots$, $\mathcal{N}^A$, c=1, 2, $\cdots$, $\mathcal{N}^C$). Thus, $\boldsymbol{x}$=$[\lambda_{\tau}^{1,1}, \cdots, \lambda_{\tau}^{\mathcal{N}^A,\mathcal{N}^C}, \mu_{\tau}^{1,1}, \cdots, \mu_{\tau}^{\mathcal{N}^A,\mathcal{N}^C}]$.

This chapter uses a penalty function method to transform the constrained bi-objective problem into an unconstrained one. The transformation method is shown as follows. For clarity, this chapter uses the following single-objective optimization

problem to show the details of the penalty function method.

$$\begin{cases} \textbf{Min } \beth(\boldsymbol{x}), \\ \text{subject to } g_l(\boldsymbol{x}) \geq 0, l=1,\cdots,\mathcal{N}^{\neq} \\ h_m(\boldsymbol{x})=0, m=1,\cdots,\mathbb{N}^{=} \end{cases} \tag{9.17}$$

(9.17) is an objective function with constraints, and it includes both equality constraints $h_m(\boldsymbol{x})$ and inequality ones $g_l(\boldsymbol{x})$. $\mathcal{N}^{\neq}$ denotes the number of inequality constraints. $\mathbb{N}^{=}$ denotes the number of equality constraints. This chapter transforms the constrained problem into an unconstrained one with an auxiliary function in (9.18), which is given as follows:

$$\begin{aligned} \Delta^{15} &= \beth(\boldsymbol{x}) + \overset{\infty}{\mathcal{N}} \mho \\ \mho &= \sum_{l=1}^{\mathcal{N}^{\neq}} (\max\{0, -g_l(\boldsymbol{x})\})^{\overset{0}{\gamma}_1} + \sum_{m=1}^{\mathbb{N}^{=}} \left|h_m(\boldsymbol{x})\right|^{\overset{0}{\gamma}_2} \end{aligned} \tag{9.18}$$

where $\overset{0}{\gamma}_1 \geq 1$ and $\overset{0}{\gamma}_2 \geq 0$, Δ^{15} is a transformed objective function, $(\max\{0, -g_l(\boldsymbol{x})\})^{\overset{0}{\gamma}_1}$ is the penalty for each inequality constraint l, and $\left|h_m(\boldsymbol{x})\right|^{\overset{0}{\gamma}_2}$ is the penalty for each equality constraint m.

Then, the constrained optimization problem (9.17) is transformed as:

$$\underset{\boldsymbol{x}}{\textbf{Min}} \left\{ \beth(\boldsymbol{x}) + \overset{\infty}{\mathcal{N}} \mho \right\} \tag{9.19}$$

where $\overset{\infty}{\mathcal{N}}$ is a large positive constant.

This chapter designs an IMEAD algorithm that includes crossover, mutation and selection operations. It has several advantages. First, the parameters of its crossover and mutation operations dynamically change in the convergence process. The dynamic parameters strengthen the ability of global search and accelerate the convergence speed. It prevents good genes from being destroyed by mutation, and introduces new genes into the populations when they fall into the locally optimal solutions. Second, this chapter uses a decomposition-based Tchebycheff method [269]

to decompose the problem into a set of single-objective-optimization subproblems. Third, this chapter adopts a crowded degree evaluation method based on crowded distance. Based on it, individuals with lower crowded degrees are removed from the external elite set when the number of individuals exceeds the capacity limit of the external elite set. Its main operations are shown as follows.

9.3.1 Population Initialization

The population initialization adopts a random initialization method, *i.e.*, random initialization is performed within the feasible domain of each decision variable. Let $\mathbb{N}^D$ denote the number of decision variables. Let $|\mathbb{X}|$ denote the number of individuals. Let $\mathbf{x}_{i,d}$ denote the value of decision variable d of individual i ($i \in \{1, 2, \cdots, |\mathbb{X}|\}$, $d \in \{1, 2, \cdots, \mathbb{N}^D\}$). The individuals are initialized as follows.

$$\mathbf{x}_{i,d} = \check{\boldsymbol{\theta}}_5^d + w_4 * (\hat{\boldsymbol{\theta}}_5^d - \check{\boldsymbol{\theta}}_5^d) \tag{9.20}$$

where $\hat{\boldsymbol{\theta}}_5^d$ and $\check{\boldsymbol{\theta}}_5^d$ denote the maximum and minimum values of decision variable d. w_4 denotes a random number generated uniformly from (0,1], *i.e.*, $w_4 \in (0, 1]$.

9.3.2 Dynamic Crossover and Mutation

This chapter adopts dynamic crossover and mutation parameters to produce a new population. In the early stage, large crossover and mutation parameters are adopted to increase the reproduction efficiency and diversity of population. In the late stage, as the number of iterations increases, small crossover and mutation parameters are adopted to prevent high-quality genes from being destroyed. Then, better individuals are kept. The crossover and mutation parameters are updated as:

$$\theta_7 = \hat{\theta}_7 - \frac{(\hat{\theta}_7 - \check{\theta}_7)}{(\hat{g}/g)} \tag{9.21}$$

$$\theta_8 = \hat{\theta}_8 - \frac{(\hat{\theta}_8 - p_\iota^{min})}{(\hat{g}/g)} \tag{9.22}$$

where θ_7 denotes the crossover possibility, $\hat{\theta}_7$ denotes the maximum crossover possibility, $\check{\theta}_7$ denotes the minimum crossover possibility, θ_8 denotes the mutation possibility, $\hat{\theta}_8$ denotes the maximum mutation possibility, $\check{\theta}_8$ denotes the minimum mutation possibility, q denotes the current iteration number, and $\hat{g}$ denotes the total number of iterations.

9.3.3 Crossover Operation

The crossover operation can increase the population diversity for generating new individuals. Let $\theta_{1,i}^M$ denote the neighboring area of individual i. $\mathbf{x}_{w_5}$ and $\mathbf{x}_{w_6}$ denote two random individuals in $\theta_{1,i}^M$. $\mathbf{x}_{w_7}$ denotes an individual randomly selected from populations $\mathbb{X}$. Let $\mathbf{x}_i^1$ denote a new individual. Then, $\mathbf{x}_i^1$ is obtained as:

$$\mathbf{x}_i^1 = \begin{cases} w_4 * \mathbf{x}_{w_5} + (1 - w_4) * \mathbf{x}_{w_6}, w_4 < \theta_7 \\ \mathbf{x}_{w_7}, \text{otherwise} \end{cases} \tag{9.23}$$

9.3.4 Mutation Operation

Mutation operations are equivalent to genetic mutations in biology. In evolutionary computational patterns, they refer to changes or disturbances to a random factor.

In this chapter, the Gaussian mutation operation is used to increase the population diversity, which is beneficial to jump out of local extreme points for global search and avoid the algorithm to fall into local optima. It also improves the search speed. Gaussian distribution is a class of probability distributions that are important in mathematics, physics and engineering.

It has significant impact on many aspects of statistics. Gaussian mutation aims to add a random vector obeying Gaussian distribution to the state of an individual, and then select individual decision variables according to the given mutation probability. The mutation operation is obtained as:

$$\boldsymbol{\theta}_3^M = (\hat{\boldsymbol{\theta}}_5 - \check{\boldsymbol{\theta}}_5)\mathbb{N}^D \tag{9.24}$$

$$\theta_5^M = min(\max(w_8(\mathbf{x}_i^1, \boldsymbol{\theta}_3^M), \check{\boldsymbol{\theta}}_5), \hat{\boldsymbol{\theta}}_5) \tag{9.25}$$

$$\boldsymbol{\theta}_4^M = rand(1, \mathbb{N}^D) < \theta_8 \tag{9.26}$$

$$X_{new} = \theta_5^M(\boldsymbol{\theta}_4^M) \tag{9.27}$$

where $\check{\boldsymbol{\theta}}_5$ denotes the minimum value vector of decision variables, $\hat{\boldsymbol{\theta}}_5$ denotes the maximum value vector of decision variables, $\boldsymbol{\theta}_3^M$ denotes the variance vector, and w_8 denotes a random vector generated from the Gaussian distribution.

In addition, (9.25) produces a new solution around $\mathbf{x}_i^1$ and guarantees that this solution is between $\check{\boldsymbol{\theta}}_5$ and $\hat{\boldsymbol{\theta}}_5$. Besides, (9.26) returns a binary vector $\boldsymbol{\theta}_4^M$ with the length of $\mathbb{N}^D$. In $\boldsymbol{\theta}_4^M$, if a random number $rand(1, d)(1 \leq d \leq \mathbb{N}^D)$ is less than θ_8, $\boldsymbol{\theta}_4^M(1, d)=1$; otherwise, $\boldsymbol{\theta}_4^M(1, d)=0$. Then, if $\boldsymbol{\theta}_4^M(1, d)=1$, $\mathbf{x}_i^1(1, d)$ is updated by $\theta_5^M(1, d)$; otherwise, $\mathbf{x}_i^1(1, d)$ is not changed.

9.3.5 Update of Neighboring Solutions

Then, the update of neighboring solutions $\theta_{1,i}^M$ for individual i is introduced here. It is assumed that there are $\tilde{M}$ individuals in $\theta_{1,i}^M$. Given individual i, $\tilde{M}$ individuals are chosen as its neighboring solutions, *i.e.*, $\theta_{1,i}^M$. For each $\mathbf{x}_j \in \theta_{1,i}^M$, based on (9.28), $\mathbf{x}_i^1$ is chosen to update $\mathbf{x}_j$ ($\mathbf{x}_j \in \theta_{1,i}^M$), *i.e.*,

$$\mathbf{x}_j = \mathbf{x}_i^1, \text{if } \theta_{2,i}^M(\mathbf{x}_i^1 | \boldsymbol{\theta}_{7,j}^M, \boldsymbol{\theta}_6^M) \leq \theta_{2,i}^M(\mathbf{x}_j | \boldsymbol{\theta}_{7,j}^M, \boldsymbol{\theta}_6^M) \tag{9.28}$$

$$\theta_{2,i}^M(\mathbf{x}_j | \boldsymbol{\theta}_{7,j}^M, \boldsymbol{\theta}_6^M) = max_{1 \leq \iota \leq \overset{o}{M}} \left\{ \boldsymbol{\theta}_{7,j,\iota}^M | \widetilde{f}_\iota(\mathbf{x}_j) - \boldsymbol{\theta}_{6,\iota}^M | \right\} \tag{9.29}$$

where $\overset{o}{M}$ denotes the number of objective functions, $\boldsymbol{\theta}_{7,j,\iota}^M$ ($\boldsymbol{\theta}_{7,j,\iota}^M \in (0, 1)$) denotes the weight of objective function ι for $\mathbf{x}_j$, $\boldsymbol{\theta}_{7,j}^M$ denotes a vector of $\boldsymbol{\theta}_{7,j,\iota}^M$, $\widetilde{f}_\iota$ denotes the ιth ($\iota \in \{1, 2, \ldots, \overset{o}{M}\}$) augmented objective function. and $\boldsymbol{\theta}_6^M$ is a vector of the best objective function values currently obtained. In this way, neighboring solutions $\theta_{1,i}^M$ for individual i is obtained.

9.3.6 Crowded Degree Evaluation based on Crowded Distance

To preserve better elite individuals, this chapter uses the external elite set $\breve{\Omega}$ to store elite individuals. Let $|\breve{\Omega}|$ denote the number of individuals in $\breve{\Omega}$. After initializing the population, the objective function values in the population are calculated, and the dominant relation of individuals is determined according to the objective function values. The non-dominated individuals are stored in $\breve{\Omega}$. In each iteration, the newly generated individual is compared with individuals in $\breve{\Omega}$. If the new individual dominates an individual in $\breve{\Omega}$, the dominated individual is removed. Otherwise, if the number of individuals in $\breve{\Omega}$ is less than its specified capacity, the new individual is directly added to $\breve{\Omega}$.

If it is greater than its capacity, the crowded degree evaluation method based on crowded distance is adopted to increase the diversity of population and the individuals with lower crowded degrees are removed and replaced with the newly generated non-dominated solutions. Specifically, first, the solutions in $\breve{\Omega}$ are sorted according to their values of each objective function. Then, the maximum ($\hat{\theta}_6^{\iota}$) and minimum ($\breve{\theta}_6^{\iota}$) of objective function ι are obtained. $\theta_{8,i}^{M}$ denotes the crowded distance of individual i. The crowding distance is ∞ when $\hat{\theta}_6^{\iota}$ is equal to $\breve{\theta}_6^{\iota}$, or the first/last individual in the Pareto-optimal front occurs. Besides, the sum of distances on each objective function is calculated as follows:

$$\theta_{8,i}^{M} = \begin{cases} \infty, \text{if } \hat{\theta}_6^{\iota} - \breve{\theta}_6^{\iota} = 0 \text{ or } i = 1 \; or \; i = |\breve{\Omega}| \\ \sum\limits_{\iota=1}^{o} \left(\frac{\acute{\theta}_{9,i}^{M} - \grave{\theta}_{9,i}^{M}}{\hat{\theta}_6^{\iota} - \breve{\theta}_6^{\iota}} \right), \text{otherwise} \end{cases} \tag{9.30}$$

where $\acute{\theta}_{9,i}^{M}$ and $\grave{\theta}_{9,i}^{M}$ denote the next and previous individuals neighboring to individual i according to the ascending ranking of objective function values, respectively.

Based on above operations, IMEAD is described in Algorithm 7. Its details are described here. Line 1 initializes $|\mathbb{X}|$, $\tilde{M}$, the initial weight vectors $\boldsymbol{\theta}_{7,i}^{M}$ and $\hat{g}$. Line 2 computes the Euclidean distance between any two weight vectors, and determines

Algorithm 7 IMEAD (Improved Multi-objective Evolutionary Algorithm based on Decomposition)

1: Initialize $|\mathbb{X}|$, $\tilde{M}$, the initial weight vectors $\boldsymbol{\theta}_{7,i}^{M}$ and $\hat{g}$
2: Compute Euclidean distances between any two weight vectors, and determine $\tilde{M}$ for each individual
3: Initialize $\mathbb{X}$ and determine $\boldsymbol{\theta}_{6}^{M}$
4: Determine non-dominated individuals in $\mathbb{X}$ and store them in $\breve{\Omega}$
5: Initialize θ_7 and θ_8
6: $g \leftarrow 1$
7: **for** $g \leq \hat{g}$ **do**
8: Generate a new individual $\mathbf{x}_i^1$
9: **for** $i \leftarrow 1$ to $|\mathbb{X}|$ **do**
10: Randomly select two individuals w_5 and w_6 in $\tilde{M}$
11: **if** $rand < \theta_7$ **then**
12: $\mathbf{x}_i^1 \leftarrow w_4 * \mathbf{x}_{w_5} + (1 - w_4) * \mathbf{x}_{w_6}$
13: **else**
14: $\mathbf{x}_i^1 \leftarrow \mathbf{x}_{w_7}$
15: **end if**
16: **end for**
17: Perform Gaussian mutation to produce $\mathbf{x}_i^1$ with (9.24)–(9.27)
18: Calculate objective function values of $\mathbf{x}_i^1$
19: Update $\boldsymbol{\theta}_{6}^{M}$
20: Update $\tilde{M}$ for each individual i with (9.28) and (9.29)
21: Update $\breve{\Omega}$
22: **end for**
23: Output $\breve{\Omega}$

$\theta_{1,i}^{M}$ for each individual i. Line 3 initializes $\mathbb{X}$ and determine $\boldsymbol{\theta}_{6}^{M}$. Line 4 determines non-dominated individuals in $\mathbb{X}$ and stores them in $\breve{\Omega}$. Line 5 initializes θ_7 and θ_8. Line 8 generates a new individual $\mathbf{x}_i^1$. Lines 9–16 randomly select two individuals w_5 and w_6 in $\tilde{M}$, and further perform the crossover operation based on (9.23). Line 17 performs the Gaussian mutation to produce $\mathbf{x}_i^1$ with (9.24)–(9.27). Line 18 calculates objective function values of $\mathbf{x}_i^1$. Line 19 updates $\boldsymbol{\theta}_{6}^{M}$. Line 20 updates $\tilde{M}$ for each individual i with (9.28) and (9.29). Line 21 updates $\breve{\Omega}$ and removes all individuals dominated by $\mathbf{x}_i^1$ from $\breve{\Omega}$.

Specifically, if the number of individuals in $\breve{\Omega}$ is less than its capacity limit, $\mathbf{x}_i^1$ is directly added to $\breve{\Omega}$ if no individuals in $\breve{\Omega}$ dominate $\mathbf{x}_i^1$. Otherwise, if the number of individuals in $\breve{\Omega}$ exceeds its capacity limit, according to the crowded degree evaluation

method based on crowded distance, an individual with the smallest crowded degree is selected and removed, and further replaced by $\mathbf{x}_i^1$. Finally, Line 23 outputs $\breve{\Omega}$.

9.4 Performance Evaluation

The proposed IMEAD is evaluated with the realistic data. It is realized with MATLAB R2017b, and executed on a server with 8-GB DDR4 memory and an Intel(R) Core(TM) i7-6700HQ CPU with 2.6 GHz.

9.4.1 Parameter Setting

This section uses real-life tasks in Google cluster[1], solar irradiance and wind speed[2] on May 1, 2011. Figure 9.3 illustrates arriving rates of tasks of three applications, which are sampled every 15 minutes, *i.e.*, L= 15 minutes. Here there are 96 time slots in total. Figures 9.4 and 9.5 show the solar irradiance and the wind speed in three CDCs. Figure 9.6 shows the electricity prices in three CDCs. According to the work in [24], the parameter setting of wind and solar energy is presented in Table 9.2. Besides, similar to the work in [18, 52], the parameter setting of three CDCs is presented in Table 9.3. In addition, the parameter setting well reflects the real-life characteristics of real large-scale data centers, *e.g.*, Google and Amazon.

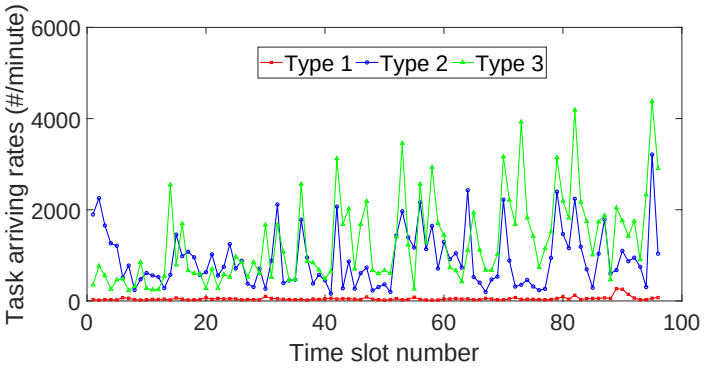


Figure 9.3 Task arriving rates of three applications.

This chapter chooses three Google CDCs located in Colorado (CDC 1), Texas (CDC 2) and Oklahoma (CDC 3), respectively. According to the work in [22],

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).
[2]https://midcdmz.nrel.gov (accessed on May 6, 2019).

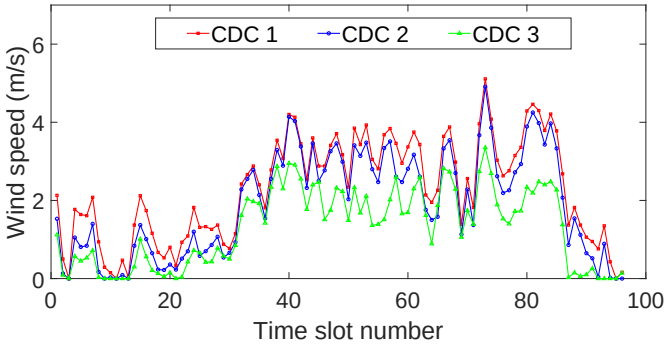


Figure 9.4 Wind speed of three CDCs.

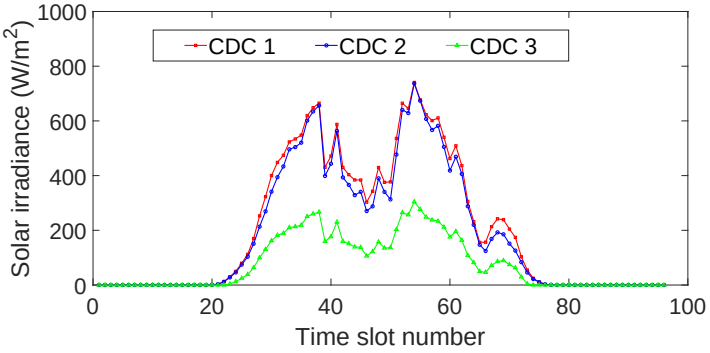


Figure 9.5 Solar irradiance of three CDCs.

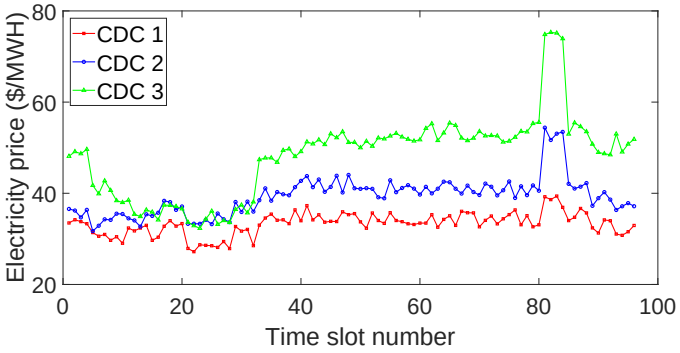


Figure 9.6 Electricity prices of three CDCs.

$\hat{N}_1$=$\hat{N}_2$=$\hat{N}_3$=2000, $\check{\Phi}$=500 (W) and $\hat{\Phi}$=2000 (W). In addition, $\overset{\bullet}{\mathbb{N}}_1$=0.01 (tasks/second), $\overset{\bullet}{\mathbb{N}}_2$=0.03 (tasks/second) and $\overset{\bullet}{\mathbb{N}}_3$=0.05 (tasks/second), respectively.

According to the work in [270, 271], the parameters of IMEAD are set as follows. The maximum number of iterations is 500, *i.e.*, $\hat{g}$=500. The population size is 100, *i.e.*, $|\mathbb{X}|$=100. $\hat{\theta}_7$=1, $\check{\theta}_7$=0.4, $\hat{\theta}_8$=0.25 and $\check{\theta}_8$=0.02. In addition, $\overset{0}{\gamma}_1$=$\overset{0}{\gamma}_2$=2.

Table 9.2 Parameter Setting of Wind and Solar Energy

	Wind energy			Solar energy	
	ϕ_{1c}	ϕ_{3c} (m^2)	ϕ_{2c} (kg/m^3)	ψ_{1c}	ψ_{2c} (m^2)
c=1	0.3	2000	1.0	0.2	1400
c=2	0.35	2500	1.25	0.25	1600
c=3	0.4	3000	1.5	0.3	1800

Table 9.3 Parameter Setting of Three CDCs

	$\dot{\mathbb{N}}_n$ (tasks/second)			$\hat{\mathbb{Q}}_n$		
	n=1	n=2	n=3	n=1	n=2	n=3
c=1	0.6	1.8	3.0	50	55	60
c=2	0.6	1.8	3.0	50	55	60
c=3	0.6	1.8	3.0	50	55	60

9.4.2 Experimental Results

To demonstrate the performance of IMEAD, this chapter further compares it with two widely used dominance-based algorithms including Strength Pareto Evolutionary Algorithm 2 (SPEA2) and an elitist based Non-dominated Sorting Genetic Algorithm (NSGA2) [272]. In this chapter, these two algorithms (SPEA2 and NSGA2) are selected as benchmark algorithms based on the following two reasons. First, they are currently recognized as fast state-of-the-art algorithms, both of which have been already improved with respect to efficiency and accuracy on the basis of their previous versions. Therefore, they are representative and have been used in many different areas [273, 274]. Second, the proposed IMEAD and these two algorithms have several similarities. All of them adopt the same initialization, crossover and mutation technologies to produce new individuals. In addition, all of them adopt the external elite solution set, and they have similar time complexity. Importantly, the performance of the proposed IMEAD can be well demonstrated by its comparison with these two algorithms. Consequently, this chapter selects them as the benchmark algorithms for comparison. The maximum number of iterations for three algorithms is set to 500. Their population sizes are 100. The crossover parameters of SPEA2 and NSGA2 are 0.98. The mutation parameters of SPEA2 and NSGA2 are 0.15 and

0.007, respectively. The setting of the above crossover and mutation parameters is determined optimally after a series of experiments, and the hardware configuration and software tool are the same as that of the proposed IMEAD. Three algorithms are summarized as follows:

1) SPEA2: SPEA2 [273] is widely used because it has several advantages. First, the fitness value evaluation mechanism of each individual is reasonable. It jointly considers the information about each individual in the evolutionary population, the domination in the external elite set, and the Euclidean distance between each individual and its neighboring individuals. Second, the clustering-based remove of individuals is simplified in the update process of the external elite set. Therefore, the distribution of the obtained Pareto front end is diverse. However, it has several drawbacks. First, the setting of the external elite set is not reasonable because some dominated individuals are still put into the external elite set and therefore, the elitism of the external elite set may be worsened. Second, its local search ability is not satisfying because it cannot guarantee to search within the neighboring areas of individuals where the crossover and mutation operations are conducted. Third, the crossover and mutation parameters are fixed and therefore, high-quality individuals are easy to be changed due to the mutation in the later stage of population evolution. Thus it suffers from premature problems when it is adopted to solve complex optimization problems. The complexity of the fitness calculation of SPEA2 is $O((2|\mathbb{X}|)^2)) + O((2|\mathbb{X}|)^2 log(2|\mathbb{X}|))$, *i.e.*, $O(|\mathbb{X}|^2 log|\mathbb{X}|)$. The complexity of the selection operation is $O((2|\mathbb{X}|)^2) + O((2|\mathbb{X}|)^2 log(2|\mathbb{X}|)) + O((2|\mathbb{X}|)^2)$, *i.e.*, $O(|\mathbb{X}|^2 log|\mathbb{X}|)$. Thus, the complexity of SPEA2 is $O(|\mathbb{X}|^2 log|\mathbb{X}|)$.

2) NSGA2: NSGA2 [274] is widely used to solve different kinds of multi-objective optimization algorithms because it has several advantages. First, it proposes the concept of crowded distance, and individuals in the Pareto front end are distributed evenly to increase the diversity of individuals. Second, it adopts a selection operator generating a mating pool that combines the offspring and parent populations. However, it has some drawbacks. First, the offspring and parent populations are ranked together in the next-generation evolutionary populations. Therefore, the ranking time of the Pareto dominance levels can be very long. Second, the external elite set is not adopted to keep the currently obtained non-dominated populations. In addition, the mutation and crossover parameters do not change and therefore, high-quality individuals cannot be kept in its optimization process. Thus, some high-quality solutions may be lost and NSGA2 has the problems of premature convergence. The complexity of the initial non-dominant sorting of population of NSGA2 is $O(\overset{o}{M}|\mathbb{X}|^2)$, and the complexity of its congestion sorting is $O(\overset{o}{M}|\mathbb{X}|^2)$. In the main loop, the fast non-dominant sorting is $O(\overset{o}{M}((2|\mathbb{X}|)^2))$, *i.e.*, $O(\overset{o}{M}|\mathbb{X}|^2)$, the complexity

Table 9.4 Execution Time with Different Population Sizes

Scale	1/4	1/2	1	2	4
NSGA2	0.2386	0.3840	0.7659	1.8938	4.9410
SPEA2	0.1560	0.2921	0.5070	2.2925	8.2673
IMEAD	**0.1119**	**0.1685**	**0.3035**	**0.7018**	**1.4804**

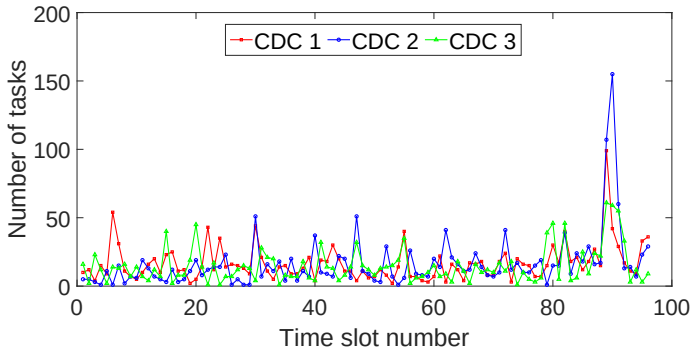


Figure 9.7 Number of type 1 tasks scheduled to three CDCs with IMEAD.

of its congestion sorting has the complexity of $O(\overset{o}{M}(2|\mathbb{X}|)log(2|\mathbb{X}|))$, *i.e.*, $O(\overset{o}{M}|\mathbb{X}|log|\mathbb{X}|)$, and the complexity of the elite population is $O((2|\mathbb{X}|)log(2|\mathbb{X}|))$, *i.e.*, $O(|\mathbb{X}|log|\mathbb{X}|)$. Thus, the complexity of NSGA2 is $O(\overset{o}{M}|\mathbb{X}|^2)$.

3) IMEAD: Different from SPEA2 and NSGA2, IMEAD has several advantages. First, the mutation and crossover parameters are changed dynamically to solve the problems of premature convergence. In addition, the global search ability is increased and the convergence speed is also improved. Better individuals are kept and new genes are imported when the entire population is trapped into local optima. Second, the Gaussian mutation mechanism is adopted to utilize the information of the current population to increase the diversity of populations. Third, a series of subproblems are solved to optimize the whole problem, and IMEAD has lower computational complexity. Fourth, the external elite set is adopted and the crowded distance-based congestion evaluation method is designed to guarantee the diversity of all population. The complexity of the neighboring solution update of IMEAD is $O(|\mathbb{X}|\tilde{M})$, the complexity of the $\breve{\Omega}$ update is $O(\overset{o}{M}2|\mathbb{X}|^2)$, *i.e.*, $O(\overset{o}{M}|\mathbb{X}|^2)$, and the complexity of the congestion sorting is $O(\overset{o}{M}|\mathbb{X}|log|\mathbb{X}|)$. Thus, the complexity of IMEAD is $O(\overset{o}{M}|\mathbb{X}|^2)$.

To further evaluate the performance of IMEAD, the scalability of IMEAD, SPEA2 and NSGA2 with respect to different population sizes is verified. As is shown in Table 9.4, IMEAD's execution time is less than those of SPEA2 and NSGA2 for each population size. Figures 9.7–9.9 show the number of tasks scheduled to three

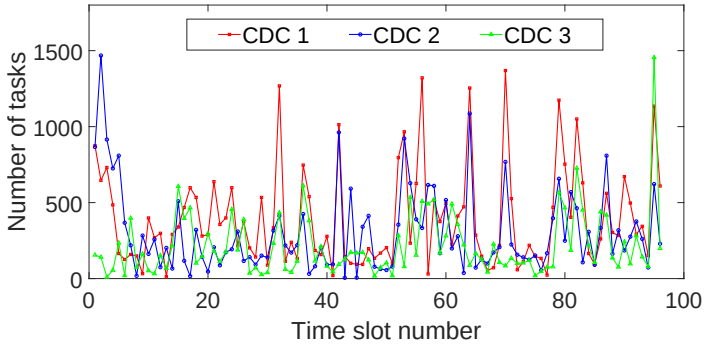


Figure 9.8 Number of type 2 tasks scheduled to three CDCs with IMEAD.

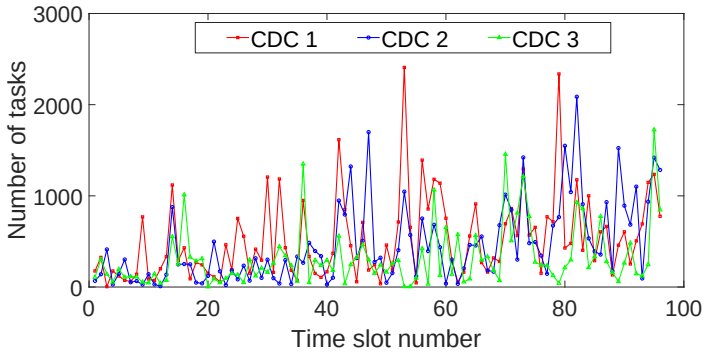


Figure 9.9 Number of type 3 tasks scheduled to three CDCs with IMEAD.

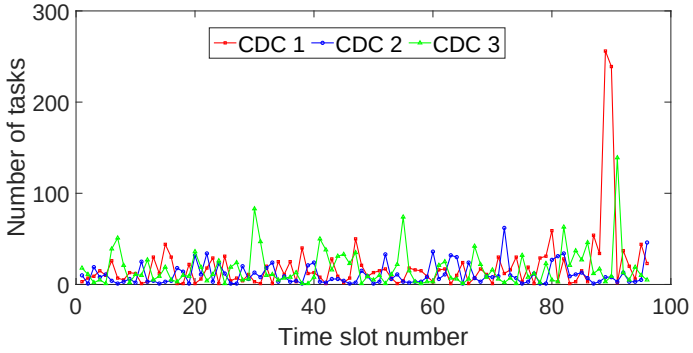


Figure 9.10 Number of type 1 tasks scheduled to three CDCs with SPEA2.

CDCs with IMEAD. It is observed in Figure 9.7 that the total number of type 1 tasks scheduled to three CDCs equals the number of arriving type 1 tasks. Similarly, Figures 9.10–9.15 show the number of tasks scheduled to three CDCs with SPEA2 and NSGA2, respectively. It is also observed in Figures 9.10–9.15 that the total number of tasks of types 2 and 3 scheduled to three CDCs equals the number of arriving tasks of types 2 and 3, respectively.

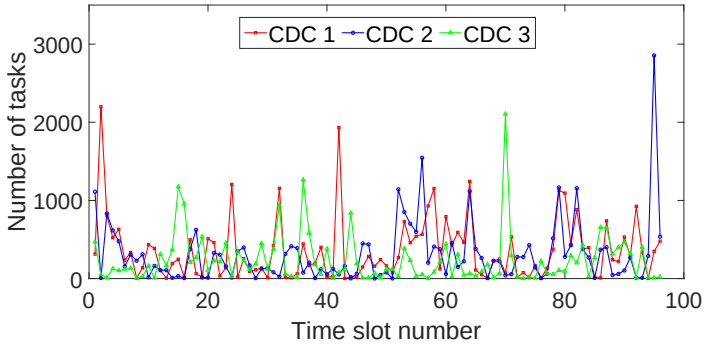


Figure 9.11 Number of type 2 tasks scheduled to three CDCs with SPEA2.

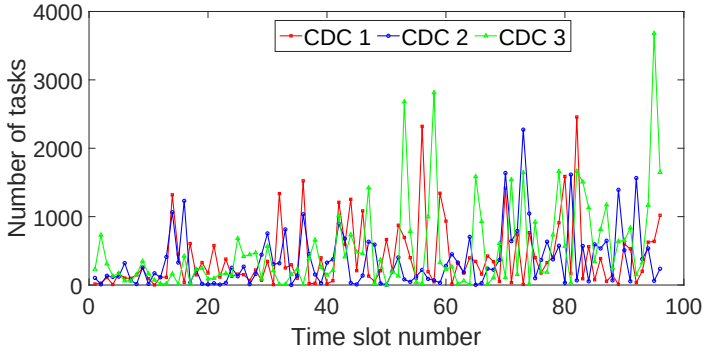


Figure 9.12 Number of type 3 tasks scheduled to three CDCs with SPEA2.

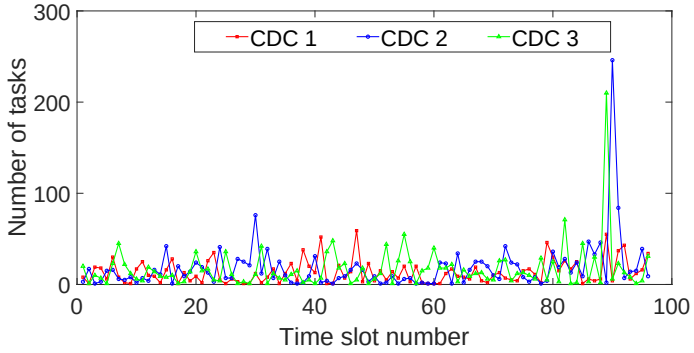


Figure 9.13 Number of type 1 tasks scheduled to three CDCs with NSGA2.

Figures 9.16–9.18 show the number of switched-on servers in each CDC with IMEAD. It is observed that the number of switched-on servers in each CDC is less than or equal to its corresponding total number of available servers. In addition, the number of switched-on servers in each CDC is consistent with the number of tasks scheduled to it. As is shown in Figures 9.25–9.27, it is also worth noting that in time slot 40, the amount of green energy is sufficient to execute all arriving tasks.

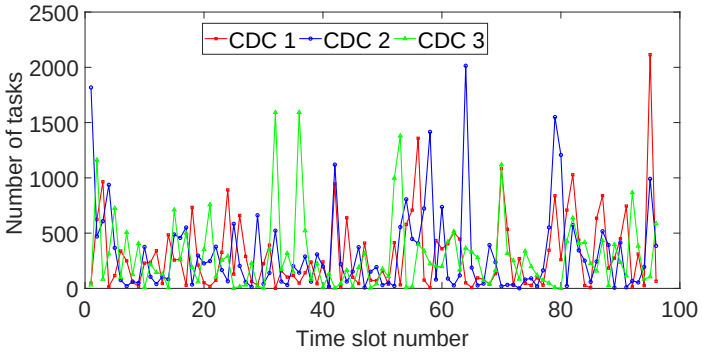


Figure 9.14 Number of type 2 tasks scheduled to three CDCs with NSGA2.

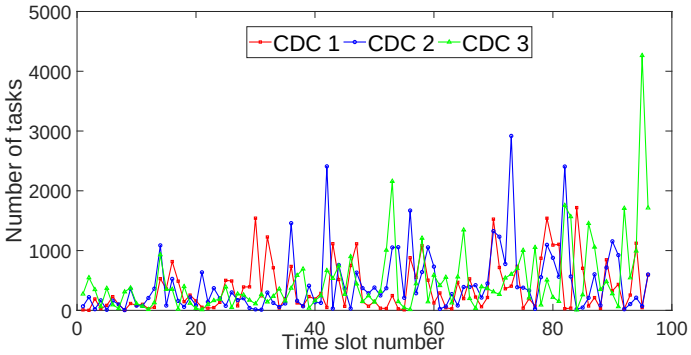


Figure 9.15 Number of type 3 tasks scheduled to three CDCs with NSGA2.

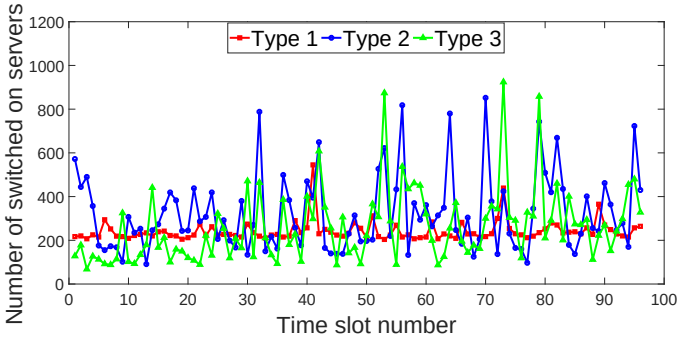


Figure 9.16 Number of switched-on servers in CDC 1 with IMEAD.

Therefore, the amount of power grid energy consumed by tasks is zero, and the energy cost is also zero because the cost of green energy is zero. Figures 9.19–9.24 show that SPEA2 and NSGA2 also achieve similar results.

Figure 9.28 shows the execution time required by each iteration of IMEAD, SPEA2 and NSGA2 in each time slot, respectively. It is observed that IMEAD is much faster than SPEA2 and NSGA2. Figure 9.29 shows the execution time of

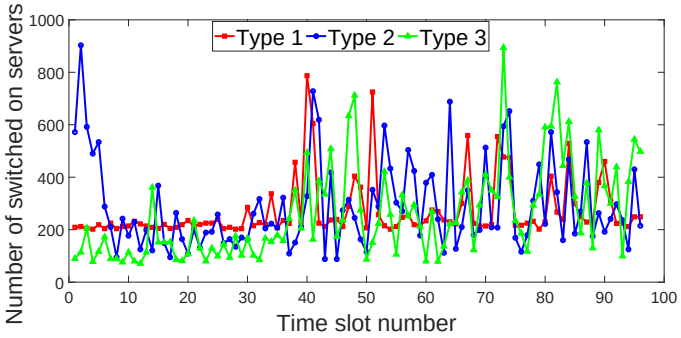


Figure 9.17 Number of switched-on servers in CDC 2 with IMEAD.

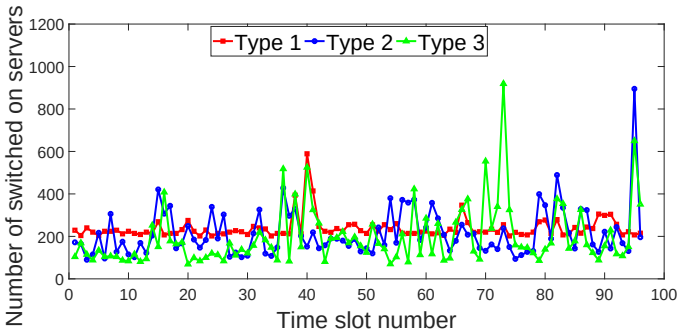


Figure 9.18 Number of switched-on servers in CDC 3 with IMEAD.

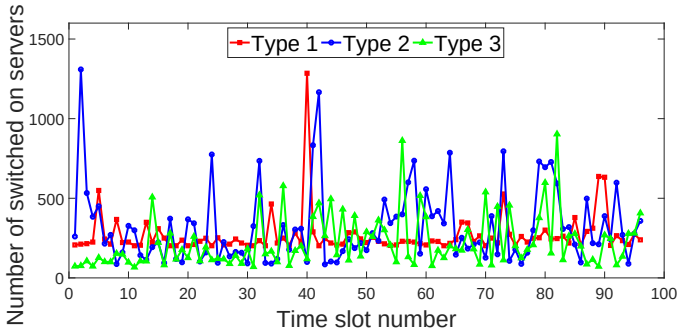


Figure 9.19 Number of switched-on servers in CDC 1 with SPEA2.

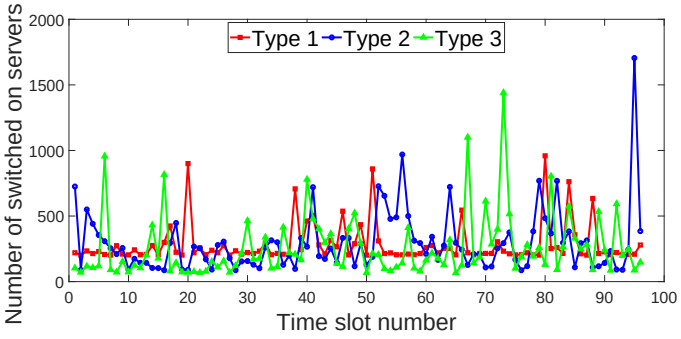


Figure 9.20 Number of switched-on servers in CDC 2 with SPEA2.

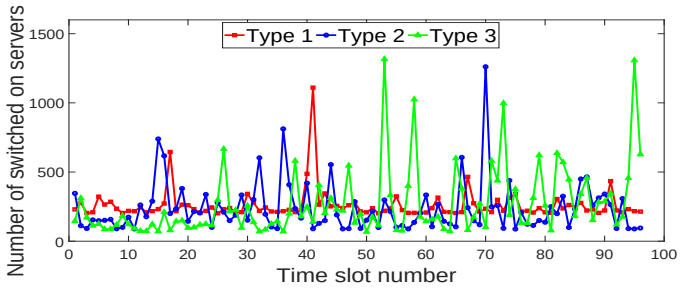


Figure 9.21 Number of switched-on servers in CDC 3 with SPEA2.

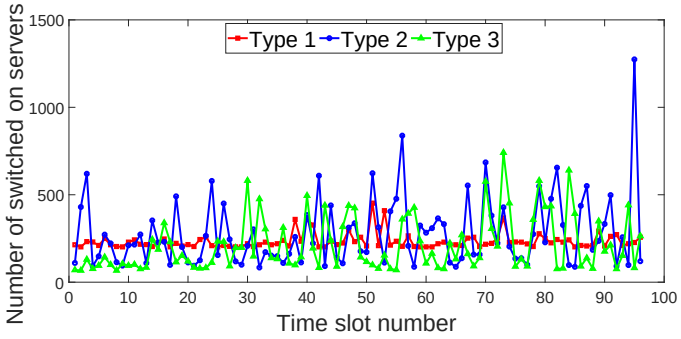


Figure 9.22 Number of switched-on servers in CDC 1 with NSGA2.

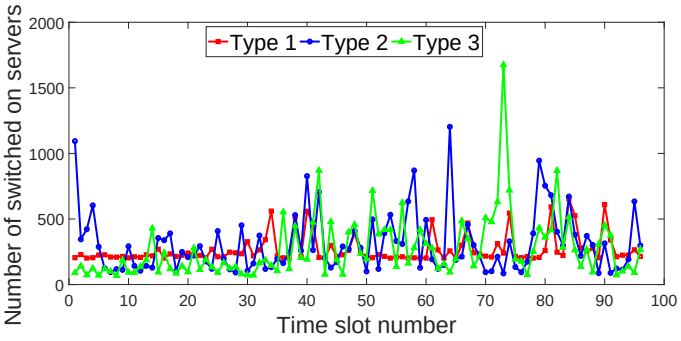


Figure 9.23 Number of switched-on servers in CDC 2 with NSGA2.

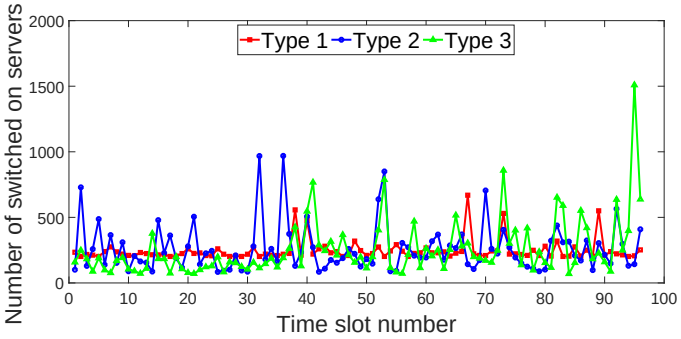


Figure 9.24 Number of switched-on servers in CDC 3 with NSGA2.

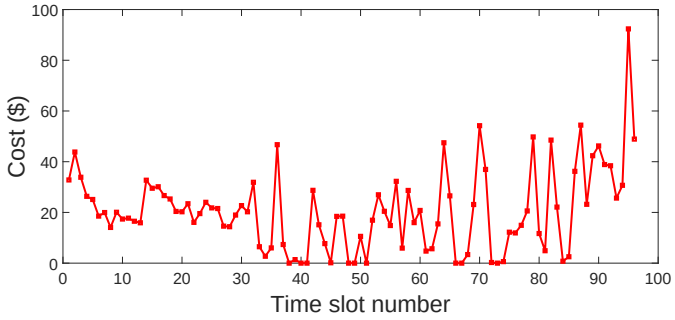


Figure 9.25 Cost of IMEAD.

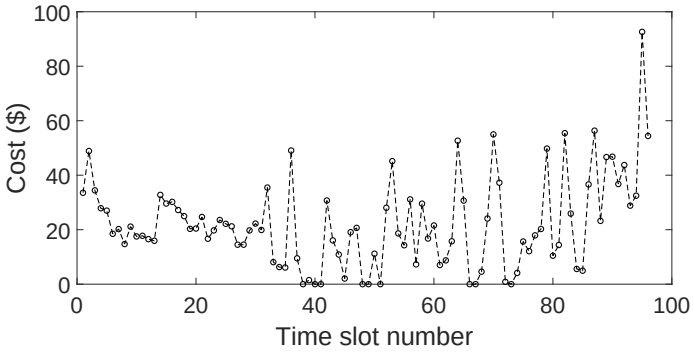


Figure 9.26 Cost of NSGA2.

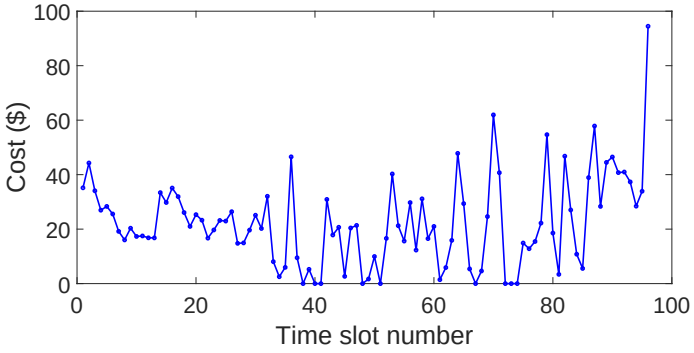


Figure 9.27 Cost of SPEA2.

IMEAD, SPEA2 and NSGA2, respectively. It is clearly observed that IMEAD is 40.14% and 60.37% faster than SPEA2 and NSGA2, respectively.

Figure 9.30 shows the revenue, cost and profit in each day with IMEAD, SPEA2 and NSGA2, respectively. It is clearly observed that IMEAD achieves slightly higher revenue, 8.00% and 6.80% lower cost, than SPEA2 and NSGA2, respectively. Therefore, it is observed that IMEAD achieves 1.63% and 1.37% higher profit than

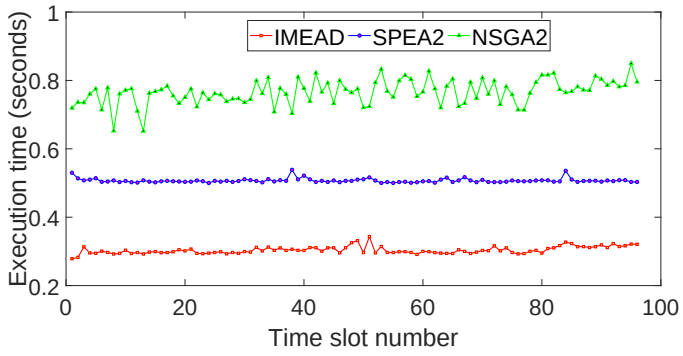


Figure 9.28 Execution time of IMEAD, SPEA2 and NSGA2.

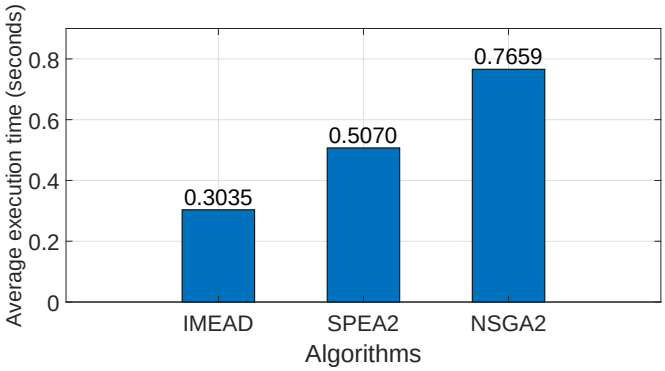


Figure 9.29 Average execution time of IMEAD, SPEA2 and NSGA2.

SPEA2 and NSGA2, respectively. The reason is that IMEAD prefers to use green energy and purchases the energy from local power grid only when the amount of wind and solar energy is not sufficient. The reason is explained as follows. It is assumed that the cost of green energy is zero after the renewable facilities have been installed in CDCs for all three algorithms. Thus, the usage of green energy and the cost of electricity purchased from the power grid directly affect the optimization result of the energy cost. If the green energy cannot be fully used and tasks of applications cannot be optimally allocated, the energy cost is higher.

Figure 9.30 shows that IMEAD's cost is lower and its profit is higher than those of SPEA2 and NSGA2, respectively. The reason is that IMEAD makes full use of green energy and provides better scheduling of different tasks. Consequently, the green energy is consumed sufficiently and the cost of the CDC provider is reduced accordingly. The tasks of each application are allocated optimally for the

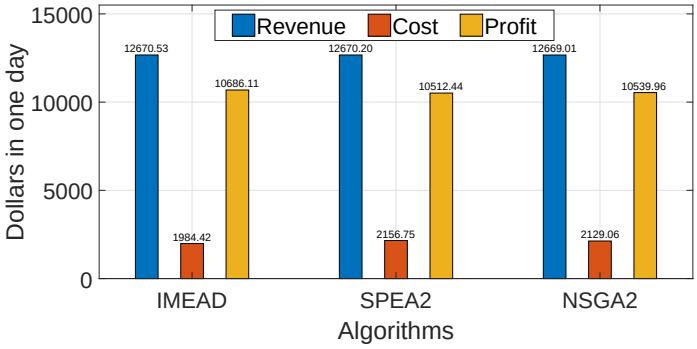


Figure 9.30 Revenue, cost and profit in one day with IMEAD, SPEA2 and NSGA2.

aim of maximizing the profit, *i.e.*, minimizing the cost and maximizing the revenue. Specifically, given the finally optimal decision variable $\boldsymbol{x}$, the rate at which tasks are removed from the queue of application n and handled by CDC c is set as $\mu_\tau^{c,n}$. Then, $\lambda_\tau^{c,n}L$ tasks of application n are scheduled to CDC c. In this way, tasks of each application are optimally allocated.

9.5 Summary

Green cloud data centers (CDCs) require a great amount of energy to run tasks from global users. Users' tasks contribute the revenue to CDC providers. The spatial differences in many factors, *e.g.*, prices of electricity, availability of renewable power generation and service level agreements bring a big challenge to jointly optimize providers' revenue and their energy cost. A constrained bi-objective optimization problem is for the first time formulated and solved with an Improved Multiobjective Evolutionary Algorithm based on Decomposition. The obtained result determines the optimal tradeoff between maximizing the revenue of CDC providers and minimizing their energy cost by taking the advantages of various spatial differences in CDCs while meeting delay constraints of tasks of applications. Real-life data-driven simulation results prove that it increases the profit of CDC providers and reduces the convergence time compared with two typical multi-objective optimization algorithms.

CHAPTER 10

FINE-GRAINED SPATIAL TASK SCHEDULING FOR HETEROGENEOUS APPLICATIONS IN DISTRIBUTED GREEN CLOUDS

This chapter presents the details of the proposed Fine-grained Spatial Task Scheduling (FSTS) algorithm, and it is organized as follows. Section 10.1 introduces the framework of multiple green cloud data centers (CDCs), and formulates a single-objective constrained optimization problem for FSTS. It adopts a $G/G/1$ queuing system to analyze the performance of servers in distributed CDCs. Section 10.2 describes the proposed Bees Algorithm based on Simulated annealing (BAS) to solve the problem for obtaining a close-to-optimal solution such that the energy cost is minimized. Real-life data-driven simulation experiments are conducted to evaluate the proposed FSTS in Section 10.3. Section 10.4 provides the appendix. Finally, Section 10.5 concludes this chapter.

10.1 Problem Formulation

This section formulates a constrained optimization problem for FSTS. The illustrative framework of multiple CDCs is similar to Figure 7.1 in Chapter 7. Let $\mathcal{N}^C$ denote the number of CDCs. Users' arriving tasks are sent through electronic devices, *e.g.*, smart phones, laptops, computers and servers to $\mathcal{N}^C$ CDCs, and they are scheduled by *Task Scheduler* based on an FCFS manner.

FSTS is periodically executed in *Task Scheduler* to minimize the energy cost by intelligently scheduling tasks of each application among multiple centers, and optimally determining the running speed of each server and the number of switched-on servers in each CDC. For clarity, main notations in this chapter are summarized in Tables 10.1 and 10.2.

Table 10.1 Main Symbols-Part 1

Notations	Definition
$\mathcal{N}^C$	Number of CDCs
$\mathcal{N}^A$	Number of applications
$\hat{N}_{c,n}$	Number of homogeneous servers for application n in CDC c
$\overset{o}{N}_{\tau,c,n}$	Number of switched-on servers for application n in CDC c in time slot τ
$\mathcal{T}_\tau^{c,n}$	Interarrival time for each server of application n in CDC c in time slot τ
$\hat{\sigma}_{\tau,c,n}$	Variance of $\mathcal{T}_\tau^{c,n}$
$\bar{t}_\tau^{c,n}$	Mean of $\mathcal{T}_\tau^{c,n}$
$\lambda_\tau^{c,n}$	Task arriving rate of application n in CDC c in time slot τ
r_c^n	Random size of each task of application n scheduled to each server in CDC c
$\breve{\sigma}_{c,n}$	Variance of r_c^n
$\bar{r}_c^n$	Mean of r_c^n
$t_\tau^{c,n}$	Running time of each task on a server of application n in CDC c in time slot τ
$\varrho_\tau^{c,n}$	Running speed of each server of application n in CDC c in time slot τ
$\bar{t}_\tau^{c,n}$	Variance of $t_\tau^{c,n}$
$\tilde{\sigma}_{\tau,c,n}$	Mean of $t_\tau^{c,n}$
$\hat{\varrho}_c^n$	Maximum running speed of each server of application n in CDC c
λ_τ^n	Arriving rate of tasks of application n in time slot τ
$\lambda_\tau^{c,n}$	Task arriving rate of application n in CDC c in time slot τ
$T_\tau^{c,n}$	Response time of tasks of application n in CDC c in time slot τ
L	Time slot length

10.1.1 Task Response Time Model

This chapter presents a task response time model. Let $\mathcal{N}^A$ denote the number of applications. In CDC c ($1 \leq c \leq \mathcal{N}^C$), there are $\hat{N}_{c,n}$ homogeneous servers for application n ($1 \leq n \leq \mathcal{N}^A$). Let $\overset{o}{N}_{\tau,c,n}$ denote the number of switched-on servers for application n in CDC c in time slot τ. Thus, $\overset{o}{N}_{\tau,c,n}$ must not exceed $\hat{N}_{c,n}$, *i.e.*,

$$0 \leq \overset{o}{N}_{\tau,c,n} \leq \hat{N}_{c,n}, \ \overset{o}{N}_{\tau,c,n} \in N^+ \tag{10.1}$$

This chapter uses a $G/G/1$ queuing model to analyze the performance of each switched-on server. The execution time and interarrival time of each task have arbitrary probability distributions in each switched-on server. Let $\mathcal{T}_\tau^{c,n}$ denote the interarrival time for each server of application n in CDC c in time slot τ. Its variance

Table 10.2 Main Symbols-Part 2

Notations	Definition
$\hat{T}_n$	Response time limit of application n
$P_\tau^{c,n}$	Amount of power consumed by each server of application n in CDC c in time slot τ
$U_\tau^{c,n}$	Supply voltage of each server of application n in CDC c in time slot τ
$\chi_\tau^{c,n}$	Clock frequency of each server of application n in CDC c in time slot τ
$\check{\Phi}_n^c$	Amount of power consumption of each idle server of application n in CDC c
P_τ^c	Total power consumed by CDC c in time slot τ
$\hat{\mathbb{E}}_c$	Maximum amount of energy in CDC c
$\widetilde{P}_{\tau,c}$	Amount of wind power consumed by tasks of each application in CDC c in time slot τ
$\mathring{P}_{\tau,c}$	Amount of solar power consumed by tasks of each application in CDC c in time slot τ
ϕ_{1c}	Wind-to-electricity conversion rate of CDC c
ϕ_{2c}	On-site air density of CDC c
ϕ_{3c}	Rotor area of wind turbines of CDC c
$\phi_{\tau,4c}$	Wind speed of CDC c in time slot τ
$\psi_{\tau,3c}$	Solar irradiance of CDC c in time slot τ
ψ_{2c}	Active irradiance area of solar panels of CDC c
ψ_{1c}	Solar-irradiance-to-electricity conversion rate of CDC c
p_τ^c	Price of power grid of CDC c in time slot τ
E_τ^c	Energy cost of CDC c in time slot τ

is $\hat{\sigma}_{\tau,c,n}$ and its mean is $\bar{\mathcal{T}}_\tau^{c,n}$. It is worth noting that $\hat{\sigma}_{\tau,c,n}$ and $\bar{\mathcal{T}}_\tau^{c,n}$ can be set by analyzing tasks in real-life data, *e.g.*, Google cluster trace data. Let $\lambda_\tau^{c,n}$ denote the task arriving rate of application n in CDC c in time slot τ. Thus, $\lambda_\tau^{c,n}$ is obtained as $\lambda_\tau^{c,n}=\frac{1}{\bar{\mathcal{T}}_\tau^{c,n}}$. Let r_c^n denote the random size of each task of application n scheduled to each server in CDC c. The probability distribution of r_c^n can be arbitrary. The variance of r_c^n is $\breve{\sigma}_{c,n}$ and its mean is $\bar{r}_c^n$.

Let $t_\tau^{c,n}$ denote the running time of each task on a server of application n in CDC c in time slot τ. Let $\varrho_\tau^{c,n}$ denote the running speed of each server of application n in CDC c in time slot τ. Thus, $t_\tau^{c,n}=r_c^n/\varrho_\tau^{c,n}$. Let $\bar{t}_\tau^{c,n}$, $\tilde{\sigma}_{\tau,c,n}$ and $\Omega_\tau^{c,n}$ denote the mean, variance and variation coefficient of $t_\tau^{c,n}$, respectively. Then,

$$\bar{t}_\tau^{c,n}=\frac{\bar{r}_c^n}{\varrho_\tau^{c,n}} \tag{10.2}$$

$$\tilde{\sigma}_{\tau,c,n}=\frac{\breve{\sigma}_{c,n}}{\left(\varrho_{\tau}^{c,n}\right)^2} \tag{10.3}$$

$$\Omega_{\tau}^{c,n}=\frac{\sqrt{\tilde{\sigma}_{\tau,c,n}}}{\bar{t}_{\tau}^{c,n}} \tag{10.4}$$

It is assumed that servers in the same CDC are homogeneous while those in different CDCs are heterogeneous. Let $\hat{\varrho}_c^n$ denote the maximum running speed of each server of application n in CDC c. Consequently, $\varrho_{\tau}^{c,n}$ needs to be less than or equal to $\hat{\varrho}_c^n$, *i.e.*,

$$0\leq\varrho_{\tau}^{c,n}\leq\hat{\varrho}_c^n \tag{10.5}$$

The running speed of each server of application n in CDC c in time slot τ needs to be large enough to execute the tasks scheduled to that server. Specifically, $\varrho_{\tau}^{c,n}$ cannot be less than $\frac{\bar{r}_c^n}{\bar{T}_{\tau}^{c,n}}$, *i.e.*,

$$\varrho_{\tau}^{c,n}>\frac{\bar{r}_c^n}{\bar{T}_{\tau}^{c,n}} \tag{10.6}$$

In addition, the arriving rate of tasks of application n in time slot τ is denoted by λ_{τ}^n. Let $\lambda_{\tau}^{c,n}$ denote the arriving rate of tasks of application n in CDC c in time slot τ. Then, λ_{τ}^n is the sum of task arriving rates of application n in all CDCs in time slot τ. Therefore, (10.7) is obtained.

$$\lambda_{\tau}^n=\sum_{c=1}^{\mathcal{N}^C}\lambda_{\tau}^{c,n} \tag{10.7}$$

Then, let $T_{\tau}^{c,n}$ denote the response time of tasks of application n in CDC c with $\overset{o}{N}_{\tau,c,n}$ servers in time slot τ. L denotes the length of each time slot. Consequently,

$$T_{\tau}^{c,n}=\frac{\bar{r}_c^n}{\varrho_{\tau}^{c,n}}+\left((\bar{r}_c^n)^2+\breve{\sigma}_{c,n}\right)\Delta_{\tau,c,n}^{16} \tag{10.8}$$

$$\Delta_{\tau,c,n}^{16}=\frac{\hat{\sigma}_{\tau,c,n}(s_{\tau}^{c,n})^2+\breve{\sigma}_{c,n}}{2\varrho_{\tau}^{c,n}(\bar{T}_{\tau}^{c,n}\varrho_{\tau}^{c,n}-\bar{r}_c^n)\left((\bar{T}_{\tau}^{c,n})^2(\varrho_{\tau}^{c,n})^2+\breve{\sigma}_{c,n}\right)}$$

where $\bar{r}_c^n=\frac{\lambda_\tau^{c,n}L}{\overset{o}{N}_{\tau,c,n}}$ and $\bar{\mathcal{T}}_\tau^{c,n}=\frac{\overset{o}{N}_{\tau,c,n}}{\lambda_\tau^{c,n}}$. $\breve{\sigma}_{c,n}$ is obtained by calculating the variance of the data including r_1^1, r_1^2, $\cdots$, and $r_{\mathcal{N}^C}^{\mathcal{N}^A}$. $\hat{\sigma}_{\tau,c,n}$ is obtained by calculating the variance of the data including $\mathcal{T}_\tau^{1,1}$, $\mathcal{T}_\tau^{1,2}$, $\cdots$, and $\mathcal{T}_\tau^{\mathcal{N}^C,\mathcal{N}^A}$.

It is worth noting that the derivation of (10.8) is presented in Appendix. Let $\hat{T}_n$ denote the specified response time limit of application n. Then, $T_\tau^{c,n}$ needs to be less than or equal to $\hat{T}_n$, *i.e.*,

$$T_\tau^{c,n}\leq\hat{T}_n \tag{10.9}$$

10.1.2 Energy Cost Model

This chapter introduces the energy cost model of CDCs. The power consumption accounts for a majority of energy cost of CDCs [18]. Let $P_\tau^{c,n}$ denote the amount of power consumed by each server of application n in CDC c in time slot τ. Let $U_\tau^{c,n}$ and $\chi_\tau^{c,n}$ denote the supply voltage and clock frequency of each server of application n in CDC c.

According to the work in [244], $U_\tau^{c,n}\propto(\chi_\tau^{c,n})^{\overset{0}{\gamma}_{9,c,n}}$ for a constant $\overset{0}{\gamma}_{9,c,n}>0$ in the perfect case. In addition, $\varrho_\tau^{c,n}$ is linearly proportional to $\chi_\tau^{c,n}$, *i.e.*, $\varrho_\tau^{c,n}\propto\chi_\tau^{c,n}$. Therefore, similar to the work in [36], $U_\tau^{c,n}=\overset{0}{\gamma}_{10,c,n}(\chi_\tau^{c,n})^{\overset{0}{\gamma}_{9,c,n}}$ and $\varrho_\tau^{c,n}=\overset{0}{\gamma}_{11,c,n}\chi_\tau^{c,n}$. $\overset{0}{\gamma}_{10,c,n}$ and $\overset{0}{\gamma}_{11,c,n}$ are constants for each server of application n in CDC c. Consequently, $P_\tau^{c,n}$ is obtained as:

$$\begin{aligned}
P_\tau^{c,n}&=\vartheta_c^n\omega_c^n\left(U_\tau^{c,n}\right)^2\chi_\tau^{c,n}\\
&=\vartheta_c^n(\overset{0}{\gamma}_{10,c,n})^2\omega_c^n\left(\chi_\tau^{c,n}\right)^{2\overset{0}{\gamma}_{9,c,n}+1}\\
&=\vartheta_c^n(\overset{0}{\gamma}_{10,c,n})^2\omega_c^n\frac{\left(\varrho_\tau^{c,n}\right)^{2\overset{0}{\gamma}_{9,c,n}+1}}{\left(\overset{0}{\gamma}_{11,c,n}\right)^{2\overset{0}{\gamma}_{9,c,n}+1}}\\
&=\daleth_c^n\left(\varrho_\tau^{c,n}\right)^{\Psi_c^n}
\end{aligned} \tag{10.10}$$

where ϑ_c^n, ω_c^n, $U_\tau^{c,n}$ and $\chi_\tau^{c,n}$ denote the activity factor, loading capacitance, supply voltage, and clock frequency of each server of application n in CDC c. In addition, $\daleth_c^n = \vartheta_c^n (\overset{0}{\gamma}_{10,c,n})^2 \omega_c^n / (\overset{0}{\gamma}_{11,c,n})^{2\overset{0}{\gamma}_{9,c,n}+1}$ and $\Psi_c^n = 2\overset{0}{\gamma}_{9,c,n}+1$.

Note that each idle server also consumes some amount of power because of the dissipation of static and short-circuit power, and other leakage and wasted power [275]. $\check{\Phi}_n^c$ denotes the amount of power consumption of each idle server of application n in CDC c. Besides, P_τ^c denotes the total power consumed by CDC c in time slot τ. Similar to the work in [24, 246], P_τ^c is given as:

$$P_\tau^c = \sum_{n=1}^{\mathcal{N}^A} P_\tau^{c,n} = \sum_{n=1}^{\mathcal{N}^A} \left(\overset{o}{N}_{\tau,c,n} \left(\daleth_c^n \left(\varrho_\tau^{c,n} \right) \Psi_c^n + \check{\Phi}_n^c \right) \right) \tag{10.11}$$

Then, let $\hat{\mathbb{E}}_c$ denote the maximum amount of energy in CDC c. Based on (10.11), the total energy consumed by CDC c in time slot τ is $P_\tau^c L$, which must be less than or equal to $\hat{\mathbb{E}}_c$, *i.e.*,

$$\sum_{n=1}^{\mathcal{N}^A} \left(\overset{o}{N}_{\tau,c,n} \left(\daleth_c^n \left(\varrho_\tau^{c,n} \right) \Psi_c^n + \check{\Phi}_n^c \right) \right) L < \hat{\mathbb{E}}_c \tag{10.12}$$

Let $\widetilde{P}_{\tau,c}$ and $\overset{\circ}{P}_{\tau,c}$ denote the amount of wind and solar power consumed by tasks in CDC c in time slot τ, respectively. According to the work in [246],

$$\widetilde{P}_{\tau,c} = \frac{1}{2} \phi_{1c} \phi_{2c} \phi_{3c} \left(\phi_{\tau,4c} \right)^3 \tag{10.13}$$

where ϕ_{1c}, ϕ_{2c}, ϕ_{3c} and $\phi_{\tau,4c}$ are the wind-to-electricity conversion rate, on-site air density, rotor area of wind turbines, and wind speed of CDC c in time slot τ, respectively. Similarly, the following equation is obtained.

$$\overset{\circ}{P}_{\tau,c} = \psi_{1c} \psi_{2c} \psi_{\tau,3c} \tag{10.14}$$

where $\psi_{\tau,3c}$ denotes the solar irradiance in time slot τ, ψ_{2c} denotes the active irradiance area of solar panels, and ψ_{1c} denotes the solar-irradiance-to-electricity conversion rate

of CDC c. Let p_τ^c denote the price of power grid of CDC c in time slot τ, and let $f_{22}^{c,*}$ denote the energy cost of CDC c. Then,

$$f_{22}^{c,*}=\left(\max\left(P_\tau^c-\overset{\circ}{P}_{\tau,c}-\widetilde{P}_{\tau,c},0\right)\right)p_\tau^c L \tag{10.15}$$

Then, the energy cost of a CDC provider is obtained as:

$$\begin{aligned} f_{22}&=\sum_{c=1}^{\mathcal{N}^C} f_{22}^{c,*}=\sum_{c=1}^{\mathcal{N}^C}\left(\max\left(P_\tau^c-\overset{\circ}{P}_{\tau,c}-\widetilde{P}_{\tau,c},0\right)\right)p_\tau^c L \\ &=\sum_{c=1}^{\mathcal{N}^C}\left(\max\left(\sum_{n=1}^{\mathcal{N}^A}(\overset{o}{N}_{\tau,c,n}\left(\urcorner_c^n(\varrho_\tau^{c,n})\Psi_c^n+\breve{\Phi}_n^c\right))-\overset{\circ}{P}_{\tau,c}-\widetilde{P}_{\tau,c},0\right)\right)p_\tau^c L \end{aligned} \tag{10.16}$$

The objective is to minimize f_{22}, *i.e.*,

$$\underset{\overset{o}{N}_{\tau,c,n},\lambda_\tau^{c,n},\varrho_\tau^{c,n}}{\mathbf{Min}} \quad \{f_{22}\} \tag{10.17}$$

Then, according to (10.1), (10.5), (10.6), (10.7), (10.9) and (10.12), the constrained optimization problem is formulated as:

$$\underset{\overset{o}{N}_{\tau,c,n},\lambda_\tau^{c,n},\varrho_\tau^{c,n}}{\mathbf{Min}} \quad \{f_{22}\}$$

subject to

$$0\leq\overset{o}{N}_{\tau,c,n}\leq\hat{N}_{c,n} \tag{10.18}$$

$$0\leq\varrho_\tau^{c,n}\leq\hat{\varrho}_c^n \tag{10.19}$$

$$\varrho_\tau^{c,n}>\frac{\bar{r}_c^n}{\bar{\mathcal{T}}_\tau^{c,n}} \tag{10.20}$$

$$\lambda_\tau^n=\sum_{c=1}^{\mathcal{N}^C}\lambda_\tau^{c,n} \tag{10.21}$$

$$T_\tau^{c,n}\leq\hat{T}_n \tag{10.22}$$

$$\sum_{n=1}^{\mathcal{N}^A}\left(\overset{o}{N}_{\tau,c,n}\left(\urcorner_c^n\left(\varrho_\tau^{c,n}\right)\Psi_c^n+\check{\Phi}_n^c\right)\right)L<\hat{\mathbb{E}}_c \tag{10.23}$$

$$\overset{o}{N}_{\tau,c,n}\in N^+,\varrho_\tau^{c,n}\geq 0,\lambda_\tau^{c,n}\geq 0 \tag{10.24}$$

$$\lambda_\tau^{c,n}=0,\text{ if }\overset{o}{N}_{\tau,c,n}=0 \tag{10.25}$$

$$\lambda_\tau^{c,n}>0,\text{ if }\overset{o}{N}_{\tau,c,n}>0 \tag{10.26}$$

Then, its solution algorithm is introduced next.

10.2 Bees Algorithm based on Simulated annealing

f_{22} is nonlinear with respect to $\overset{o}{N}_{\tau,c,n}$, $\lambda_\tau^{c,n}$, and $\varrho_\tau^{c,n}$. Therefore, its optimization problem is a constrained and nonlinear. Let $\boldsymbol{x}$ denote the vector of decision variables including $\overset{o}{N}_{\tau,c,n}$, $\lambda_\tau^{c,n}$, and $\varrho_\tau^{c,n}$. To effectively solve it, this chapter adopts a penalty function method to transform it into an unconstrained problem, *i.e.*,

$$\underset{\boldsymbol{x}}{\mathbf{Min}}\left\{\widetilde{f_{22}}=\overset{\infty}{\mathcal{N}}\mho+f_{22}\right\} \tag{10.27}$$

In (10.27), $\widetilde{f_{22}}$ denotes an augmented objective function and $\overset{\infty}{\mathcal{N}}$ a positive number that is ten times larger than the upper bound of f_{22}. $\mho$ denotes the penalty corresponding to all constraints, and it is calculated with (3.21) in Chapter 3.

Currently, there are many classical algorithms, *e.g.*, dynamic programming [276], Lagrange multiplier [277], Branch and bound [278], and Bucket elimination [279], to solve it. Nevertheless, they usually need the first-order or second-order derivatives of the objective functions. They are effective to solve some constrained optimization problems with certain required mathematical structures [31]. Yet, their optimization processes are difficult and their final solutions are often unsatisfied.

To tackle such shortcomings, many studies design meta-heuristic algorithms that obtain near-optimal solutions to the constrained optimization problem in reasonable execution time. They have several advantages including easy implementation, robustness, handling complex nonlinearities and discontinuities of objective functions.

In the category of intelligent optimization tools, swarm-based optimization algorithms (SOAs) are search ones that can efficiently locate relatively good solutions [280]. SOAs are inspired by methods in nature to provide an effective search towards an optimal solution. SOAs differ from direct search algorithms, *e.g.*, hill climbing, because SOAs adopt a population of solutions for each iteration instead of one single solution. These solutions are updated iteration by iteration, and and output when some termination conditions are met.

Among SOAs, Bees Algorithm (BeesA) is an optimization one inspired by the foraging behavior of natural honey bees. It is commonly applied due to its easy implementation and quick convergence [281]. In BeesA, a colony of honey bees extend themselves over long distances in different directions. Flower patches with more pollen or nectar that is collected with less effort should attract more bees, whereas those with less pollen or nectar attract fewer ones. Scout bees randomly search from one patch to another, and evaluate different patches. Then, their pollen or nectar is deposited and they perform a waggle dance in a dance floor. The information including direction of flower patches, distance from their hive and fitness is communicated through their waggle dance by exchanging the angle information between sun and patches, duration and frequency of the dance. Follower bees follow a dancer bee to quickly and efficiently collect food. The same patch is advertised through the dance for many times when going to the hive if it is good enough as a food source, and more bees are attracted to that patch. The flower patches with more nectar or pollen are visited by more bees. Thus, patches may be visited by more bees, or abandoned depending on the fitness. BeesA has been applied in many areas, *e.g.*, real-time production scheduling [282] and intelligent transportation systems [283].

BeesA is very efficient in obtaining high-quality solutions to constrained optimization problems. However, there are a number of tunable parameters that need to be figured out. In addition, it is often easy to trap into a local optimum

in its search process and causes premature convergence. Thus, the quality of its final solutions is unsatisfied if it is used to solve complicated optimization problems with large solution spaces. Simulated annealing (SA) is able to escape from a locally optimal solution by conditionally enabling some moves to worsen solutions by using its criterion of Metropolis acceptance [284].

It is proven that SA can theoretically obtain a global optimum with high probability, and it can obtain high-accuracy solutions to different types of discrete and continuous optimization problems [52]. Nevertheless, its convergence process is relatively slow in most cases. Thus, this chapter designs a hybrid algorithm named Bees Algorithm based on Simulated annealing (BAS) to solve the unconstrained optimization problem by integrating the Metropolis acceptance criterion of SA into BeesA. Specifically, this chapter performs SA-based selection with to update each elite or non-elite bee. The other novelty is SA-based selection and the use of disruptive selection to increase its convergence speed and solution accuracy.

10.2.1 Individual Encoding

Each scout bee (individual) i contains decision variables including $\overset{o}{N}_{\tau,c,n}$, $\lambda_{\tau}^{c,n}$, and $\varrho_{\tau}^{c,n}$. Let $\mathbf{x}_i$ denote the position of each individual i.

$$\mathbf{x}_i = \left[\overset{o}{N}_{\tau,1,1}, \cdots, \overset{o}{N}_{\tau,\mathcal{N}^C,\mathcal{N}^A}, \lambda_{\tau}^{1,1}, \cdots, \lambda_{\tau}^{\mathcal{N}^C,\mathcal{N}^A}, \varrho_{\tau}^{1,1}, \cdots, \varrho_{\tau}^{\mathcal{N}^C,\mathcal{N}^A} \right] \tag{10.28}$$

10.2.2 Population Initialization

Let $|\mathbb{X}|$ denote the number of scout bees, *i.e.*, the population size. Let $\overset{0}{\mathbf{x}}_{i,d}$ denote the initial decision variable d of individual $i \in \{1, 2, \ldots, |\mathbb{X}|\}$, *i.e.*,

$$\overset{0}{\mathbf{x}}_{i,d} = \check{\boldsymbol{\theta}}_5^d + w_{18,i} * \left(\hat{\boldsymbol{\theta}}_5^d - \check{\boldsymbol{\theta}}_5^d \right) \tag{10.29}$$

where $\hat{\boldsymbol{\theta}}_5^d$ and $\check{\boldsymbol{\theta}}_5^d$ are upper and lower limits of decision variable d, and $w_{18,i}$ is a number for individual i randomly produced in the range of (0,1).

10.2.3 SA-based Selection

This chapter performs SA-based selection to update each elite or non-elite bee. Let $\mathbf{x}_i^g$ and $\mathbf{x}_i^{g+1}$ denote positions of bee i in iterations g and g+1. If $\widetilde{f_{22}}(\mathbf{x}_i^g) \geq \widetilde{f_{22}}(\mathbf{x}_i^{g+1})$, $\mathbf{x}_i^{g+1}$ is accepted; otherwise, it is accepted only if

$$\frac{exp^{\left(\widetilde{f_{22}}(\mathbf{x}_i^g)-\widetilde{f_{22}}(\mathbf{x}_i^{g+1})\right)}}{\theta_2^g} > w_{17} \tag{10.30}$$

where θ_2^g is the temperature in iteration g and w_{17} is a random number in (0,1).

10.2.4 Disruptive Selection

Disruptive selection provides more chances to select lower and higher individuals in the population. To achieve it, the following equations are first defined as:

$$\tilde{\mathcal{F}}_i = |\mathcal{F}_i - \bar{\mathcal{F}}| \tag{10.31}$$

$$\acute{\mathcal{F}}_i = \frac{\tilde{\mathcal{F}}_i}{\sum_{i=1}^{|\mathbb{X}|} \tilde{\mathcal{F}}_i} \tag{10.32}$$

As is shown in (10.31), a disruptive selection operation alters fitness value ($\mathcal{F}_i$) for individual i in the population. First, $\tilde{\mathcal{F}}_i$ is defined as the absolute value of the difference between the fitness ($\mathcal{F}_i$) of individual i, and the average value ($\bar{\mathcal{F}}$) for all individuals. Second, the new fitness function ($\acute{\mathcal{F}}_i$) for individual i is calculated via (10.31) as $\frac{\tilde{\mathcal{F}}_i}{\sum_{i=1}^{|\mathbb{X}|} \tilde{\mathcal{F}}_i}$. It is observed that the higher and lower-quality individuals are more preferable. This means that disruptive selection aims to increase the diversity of individuals in the population by retaining diverse individuals. Then, the population of $|\mathbb{X}|$ scout bees is sorted with disruptive selection in (10.31).

Algorithm 8 shows details of BAS. Line 1 initializes $|\mathbb{X}|$ scout bees being randomly placed in the search space. Let $\hat{g}$ denote number of generations. θ_2^g denotes current temperature in iteration g. Line 3 sets the initial temperature θ_2^0 for SA. The while loop in Lines 4–26 produces the best solution searched, $\overset{*}{\mathbf{x}}$. Line 5 evaluates the fitness of the sites searched by scout bees after return. Line 6 reorders the population

Algorithm 8 BAS (Bees Algorithm based on Simulated annealing)

1: Initialize a population of $|\mathbb{X}|$ scout bees
2: $g \leftarrow 1$
3: $\theta_2^g \leftarrow \theta_2^0$
4: **while** $g \leq \hat{g}$ **do**
5: Evaluate the fitness of the population
6: Sort the population of $|\mathbb{X}|$ scout bees with disruptive selection in (10.31)
7: Select the best scout bee ($\overset{*}{\mathbf{x}}$)
8: Select $\acute{M}$ sites for the neighborhood search
9: Determine the neighborhood size, $\tilde{M}$
10: **for** $i \leftarrow 1$ to $\grave{M}$ **do**
11: Recruit $\check{M}$ bees for elite site i
12: Select the best one among $\check{M}$ recruited bees
13: Perform SA-based selection with (10.30) to update elite bee i
14: **end for**
15: **for** $l \leftarrow i$ to $\acute{M}-\grave{M}$ **do**
16: Recruit $\bar{M}$ bees for non-elite site i
17: Select the best one among $\bar{M}$ recruited bees
18: Perform SA-based selection with (10.30) to update non-elite bee i
19: **end for**
20: Update the fittest bee from each selected site
21: Assign $|\mathbb{X}|-\acute{M}$ remaining bees to random search
22: Produce a new population of $|\mathbb{X}|$ scout bees
23: Reduce the neighborhood radius
24: $g \leftarrow g+1$
25: $\theta_2^g \leftarrow \theta_2^{g-1} * \theta_3$
26: **end while**
27: Output $\overset{*}{\mathbf{x}}$

of $|\mathbb{X}|$ scout bees with disruptive selection (10.31). Line 8 selects $\acute{M}$ fittest sites from $|\mathbb{X}|$ scout bees for neighborhood search. Line 9 determines the neighborhood size, $\tilde{M}$, and performs a neighborhood search to update $\acute{M}$ scout bees. It is important because there may be better bees in the neighborhood area. Lines 10–14 update $\grave{M}$ elite sites by randomly recruiting $\check{M}$ bees for each elite site, selecting the best one among $\check{M}$ recruited bees, and performing SA-based selection with (10.30) to update each elite bee. Similarly, Line 9 determines the neighborhood size. Lines 15–19 update $\acute{M}-\grave{M}$ non-elite sites by randomly recruiting $\bar{M}$ bees for each non-elite site, selecting the best one among $\bar{M}$ recruited bees, and performing SA-based selection with (10.30) to update each non-elite bee.

Line 20 updates the fittest bee from each selected site to form the next population. More bees are recruited to follow the elite $\grave{M}$ sites to search in their neighborhood including more promising solutions. Therefore, the differential recruitment is an important and key operation of BeesA. Line 21 assigns $|\mathbb{X}|-\acute{M}$ remaining bees to random search. Line 22 produces a new population of $|\mathbb{X}|$ scout bees. Then, the new population has two parts including a representative from each selected patch, and other scout bees that randomly searched. Line 23 reduces the neighborhood radius. Line 25 reduces θ_2^g by θ_3, which denotes the temperature cooling rate. Finally, Line 27 outputs the best scout bee ($\overset{*}{\mathbf{x}}$), which is further transformed into decision variables including $[\overset{o}{N}_{\tau,1,1}, \cdots, \overset{o}{N}_{\tau,\mathcal{N}^C,\mathcal{N}^A}, \lambda_\tau^{1,1}, \cdots, \lambda_\tau^{\mathcal{N}^C,\mathcal{N}^A}, \varrho_\tau^{1,1}, \cdots, \varrho_\tau^{\mathcal{N}^C,\mathcal{N}^A}]$.

10.3 Performance Evaluation

This chapter evaluates the proposed BAS with real-life data. BAS is implemented and coded with MATLAB 2017, and it is executed in a server with a 32-GB DDR4 memory and an Intel Xeon E5-2699AV4 CPU at 2.4 GHz. This chapter adopts realistic task arriving rates of three applications in Google cluster trace[1] to evaluate the proposed BAS. Figure 10.1 shows task arriving rates.

10.3.1 Parameter Setting

Similar to Chapter 8, this chapter adopts the real-life prices of power grid in Figure 8.3 to set three CDCs. In addition, this chapter adopts the wind speed in Figure 3.3 and solar irradiance in Figure 3.4 to set three CDCs. In addition, the length of each time slot is 5 minutes, *i.e.*, L=300 seconds.

Here, this chapter considers three applications deployed in three CDCs, *i.e.*, $\mathcal{N}^C$=3 and $\mathcal{N}^A$=3. Following the work in [22, 24], Table 10.3 shows the setting of parameters related to energy suppliers including power grid, wind energy and solar energy. In addition, this chapter adopts real-life data about wind speed[2] and solar

[1]https://github.com/google/cluster-data (accessed on May 6, 2019).
[2]http://www.nrel.gov/midc/nwtc_m2/ (accessed on May 10, 2019).

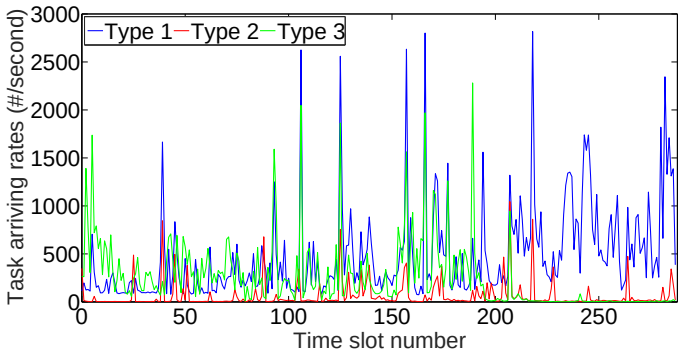


Figure 10.1 Task arriving rates of three applications.

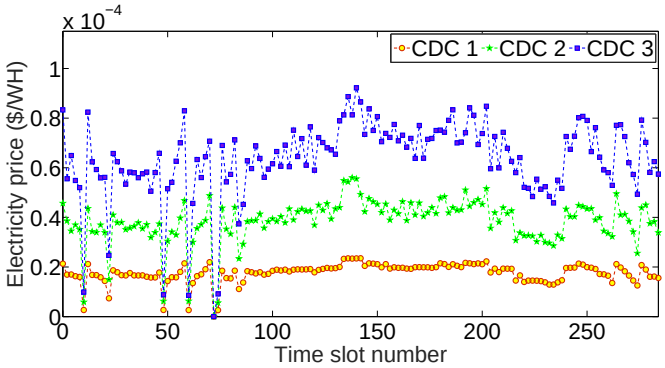


Figure 10.2 Prices of power grid in three CDCs.

Table 10.3 Parameter Setting of Wind and Solar Energy

	Wind energy			Solar energy	
	ϕ_{1c}	ϕ_{3c} (m^2)	ϕ_{2c} (kg/m^3)	ψ_{1c}	ψ_{2c} (m^2)
c=1	0.3	250	1.225	0.2	150
c=2	0.375	312.5	1.5313	0.25	187.5
c=3	0.45	375	1.8375	0.3	225

Table 10.4 Parameter Setting of Three CDCs-Part 1

	$\hat{\mathbb{E}}_c$(WH)	$\beth_c^n$			Ψ_c^n			$\check{\Phi}_n^c$ (W)		
		n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
c=1	2.1×10^{11}	10.5	7.4	4.4	1.1	1.15	1.25	200	300	400
c=2	2.5×10^{11}	12.5	9.4	6.4	1.0	1.1	1.2	400	500	600
c=3	1.25×10^{11}	14.5	11.4	6.4	1.1	1.2	1.3	600	700	800

irradiance[3] for 24 hours. In addition, similar to Chapter 9, this chapter adopts the wind speed in Figure 3.3 and solar irradiance in Figure 3.4 to set three CDCs.

[3]http://www.nrel.gov/midc/srrl_bms/ (accessed on May 10, 2019).

Table 10.5 Parameter Setting of Three CDCs-Part 2

	$\hat{N}_{c,n}$			$\hat{\varrho}_c^n$		
	n=1	n=2	n=3	n=1	n=2	n=3
c=1	110	120	130	2.9×10^4	3.0×10^4	3.1×10^4
c=2	140	150	160	3.0×10^4	3.1×10^4	3.2×10^4
c=3	160	170	180	3.2×10^4	3.3×10^4	3.4×10^4

In addition, $\urcorner_c^n$, Ψ_c^n, $\check{\Phi}_n^c$, $\hat{N}_{c,n}$ and $\hat{\varrho}_c^n$ are set in Tables 10.4–10.5. Besides, $\hat{T}_1$=0.05 seconds, $\hat{T}_2$=0.1 seconds, and $\hat{T}_3$=0.15 seconds. It is worth noting that BAS is sensitive to its parameter setting. Therefore, according to the parameter setting in previous studies [52, 207, 285], many trials are performed to investigate the optimal parameter setting in BAS with a grid search approach [286, 287]. The final parameter setting of BAS is given as follows. $\mathcal{N}^C$=3, $\mathcal{N}^A$=3, $\check{\boldsymbol{\theta}}_5^d$=0 ($1\leq d\leq 3*\mathcal{N}^C*\mathcal{N}^A$), $\hat{\boldsymbol{\theta}}_5^{1:9}$=[110, 120, 130, 140, 150, 160, 160, 170, 180], $\hat{\boldsymbol{\theta}}_5^{10:18}=2\times10^3$, and $\hat{\boldsymbol{\theta}}_5^{19:27}$=[$2.9\times10^4$, 3.0×10^4, 3.1×10^4, 3.0×10^4, 3.1×10^4, 3.2×10^4, 3.2×10^4, 3.3×10^4, 3.4×10^4]. $\hat{g}$=1000, $|\mathbb{X}|$=30, $\acute{M}$=15, $\grave{M}$=6, $\check{M}$=30 and $\bar{M}$=15. Let θ_1^S, $\acute{\theta}_{1,c,n}^S$ and $\tilde{\theta}_{1,c,n}^S$ denote the neighborhood radius of $\overset{o}{N}_{\tau,c,n}$, $\lambda_\tau^{c,n}$ and $\varrho_\tau^{c,n}$, respectively. Then, θ_1^S=14.7, $\acute{\theta}_{1,c,n}^S$=200 and $\tilde{\theta}_{1,c,n}^S$=3133. Let θ_2^S denote the reduction rate of neighborhood radius and θ_2^S=0.99. Besides, $\theta_2^0=5\times10^6$ and θ_3=0.985. In addition, $\overset{\infty}{\mathcal{N}}=10^{17}$ and $\overset{0}{\gamma}_1=\overset{0}{\gamma}_2$=1.

10.3.2 Experimental Results

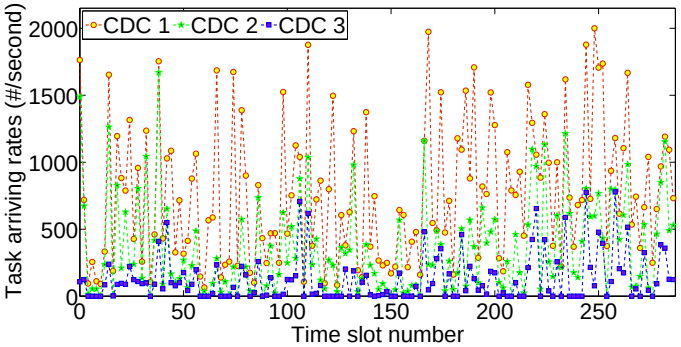


Figure 10.3 Arriving rates of type 1 tasks allocated to three CDCs.

Figures 10.3–10.5 show the arriving rates of tasks of three applications allocated to three CDCs, respectively. It is clearly observed that the number of tasks of

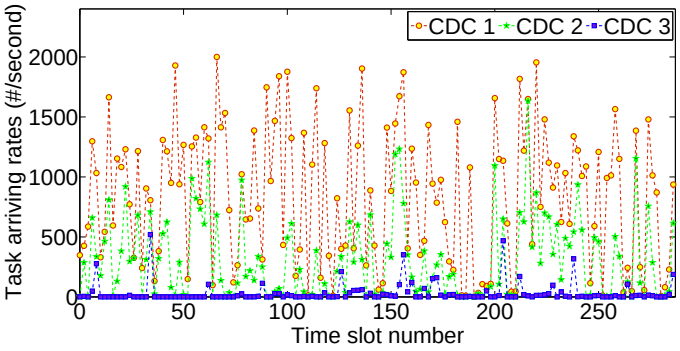


Figure 10.4 Arriving rates of type 2 tasks allocated to three CDCs.

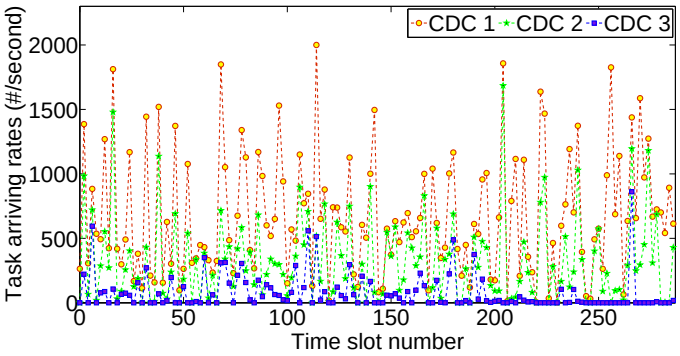


Figure 10.5 Arriving rates of type 3 tasks allocated to three CDCs.

each application allocated to CDC 1 is the highest and that allocated to CDC 3 is the lowest. Figures 10.6–10.8 show the number of switched-on servers for three applications, respectively. It is clearly observed that they all do not exceed their corresponding limits. Besides, it is also observed that the number of switched-on servers in CDC 1 for each application is the highest and that in CDC 3 is the lowest. This is because the price of power grid of CDC 1 is the lowest and that of CDC 3 is the highest. Therefore, the largest number of tasks are scheduled to CDC 1 with the largest number of switched-on servers among three CDCs.

Figure 10.9 illustrates the amount of power grid energy consumed by three CDCs. As is shown in Figure 3.6, prices of power grid of three CDCs vary from each other. BAS aims to minimize the energy cost of the CDC provider by smartly scheduling tasks of heterogeneous applications among multiple CDCs while satisfying delay-bound constraints of all tasks of each application. It is shown that the amount

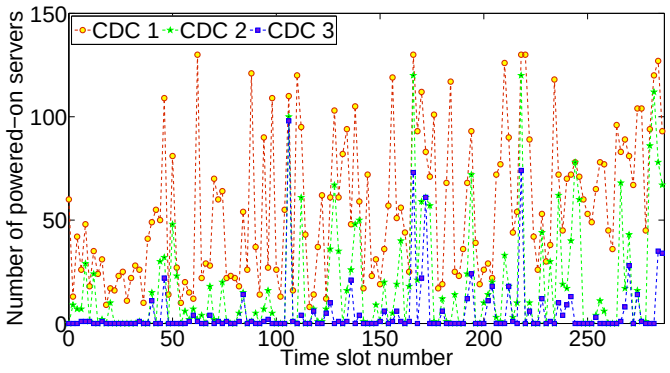


Figure 10.6 Number of switched-on servers in three CDCs for type 1 application.

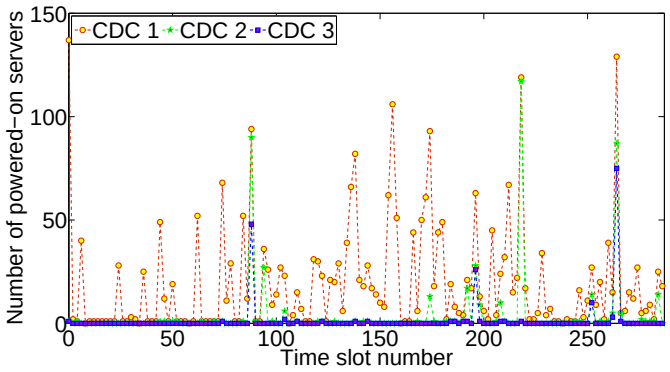


Figure 10.7 Number of switched-on servers in three CDCs for type 2 application.

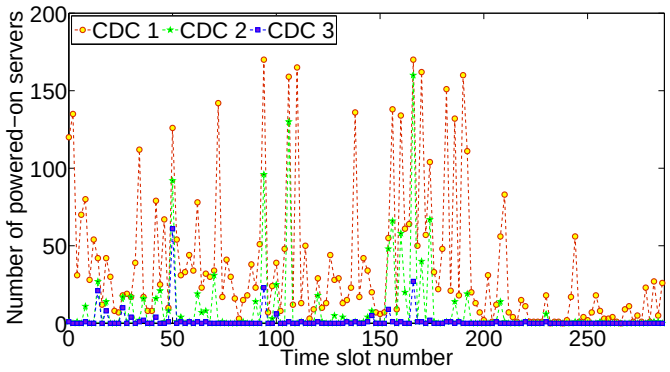


Figure 10.8 Number of switched-on servers in three CDCs for type 3 application.

of power grid energy consumed by CDC 1 is the highest while that in CDC 3 is the lowest. The result is consistent with prices of power grid in three CDCs, *i.e.*, the price of power grid in CDC 1 is the lowest while that in CDC 3 is the highest.

Figure 10.10 shows the total energy consumption of three CDCs. It is shown that the total energy consumption of each CDC does not exceed its maximum available

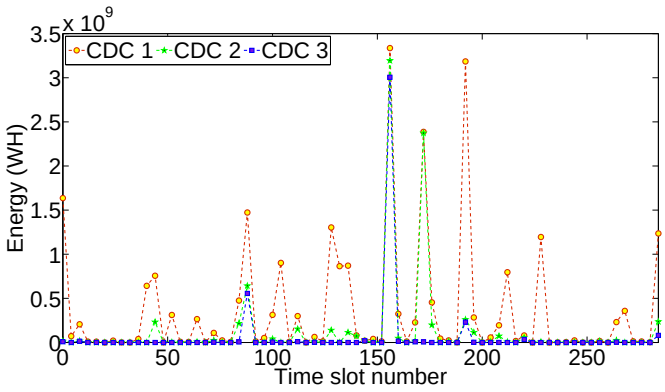


Figure 10.9 Amount of power grid energy consumed by three CDCs.

energy in each time slot. The reason is that BAS dynamically consumes the power grid, wind and solar energy by smartly scheduling tasks of heterogeneous applications among multiple CDCs, and suitably setting the running speed of each server in each CDC and the number of switched-on servers in each CDC.

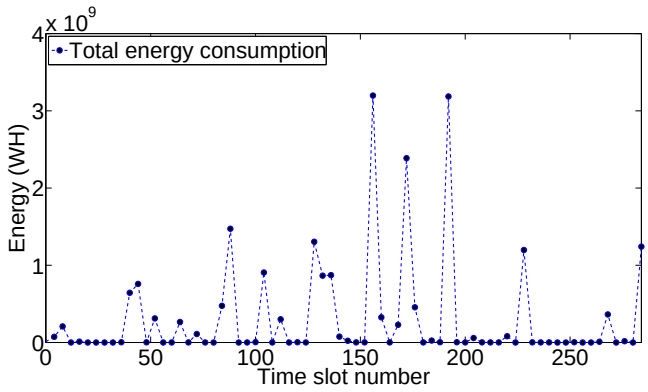
10.3.3 Comparison Results

To demonstrate the performance of BAS, this chapter compares it with two typical meta-heuristic optimization algorithms, *i.e.*, BeesA [288] and Genetic Learning Particle Swarm Optimization (GL-PSO) [289]. Each algorithm is repeated for 30 times independently to generate the statistical results. The reasons of selecting them as BAS's peers are:

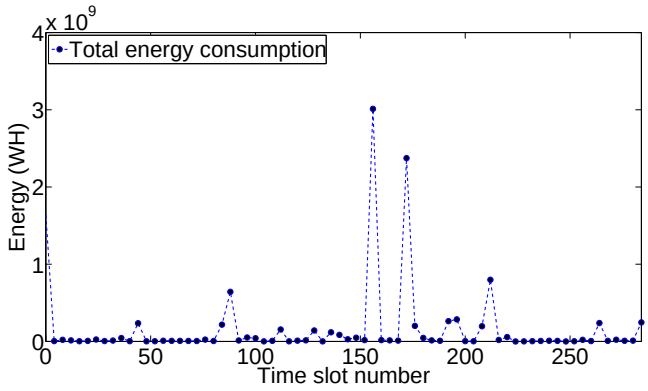
1) BeesA [288]: As a swarm-based optimization algorithm, BeesA is very efficient in finding high-quality solutions. BeesA has a search procedure that is inspired by the way honey-bee forage for food. However, it needs to figure out a number of tunable parameters, and it often easily traps into a local optimum and causes premature convergence.

2) GL-PSO [289]: GL-PSO performs crossover, mutation and selection on particles' historical information to construct well diversified and highly qualified exemplars that guide particles' search processes. GL-PSO enhances both the search efficiency, robustness, scalability and the global search ability of PSO.

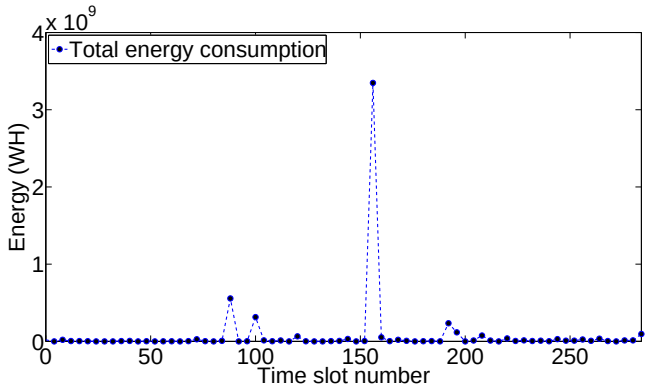
The key parameter setting of BeesA is the same as that of BAS. In addition, the key parameter setting of GL-PSO is given as follows. The number of iterations is 1000. The population size is 100. The intertia weight is 0.7298. The accelerate



(a) CDC 1



(b) CDC 2



(c) CDC 3

Figure 10.10 Total energy consumption of three CDCs

coefficients of the locally best and the globally best individuals are both set to 2. The exemplar learning coefficient is set to 1.49618. The maximum velocity is 10. The probability of mutation is 0.1. The comparison among BAS, BeesA and GL-PSO can demonstrate the accuracy and the convergence speed of the final solution of BAS. In addition, it is worth noting that BeesA and GL-PSO are all sensitive to their

parameter setting. Consequently, similar to BAS, many experiments are performed to specify the optimal parameter setting of both BeesA and GL-PSO according to the grid search method [286] and similar setting of parameters in previous studies [288, 289]. In addition, BeesA and GL-PSO terminate their search processes if they do not find better solutions in successive 10 iterations.

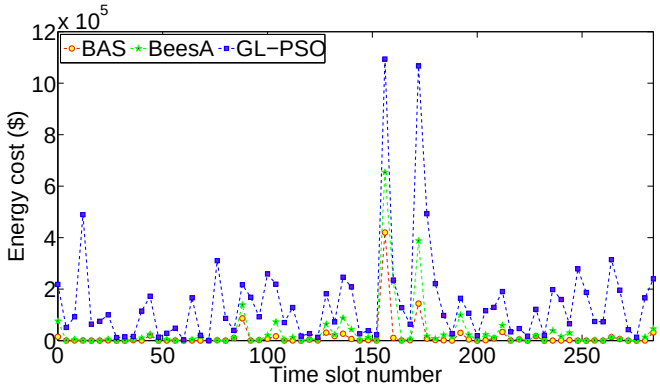


Figure 10.11 Energy cost comparison of BAS, BeesA and GL-PSO.

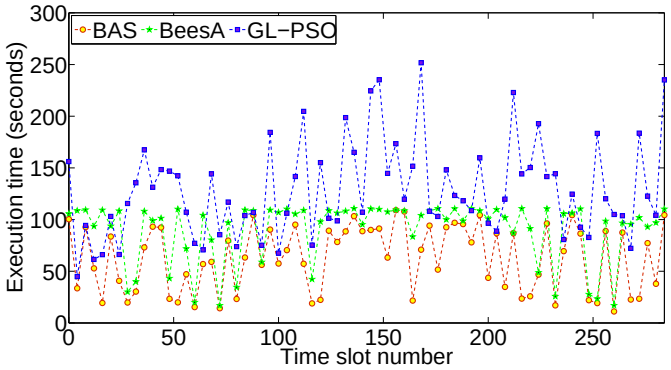


Figure 10.12 Execution time comparison of BAS, BeesA and GL-PSO.

Figure 10.11 shows the energy cost comparison of BAS, BeesA and GL-PSO. It is shown that compared with BeesA and GL-PSO, the energy cost of BAS is decreased by 59.07% and 92.83% on average, respectively. Figure 10.12 shows the execution time comparison of BAS, BeesA and GL-PSO. It is observed that compared with BeesA and GL-PSO, the execution time of BAS is decreased by 26.31% and 46.15% on average, respectively. BAS's average execution time of all time slots is 65.94 seconds, and it is 26.16% smaller than that of BeesA, 89.30 seconds, and 49.27% smaller than

that of GL-PSO, 129.97 seconds. Figures 10.11 and 10.12 demonstrate that BAS obtains a more accurate solution in less convergence time than BeesA and GL-PSO.

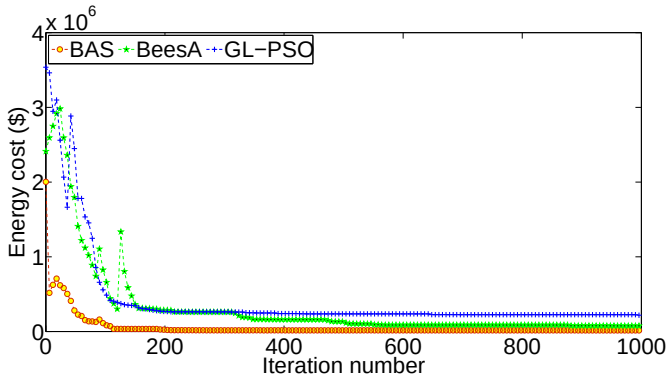


Figure 10.13 Energy cost of each iteration in time slot 1.

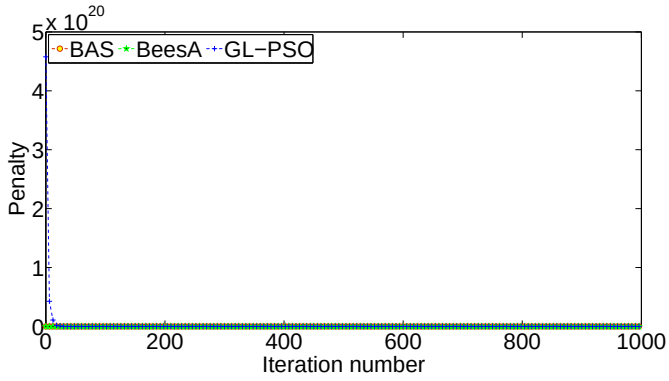


Figure 10.14 Penalty of each iteration in time slot 1.

Figure 10.13 illustrates the energy cost comparison of each iteration of BAS, BeesA and GL-PSO in time slot 1. Here, each iteration of BAS means Lines 5–25 in Algorithm 8. The meaning of iterations of BeesA and GL-PSO is similar to that of BAS. Figure 10.14 illustrates the penalty of each iteration in time slot 1, which is obtained with (3.21) in Chapter 3. It is shown the penalty of final solutions of BAS, BeesA and GL-PSO is near 0. This result demonstrates that their final solutions satisfy all the constraints of the formulated problem. BeesA and GL-PSO need 951 and 996 iterations to converge to their final solutions, and their final energy cost are \$76165.77 and \$218339.94, respectively. BAS only needs 201 iterations to converge to its final solution, and its energy cost is \$14832.58. Consequently, BAS significantly

reduces the energy cost of the CDC provider in much fewer iterations than BeesA and GL-PSO. Figures 10.13–10.14 demonstrate that the integration of the Metropolis acceptance criterion of SA in BAS improves the solution diversity, and the global search accuracy of BeesA.

To prove the effectiveness of FSTS, this chapter compares it with several task scheduling methods [31, 76, 222, 290] in terms of energy cost and throughput.

1) M1. Similar to cheap price of power grid-first scheduling in [76], it schedules tasks to CDCs by following the order of their prices of power grid.
2) M2. Similar to green energy-first scheduling in [31], it schedules tasks to CDCs by following the order of their amount of wind and solar energy.
3) M3 [222]. It schedules tasks among distributed CDCs by leveraging geographic and temporal variations of energy prices.
4) M4 [290]. It cost-effectively schedules tasks among CDCs by exploiting spatial diversity of electricity prices.

Figures 10.15–10.17 show the throughput comparison of FSTS and M1–M4, respectively. It is shown that the throughput of FSTS is greater than those of M1–M4 for each application in each time slot, respectively. For example, for application 1, FSTS's throughput is greater than those of M1–M4 by 25.99%, 25.37%, 10.30% and 7.74% on average, respectively. The reason is that the maximum number of servers, running speed limits of each server and maximum energy in each CDC are all limited in each time slot. In addition, FSTS intelligently schedules tasks of each application among CDCs, and optimally sets the running speed of each server and the number of switched-on servers in each CDC. Therefore, some tasks of users are refused and not scheduled to CDCs when using M1–M4.

Figure 10.18 illustrates the energy cost of FSTS, M1–M4, respectively. To ensure the actual performance of tasks, the penalty is required in SLAs for each rejected task [291] after the negotiation between the CDC provider and users. The penalty of each rejected task is usually greater than the largest energy cost corresponding to the execution of each task of the same application among CDCs in each time slot. Thus,

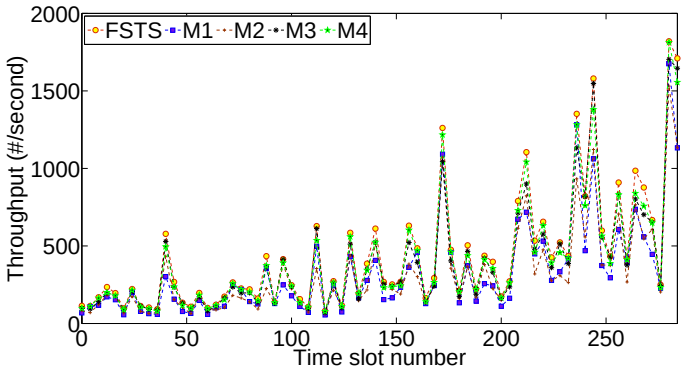


Figure 10.15 Type 1 throughput comparison of FSTS and M1–M4.

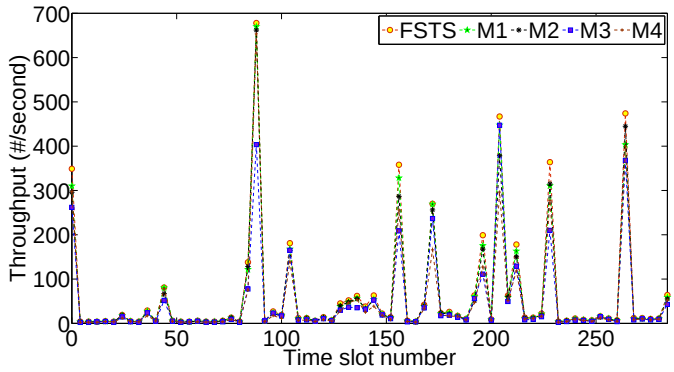


Figure 10.16 Type 2 throughput comparison of FSTS and M1–M4.

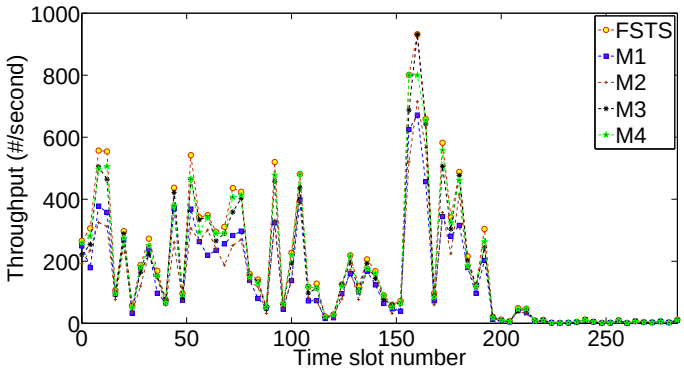


Figure 10.17 Type 3 throughput comparison of FSTS and M1–M4.

this motivates the CDC provider to strictly guarantee delay constraints of tasks of all applications. In Figure 10.18, the energy cost in each time slot is calculated by summing up the energy cost of tasks executed in CDCs, and the penalty of rejected tasks in each time slot. It is shown in Figure 10.18 that compared with M1–M4, the energy cost of FSTS is decreased by 50.11%, 51.55%, 29.15%, and 25.27% on average,

respectively. This is because FSTS intelligently schedules tasks among CDCs by jointly investigating spatial variations in prices of power grid and the amount of green energy in CDCs.

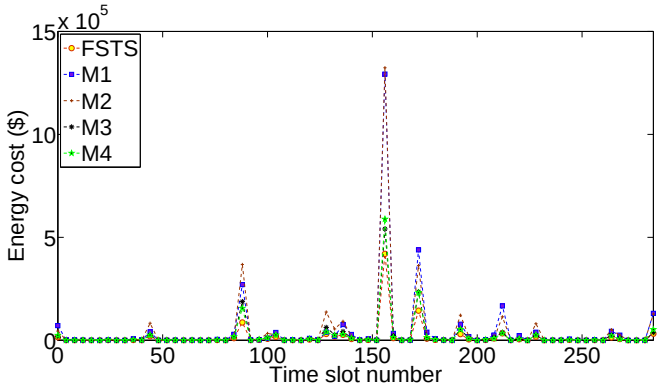


Figure 10.18 Energy cost comparison of FSTS and M1–M4.

10.4 Appendix

The derivation of (10.8) is given here. The execution time of each task on a server of application n in CDC c in time slot τ is $\frac{r_c^n}{\varrho_\tau^{c,n}}$, *i.e.*, $t_\tau^{c,n}=\frac{r_c^n}{\varrho_\tau^{c,n}}$.

Let $W_\tau^{c,n}$ denote the average waiting time of tasks of application n in each server in CDC c. According to the theory of the $G/G/1$ queuing system, $W_\tau^{c,n}$ is approximately calculated as follows.

$$W_\tau^{c,n}=\frac{1+\left(\Omega_\tau^{c,n}\right)^2}{(\frac{1}{\rho_\tau^{c,n}})^2+\left(\Omega_\tau^{c,n}\right)^2}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2\bar{T}_\tau^{c,n}(1-\rho_\tau^{c,n})} \tag{10.33}$$

According to (10.33),

$$\begin{aligned} W_\tau^{c,n}&=\frac{1+\left(\Omega_\tau^{c,n}\right)^2}{(\frac{1}{\rho_\tau^{c,n}})^2+\left(\Omega_\tau^{c,n}\right)^2}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2\bar{T}_\tau^{c,n}(1-\rho_\tau^{c,n})} \\ &=\frac{1+\frac{\tilde{\sigma}_{\tau,c,n}}{(\bar{t}_\tau^{c,n})^2}}{(\frac{\bar{T}_\tau^{c,n}}{\bar{t}_\tau^{c,n}})^2+\frac{\tilde{\sigma}_{\tau,c,n}}{(\bar{t}_\tau^{c,n})^2}}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2\bar{T}_\tau^{c,n}(1-\rho_\tau^{c,n})} \\ &=\frac{(\bar{t}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}{(\bar{T}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2\bar{T}_\tau^{c,n}(1-\rho_\tau^{c,n})} \end{aligned} \tag{10.34}$$

Then, the average response time of tasks of application n in each server in CDC c in time slot τ is denoted by $T_\tau^{c,n}$, and it is calculated as follows.

$$
\begin{aligned}
&T_\tau^{c,n}=\bar{\mathcal{T}}_\tau^{c,n}+W_\tau^{c,n}\\
&=\frac{(\bar{t}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}{(\bar{\mathcal{T}}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2\bar{\mathcal{T}}_\tau^{c,n}(1-\rho_\tau^{c,n})}\\
&=\bar{\mathcal{T}}_\tau^{c,n}+\frac{(\bar{t}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}{(\bar{\mathcal{T}}_\tau^{c,n})^2+\tilde{\sigma}_{\tau,c,n}}\frac{\hat{\sigma}_{\tau,c,n}+\tilde{\sigma}_{\tau,c,n}}{2(\bar{\mathcal{T}}_\tau^{c,n}-\bar{t}_\tau^{c,n})}\\
&=\frac{\bar{r}_c^n}{\varrho_\tau^{c,n}}+\frac{\frac{(\bar{r}_c^n)^2}{\varrho_\tau^{c,n}}+\frac{\breve{\sigma}_{c,n}}{(\varrho_\tau^{c,n})^2}}{(\bar{\mathcal{T}}_\tau^{c,n})^2+\frac{\breve{\sigma}_{c,n}}{(\varrho_\tau^{c,n})^2}}\frac{\hat{\sigma}_{\tau,c,n}+\frac{\breve{\sigma}_{c,n}}{(\varrho_\tau^{c,n})^2}}{2(\bar{\mathcal{T}}_\tau^{c,n}-\frac{\bar{r}_c^n}{\varrho_\tau^{c,n}})}\\
&=\frac{\bar{r}_c^n}{\varrho_\tau^{c,n}}+\frac{(\bar{r}_c^n)^2+\breve{\sigma}_{c,n}}{(\bar{\mathcal{T}}_\tau^{c,n})^2(\varrho_\tau^{c,n})^2+\breve{\sigma}_{c,n}}\frac{\hat{\sigma}_{\tau,c,n}(\varrho_\tau^{c,n})^2+\breve{\sigma}_{c,n}}{2\breve{\sigma}_{c,n}(\bar{\mathcal{T}}_\tau^{c,n}\varrho_\tau^{c,n}-\bar{r}_c^n)}\\
&=\frac{\bar{r}_c^n}{\varrho_\tau^{c,n}}+((\bar{r}_c^n)^2+\breve{\sigma}_{c,n})\frac{\hat{\sigma}_{\tau,c,n}(\varrho_\tau^{c,n})^2+\breve{\sigma}_{c,n}}{2\varrho_\tau^{c,n}(\bar{\mathcal{T}}_\tau^{c,n}\varrho_\tau^{c,n}-\bar{r}_c^n)((\bar{\mathcal{T}}_\tau^{c,n})^2(\varrho_\tau^{c,n})^2+\breve{\sigma}_{c,n})}
\end{aligned}
\tag{10.35}
$$

It is worth noting that equation (10.35) is equivalent to (10.8). Then, (10.8) is derived accordingly.

10.5 Summary

Cloud computing allows enterprises to achieve many benefits by reducing administrative, capital and operational cost. Yet it suffers from the high energy consumption problem that negates its advantages. Many large-scale enterprises adopt distributed green cloud data center (CDC) systems to provide application services to users through intelligent task scheduling. However, existing studies fail to minimize the energy cost of a CDC provider by providing fine-grained spatial scheduling for tasks of heterogeneous applications. In addition, many factors, *e.g.*, prices of power grid and the amount of green energy in green CDCs show their significant spatial variations. Therefore, it is a big challenge to minimize the energy cost of the CDC provider. This chapter uses a $G/G/1$ queuing model to analyze the performance of servers, and further formulates a constrained optimization problem. It is solved by a newly proposed Bat Algorithm based on Simulated annealing to find a close-to-optimal

solution. Then, a Fine-grained Spatial Task Scheduling (FSTS) algorithm is proposed to achieve the energy cost minimization for the CDC provider by optimally allocating tasks of heterogeneous applications among multiple CDCs, and specifying the running speed of each server and the number of switched-on servers in each CDC. Real-life data-driven experiments demonstrate that FSTS can decrease energy cost and ensure the highest throughput in comparison with its several up-to-date scheduling methods.

CHAPTER 11

PROFIT-MAXIMIZED COLLABORATIVE COMPUTATION OFFLOADING AND RESOURCE ALLOCATION IN DISTRIBUTED CLOUD AND EDGE COMPUTING SYSTEMS

This chapter presents the details of the proposed profit-maximized collaborative computation offloading and resource allocation, and it is organized as follows. Section 11.1 first gives an illustrative framework of the system, and then formulates a constrained profit maximization problem. Section 11.2 presents details of Simulated-annealing-based Migrating Birds Optimization (SMBO) to efficiently solve it. Real-life data is adopted to evaluate it in Section 11.3. At last, the conclusion is drawn in Section 11.4.

11.1 Problem Formulation

This section gives the formulation of a constrained optimization problem for a cloud and edge computing system as is shown in Figure 11.1. In this chapter, the framework includes three layers, *i.e.*, terminal, edge computing and cloud data center (CDC) ones. The arriving tasks of users are sent through heterogeneous smart mobile devices, *e.g.*, iPad, smart phones, computers, sensors on the road and all they are produced at the terminal layer, and the returned results are sent back to this layer eventually. These tasks need to be executed by such applications as industrial internet devices, smart home, smart city, autonomous driving and medical monitoring.

This chapter considers a heterogeneous network to deliver arriving tasks to nodes in edge computing and CDC layers. As is shown in Figure 11.1, for a given area, the wireless infrastructure of this network mainly includes many WiFi access points and multiple small-cell base stations (SBSs). The cell radius of SBSs that might be hosted by different mobile telecom carriers varies from 0.01 km to 2 km, and SBSs are mutually interconnected and reachable to transmit signals and messages. SBSs,

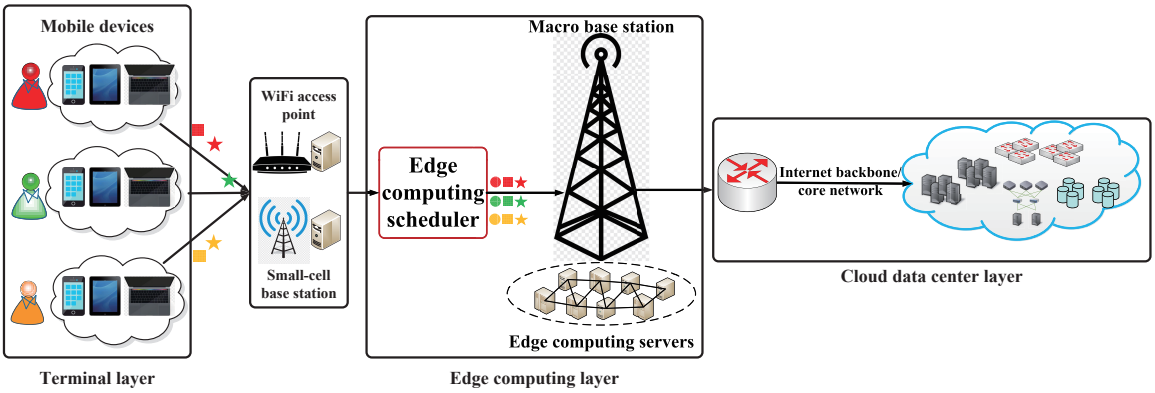


Figure 11.1 Illustrative system framework.

e.g., WiFi access points, gateways and micro cells, are part of the network for highly populated urban areas. Tasks are delivered to the edge computing layer including a task scheduler and nodes in edge computing. The scheduler smartly allocates tasks between the edge and CDC. If a task is scheduled to CDC, a macro base station just forwards it to CDC; otherwise, a macro base station (MBS) needs to execute it in its node (server) at the edge computing layer.

MBS is located at the network edge, and consists of many heterogeneous nodes with limited storage, computing and transmission ability. It provides ubiquitous coverage and its cell radius varies from 8 km to 30 km. It receives tasks from the terminal layer, and executes some of them locally and sends others to CDC through Internet backbone. It reduces tasks' latency and alleviates pressure on the CDC layer. The CDC layer typically owns and manages multiple interconnected server clusters with large computing and storage capacities and provides different kinds of cloud resources to handle complex tasks.

11.1.1 Decision Variables

Let $y_\tau^{s,q}$ be a binary variable. If task s is scheduled to execute in node q in the edge computing layer in each time slot τ, $y_\tau^{s,q}$=1; otherwise, $y_\tau^{s,q}$=0. Let $N_\tau^\dagger$ denote the number of tasks scheduled to nodes in the edge computing layer in time slot τ. Let μ_τ denote the task service rate of CDC servers in time slot τ. It is assumed that

users' arriving tasks can be executed in the edge computing or CDC layer. Thus, in time slot τ, $N_\tau^\dagger$ needs to be less than or equal to the total number of arriving tasks in τ, $\lambda_\tau L$, *i.e.*,

$$N_\tau^\dagger \leq \lambda_\tau L \tag{11.1}$$

where λ_τ is the task arriving rate in time slot τ, and L is the length of each time slot.

Let N_τ^q denote the maximum number of tasks that can be executed by node q in edge in time slot τ. Thus,

$$\sum_{s=1}^{N_\tau^\dagger} y_\tau^{s,q} \leq N_\tau^q \tag{11.2}$$

In the edge computing layer, there are N^0 heterogeneous nodes. In time slot τ, task s has to be and can only be scheduled to execute in only one node in the edge computing layer. Thus,

$$\sum_{q=1}^{N^0} y_\tau^{s,q} = 1 \tag{11.3}$$

11.1.2 Response Time

The total number of arriving tasks in time slot τ is $\lambda_\tau L$ according to (11.1). Then, the number of tasks scheduled to execute in CDC in time slot τ is $\left(\lambda_\tau L - N_\tau^\dagger\right)$. Let $\acute{\lambda}_\tau$ denote the arriving rate of tasks scheduled to CDC in time slot τ, *i.e.*, $\acute{\lambda}_\tau = \frac{\lambda_\tau L - N_\tau^\dagger}{L}$. Let $\mathbb{T}_\tau^\dagger$ denote the maximum response time of tasks executed in all nodes in the edge computing layer in time slot τ. Let $T_\tau^\ddagger$ denote the average response time of all tasks executed in the CDC layer in time slot τ.

Let $T_\tau^{s,q}$ denote the execution time of task s ($1 \leq s \leq N_\tau^\dagger$) on node q ($1 \leq q \leq N^0$). Let $\mathcal{N}_s$ denote the size of task s. Let $\hbar_q$ denote the processing speed of node q. Then,

$$T_\tau^{s,q} = \frac{\mathcal{N}_s}{\hbar_q} y_\tau^{s,q} \tag{11.4}$$

It is worth noting that the maximum response time ($\mathbb{T}_{\tau}^{\dagger}$) of all tasks executed in the edge computing layer includes transmission time and computation time in the CDC layer. Then, $\mathbb{T}_{\tau}^{\dagger}$ is obtained as:

$$\mathbb{T}_{\tau}^{\dagger}=\max_{q=1}^{N^0}\left[\sum_{s=1}^{N^0}T_{\tau}^{s,q}\right] \tag{11.5}$$

Let $\hat{T}^{\dagger}$ denote users' response time limit for tasks scheduled to execute in the edge computing layer. Thus, $\mathbb{T}_{\tau}^{\dagger}$ cannot exceed $\hat{T}^{\dagger}$, *i.e.*,

$$\mathbb{T}_{\tau}^{\dagger}\leq\hat{T}^{\dagger} \tag{11.6}$$

Similar to the work in [110, 245], this chapter adopts an $M/M/1/\hat{\mathbb{N}}^{\ddagger}/\infty$ queuing system to analyze the behavior of switched-on servers in CDC. Here, $\hat{\mathbb{N}}^{\ddagger}$ denotes the maximum number of tasks that all servers in a CDC can execute. It is assumed that users' tasks arrive in a Poisson process with mean rate $\acute{\lambda}_{\tau}$ and task service time has an exponential distribution with mean rate μ_{τ}. Let $T_{\tau}^{\ddagger}$ denote the average response time of tasks scheduled to execute in the CDC layer. Let $\hat{T}^{\ddagger}$ denote users' response time limit of tasks scheduled to execute in the CDC layer. Then, $T_{\tau}^{\ddagger}$ is obtained as:

$$T_{\tau}^{\ddagger}=\upsilon_{\tau}+\frac{\Delta_{\tau}^{18}}{\mu_{\tau}\left(1-\Delta_{\tau}^{17}\right)} \tag{11.7}$$

where

$$\Delta_{\tau}^{18}=\frac{\rho_{\tau}}{1-\rho_{\tau}}-\frac{\left(\hat{\mathbb{N}}^{\ddagger}+1\right)\left(\rho_{\tau}\right)^{\hat{\mathbb{N}}^{\ddagger}+1}}{1-\left(\rho_{\tau}\right)^{\hat{\mathbb{N}}^{\ddagger}+1}},$$

$$\Delta_{\tau}^{17}=\frac{1-\rho_{\tau}}{1-\left(\rho_{\tau}\right)^{\hat{\mathbb{N}}^{\ddagger}+1}}$$

$$\rho_{\tau}=\frac{\acute{\lambda}_{\tau}}{\mu_{\tau}}$$

υ_τ is the total transmission time of input/output data to/from the CDC layer through MBS, and Δ_τ^{17} is the possibility when there are no tasks in the CDC layer in the current time slot.

In addition, to guarantee the stability of a task queue in the CDC layer, (11.8) has to be met:

$$\acute{\lambda}_\tau \leq \mu_\tau \tag{11.8}$$

In time slot τ, $T_\tau^\ddagger$ cannot exceed its limit $\hat{T}^\ddagger$, *i.e.*,

$$T_\tau^\ddagger \leq \hat{T}^\ddagger \tag{11.9}$$

11.1.3 Profit

Then, let f_1, f_2 and F_1 denote total revenue, total cost and total profit in time slot τ. Then, F_1 is obtained as:

$$F_1 = f_1 - f_2 \tag{11.10}$$

To meet the actual response time requirements of users, this chapter uses service level agreements (SLAs) [292, 293] as legal contracts. They are typically predefined between users and cloud providers. In this chapter, SLAs specify the revenue or penalty if the response time of tasks is within or beyond their limits, respectively. Long response time of tasks leads to worse quality of applications and it reduces profit of the system providers that not only have to meet tasks' quality of service (QoS), but also to maximize the total profit. In this chapter, QoS is defined as the response time of tasks. This chapter adopts the following SLA as an example. If the actual response time of a task is less than or equal to 0.1 seconds, its revenue is \$0.5; otherwise, its penalty is -\$0.2. Let x denote the actual response time of a task, and let $\Delta^{19}(t)$ denote its corresponding revenue or penalty. Then,

$$\Delta^{19}(t) = \begin{cases} 0.5, & t \leq 0.1 \text{ seconds} \\ -0.2, & t > 0.1 \text{ seconds} \end{cases} \tag{11.11}$$

According to (11.11), the revenue brought by each task scheduled to execute in the edge computing layer is $\Delta^{19}(\mathbb{T}_{\tau}^{\dagger})$. Then, the total revenue brought by all tasks scheduled to execute in the edge computing layer is $\Delta^{19}(\mathbb{T}_{\tau}^{\dagger})N_{\tau}^{\dagger}$. Similarly, the revenue brought by each task scheduled to execute in the CDC layer is $\Delta^{19}(T_{\tau}^{\ddagger})$. Then, the revenue brought by all tasks scheduled to execute in CDC is $\Delta^{19}(T_{\tau}^{\ddagger})\left(\lambda_{\tau}L-N_{\tau}^{\dagger}\right)$. $\digamma_{\tau}$ is obtained as:

$$f_1=\Delta^{19}(\mathbb{T}_{\tau}^{\dagger})N_{\tau}^{\dagger}+\Delta^{19}(T_{\tau}^{\ddagger})\left(\lambda_{\tau}L-N_{\tau}^{\dagger}\right) \tag{11.12}$$

Let f_{22} denote the energy cost of all tasks scheduled to execute in the CDC layer in time slot τ. Let f_{23} denote the execution cost of all tasks scheduled to execute in the edge computing layer in time slot τ. Then, f_2 consists of f_{22} and f_{23}. Let $f_{23}^{s,q}$ denote the execution cost of task s scheduled to execute in node q in the edge computing layer in time slot τ. Let $\hat{\partial}$ denote the upper limit of $f_{23}^{s,q}$.

$$f_{23}^{s,q}\leq\hat{\partial} \tag{11.13}$$

f_{23} is obtained as $\sum_{s=1}^{N_{\tau}^{\dagger}}\sum_{q=1}^{N^0}\left(y_{\tau}^{s,q}f_{23}^{s,q}\right)$. Let $\mathbb{p}_{\tau}$ denote the price of power grid in time slot τ. Let E_{τ} denote the amount of energy consumed by all tasks scheduled to execute in the CDC layer in time slot τ. Thus, $f_{22}=\mathbb{p}_{\tau}E_{\tau}$. Similar to the work in [294, 295], the data transmission cost between CDC and edge computing layers is ignored. Thus, f_2 is obtained as:

$$f_2=\mathbb{p}_{\tau}E_{\tau}+\sum_{s=1}^{N_{\tau}^{\dagger}}\sum_{q=1}^{N^0}\left(y_{\tau}^{s,q}f_{23}^{s,q}\right) \tag{11.14}$$

11.1.4 Energy Consumption

$\hat{\Phi}$ and $\check{\Phi}$ denote the peak and idle power of each server in the CDC layer, respectively. $\dot{\mathbb{N}}$ denotes the number of tasks processed by each switched-on server per time in the CDC layer, and α denotes the power usage effectiveness value [296] of CDC. Following

the work in [23], energy consumption E_τ is calculated as follows.

$$E_\tau = \frac{\Delta^5 \mu_\tau + \Delta^4 \acute{\lambda}_\tau \left(1-\delta_\tau\right)}{\dot{\mathbb{N}}} L \tag{11.15}$$

$$\Delta^5 = \hat{\Phi} + \left(\alpha - 1\right) \check{\Phi}$$

$$\Delta^4 = \hat{\Phi} - \check{\Phi}$$

$$\delta_\tau = \frac{1-\rho_\tau}{1-\left(\rho_\tau\right)^{\hat{\mathbb{N}}^\ddagger+1}} \left(\rho_\tau\right)^{\hat{\mathbb{N}}^\ddagger}$$

Let $\hat{\mathbb{E}}$ denote the maximum amount of the total available energy in the CDC layer. Then, E_τ cannot exceed Δ, *i.e.*,

$$\frac{\Delta^5 \mu_\tau + \Delta^4 \acute{\lambda}_\tau \left(1-\delta_\tau\right)}{\dot{\mathbb{N}}} L \leq \hat{\mathbb{E}} \tag{11.16}$$

The execution cost of all tasks scheduled to execute in the edge computing layer needs to be less than or equal to the energy cost of all tasks scheduled to execute in the CDC layer in time slot τ, *i.e.*,

$$\sum_{s=1}^{N_\tau^\dagger} \sum_{q=1}^{N^0} \left(y_\tau^{s,q} f_{23}^{s,q}\right) \leq \mathbb{p}_\tau E_\tau \tag{11.17}$$

Let $\hat{N}$ denote the maximum number of servers in the CDC layer. Then, the number of switched-on servers is $\mu_\tau / \dot{\mathbb{N}}$, which must satisfy:

$$\frac{\mu_\tau}{\dot{\mathbb{N}}} \leq \hat{N} \tag{11.18}$$

11.1.5 Load Balance Model

In the edge computing layer, several load factors can be used to evaluate the load balance of nodes [297]. This chapter considers three important factors including CPU, memory and bandwidth [298]. To allocate resource reasonably, this chapter defines the load level of each node q in time slot τ as follows.

$$\kappa_\tau^q = \varkappa_1^0 \varphi_\tau^{1,q} + \varkappa_2^0 \varphi_\tau^{2,q} + \varkappa_3^0 \varphi_\tau^{3,q} \tag{11.19}$$

where $\varphi_\tau^{1,q}$, $\varphi_\tau^{2,q}$ and $\varphi_\tau^{3,q}$ denote the utilization of CPU, memory and bandwidth resources of node q in time slot τ, respectively. $\varkappa_1^0$, $\varkappa_2^0$ and $\varkappa_3^0$ denote their weights and satisfy:

$$\varkappa_1^0+\varkappa_2^0+\varkappa_3^0=1 \tag{11.20}$$

$\varphi_\tau^{1,q}$, $\varphi_\tau^{2,q}$ and $\varphi_\tau^{3,q}$ cannot exceed 1, *i.e.*,

$$\varphi_\tau^{1,q}=\frac{\sum_{s=1}^{N_\tau^\dagger} y_\tau^{s,q}\varsigma_1^s}{\hat{\hbar}_1}\leq 1 \tag{11.21}$$

$$\varphi_\tau^{2,q}=\frac{\sum_{s=1}^{N_\tau^\dagger} y_\tau^{s,q}\varsigma_2^s}{\hat{\hbar}_2}\leq 1 \tag{11.22}$$

$$\varphi_\tau^{3,q}=\frac{\sum_{s=1}^{N_\tau^\dagger} y_\tau^{s,q}\varsigma_3^s}{\hat{\hbar}_3}\leq 1 \tag{11.23}$$

where ς_1^s, ς_2^s and ς_3^s denote the amount of CPU, memory and bandwidth resources needed by each task s, respectively. $\hat{\hbar}_1$, $\hat{\hbar}_2$ and $\hat{\hbar}_3$ are the maximum available amount of CPU, memory and bandwidth resources in each node, respectively.

Let $\bar{\kappa}_\tau$ denote the average load level of all nodes in the edge computing layer in time slot τ. Then, (11.24) is obtained.

$$\bar{\kappa}_\tau=\frac{\sum_{q=1}^{N^0} \kappa_\tau^q}{N^0} \tag{11.24}$$

Let ε_τ denote the load balance level of all nodes in the edge computing layer in time slot τ. This section uses the standard deviation to characterize it, *i.e.*,

$$\varepsilon_\tau=\sqrt{\frac{1}{N^0}\sum_{q=1}^{N^0}\left(\kappa_\tau^q-\bar{\kappa}_\tau\right)^2}\leq\hat{\varepsilon} \tag{11.25}$$

where $\hat{\varepsilon}$ denotes the specified maximum load balance level. It is worth noting that the lower ε_τ is, the more balanced the edge computing layer is.

11.1.6 Profit Maximization Problem

The objective is to maximize F_1, *i.e.*,

$$\underset{y_\tau^{s,q},N_\tau^\dagger,\mu_\tau}{\mathbf{Max}} \quad \{F_1\} \tag{11.26}$$

subject to (11.1)–(11.3), (11.6), (11.8), (11.9), (11.13), (11.16)–(11.18), (11.21)–(11.23), (11.25) and (11.27).

$$N_\tau^\dagger \in N^+,\ \mu_\tau \geq 0,\ y_\tau^{s,q} \in \{0,1\} \tag{11.27}$$

Then, SMBO used to solve the problem is described in the next section.

11.2 Simulated-annealing-based Migrating Birds Optimization

Note that F_1 is nonlinear in terms of $y_\tau^{s,q}$, $N_\tau^\dagger$ and μ_τ. Thus, it is a constrained nonlinear optimization problem. To well solve it, this chapter designs a method of penalty function to convert it into an unconstrained one, *i.e.*,

$$\underset{y_\tau^{s,q},N_\tau^\dagger,\mu_\tau}{\mathbf{Max}} \quad \left\{\widetilde{F_1}(\boldsymbol{x}) = F_1 - \overset{\infty}{\mathcal{N}}\mho\right\} \tag{11.28}$$

In (11.28), $\widetilde{F_1}$ is a new objective function and $\overset{\infty}{\mathcal{N}}$ denotes a large positive constant. $\boldsymbol{x}2$ is a decision vector including $y_\tau^{s,q}$, $N_\tau^\dagger$ and μ_τ. $\mho$ denotes the penalty if any equality and inequality constraint is violated, and it is obtained with (3.21) in Chapter 3.

There are several traditional methods, *e.g.*, convex programming, conjugate gradient methods, quasi-Newton methods and interior point methods, to solve it. But they often require the derivatives of objective functions in their optimization problems. Thus, they are applicable to solve some typical constrained optimization problems with specific mathematical characteristics [299]. In addition, their

optimization steps are complicated and their obtained solutions to often intractable problems are usually of poor-quality. To avoid such drawbacks of traditional algorithms, several nature-inspired metaheuristic-based optimization algorithms, *e.g.*, tabu search, genetic algorithm and particle swarm optimization, have been used to find near optimal solutions to large-scale constrained optimization problems. Each metaheuristic algorithm aims to find an improving solution iteration by iteration.

Among many, an emerging nature inspired metaheuristic algorithm named Migrating Birds Optimization (MBO) is proposed in [300]. It differs from other metaheuristic algorithms because it has a benefit mechanism among solutions. It is inspired and derived from an effective V flight formation of migrating birds (solutions), which brings benefits in energy saving because birds in other positions get benefit from the birds in front. In MBO, there is a leader bird in the flock (population) and two lines of other birds follow it. It is typically assumed that if the leader bird becomes tired after flying for a certain time, it flies to the end of a line and one of other following birds becomes a new leader.

Each solution compares itself with a number of its own neighbors, and several best neighbors of the front solution. It is replaced by the best of them if it becomes worse. It means that if one solution is not improved by its own neighbors, it may be replaced by one of neighbors of its front solution. This operation is repeated for several times. Then, the leader solution goes to the last position, and one of the second solutions goes to the first position. The algorithm is terminated after a given number of iterations.

MBO explores more areas within neighbor spaces of feasible solutions by first initializing a number of parallel solutions for birds in a V formation. The leader bird in the V-formation spends the most energy. In MBO, the neighborhood within more promising solutions is explored in more details. After several iterations, these solutions might move to different directions if they are improved along their

ways. Nevertheless, finally most solutions of MBO may converge to one or several local optima, and therefore, their quality is usually unsatisfying in many cases. Consequently, the search accuracy of MBO needs to be improved.

Simulated annealing (SA) can conditionally jump from local optima by accepting some worse moves with the Metropolis acceptance rule [301]. It is pointed that SA can obtain global optima with large probability in theory, and SA is widely used to find high-quality solutions to different optimization problems. However, its main drawback is its very slow convergence speed. In this chapter, a hybrid algorithm called SA-based Migrating Birds Optimization (SMBO) is proposed by combing SA's Metropolis acceptance rule and MBO. Specifically, this chapter adopts the SA-based update mechanism to change a solution. Its novelty includes the integration of an SA-based update mechanism into MBO, and the disruptive selection of solutions to improve its solution diversity.

11.2.1 Solution (Bird) Encoding

Let $|\mathbb{X}|$ denote the number of solutions (birds) in the population. Each solution i ($1 \leq i \leq |\mathbb{X}|$) includes decision variables including $y_\tau^{s,q}$, $N_\tau^\dagger$ and μ_τ. $\mathbf{x}_i$ denotes the position of each solution i.

$$\mathbf{x}_i = \left[y_\tau^{1,1}, \cdots \text{ and } y_\tau^{N_\tau^\dagger, N^0}, N_\tau^\dagger, \mu_\tau \right] \tag{11.29}$$

11.2.2 SA-based Update Mechanism

This chapter adopts an SA-based update mechanism to change a solution i. Let $\mathbf{x}_i^g$ and $\mathbf{x}_i^{g+1}$ denote positions of each solution i in iterations g and g+1. θ_2^g denotes current temperature in iteration g and w_3 denotes a random number in (0,1). If $widetildeF_1(\mathbf{x}_i^g) \geq \widetilde{F_1}(\mathbf{x}_i^{g+1})$, $\mathbf{x}_i^{g+1}$ is selected to update solution i; otherwise, it is accepted only if

$$\frac{e^{\left(\widetilde{F_1}(\mathbf{x}_i^g) - \widetilde{F_1}(\mathbf{x}_i^{g+1})\right)}}{\theta_2^g} > w_3 \tag{11.30}$$

11.2.3 Disruptive Selection

Disruptive selection chooses higher and lower-quality solutions. First, $\tilde{\mathcal{F}}_i$ is defined as the absolute difference between fitness value ($\mathcal{F}_i$) of solution i, and the average value ($\bar{\mathcal{F}}$) for all solutions in the flock. Second, the new fitness function ($\acute{\mathcal{F}}_i$) for solution i is obtained as $\tilde{\mathcal{F}}_i / \sum_{i=1}^{|\mathbb{X}|} \tilde{\mathcal{F}}_i$. According to (11.31), decentralized solutions that are farther from the average value for all birds have a larger chance to be selected. Therefore, the disruptive selection can improve the diversity of birds in the flock by selecting diverse ones. $\acute{\mathcal{F}}_i$ is given as:

$$\acute{\mathcal{F}}_i = \frac{\tilde{\mathcal{F}}_i}{\sum_{i=1}^{|\mathbb{X}|} \tilde{\mathcal{F}}_i} \tag{11.31}$$
$$\tilde{\mathcal{F}}_i = |\mathcal{F}_i - \bar{\mathcal{F}}|$$

Let $\tilde{M}$ denote the number of neighbor solutions for each solution. Let M^0 denote the number of unused neighbor solutions of the current solution, which are shared with the next solution in the V formation. Let $\mathcal{N}^*$ denote the number of time slots. Let $\hat{g}$ denote the number of total iterations. Algorithm 9 shows details of SMBO. Line 1 initializes parameters including $|\mathbb{X}|$, $\tilde{M}$, M^0, $\mathcal{N}^*$ and $\hat{g}$. Line 2 initializes a population of $|\mathbb{X}|$ initial solutions randomly. Line 3 sorts the initial flock according to the ascending order of their fitness values obtained with (11.28). In addition, at each iteration, solutions in the current flock are similarly sorted. Line 4 organizes them in a V formation according to Figure 11.2. Figure 11.2 shows an exemplar flock including 7 solutions, and their V formation. Here, solution 1 is the best while solution 7 is the worst. Thus, solution 1 is corresponding to the leader bird, and its successor solutions are 2 and 3, respectively. It is worth noting that solution 3 is worse than solution 2, and both of them are worse than solution 1 and better than solutions 4-7.

Line 6 initializes θ_2^1 with the starting temperature θ_2^0. Lines 9–30 perform $\mathcal{N}^*$ times to update all solutions in the flock. Line 10 creates $\tilde{M}$ neighbors for the leader

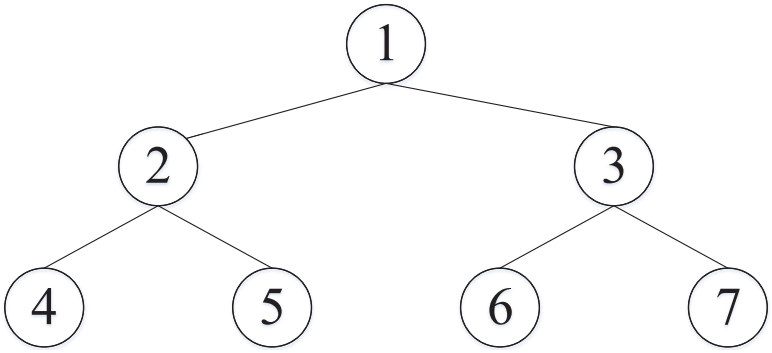


Figure 11.2 7-solution V formation.

solution and sorts them. Line 11 increases g by $\tilde{M}$. Lines 12–15 create $\tilde{M}-M^0$ neighbors for each solution i ($2\leq i\leq|\mathbb{X}|$) and sort them. g is increased by $\tilde{M}-M^0$ for each solution. Line 16 moves the best neighbor of the leader solution to $\overset{*}{\mathbb{X}}(1)$, and puts the second and third-best neighbors of the leader solution into the neighbor sets of two successors of the leader solution. Lines 17–19 move the best neighbor of solution i ($2\leq i\leq|\mathbb{X}|-2$) to $\overset{*}{\mathbb{X}}(i)$, and put M^0 unused best neighbors of solution i into the neighbor set of solution $i+2$. Line 20 moves the best neighbor of solution $|\mathbb{X}|-1$ to $\overset{*}{\mathbb{X}}(|\mathbb{X}|-1)$, and puts M^0 unused best neighbors of solution $|\mathbb{X}|-1$ into the neighbor set of solution $|\mathbb{X}|-1$. Line 21 moves the best neighbor of solution $|\mathbb{X}|$ to $\overset{*}{\mathbb{X}}(|\mathbb{X}|)$, and puts M^0 unused best neighbors of solution $|\mathbb{X}|$ into the neighbor set of solution $|\mathbb{X}|$. Lines 22–28 adopt the SA-based update mechanism to change solution i ($1\leq i\leq|\mathbb{X}|$) *via* (11.30). Specifically, if $\overset{*}{\mathbb{X}}(i)$ is better than solution i, solution i is updated with $\overset{*}{\mathbb{X}}(i)$; otherwise, the SA's metropolis rule is performed to update solution i *via* (11.30). Line 31 moves the leader solution to the end and forwards its left or right successor to the leader position. Line 32 sorts all solutions except the leader in the flock.

Let $\overset{*}{\mathbb{X}}$ denote the set of best solutions in the flock. Line 31 moves the leader solution to the end and forwards its left or right successor to the leader position. Specifically, the leader solution is alternately moved to the left or right end. For example, if the leader solution in the current iteration is moved to the left end, the left successor is forwarded to the leader position. Then, the leader solution in the next

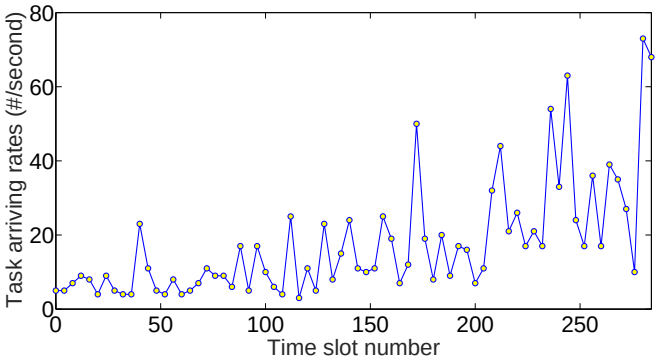


Figure 11.3 Task arriving rate.

iteration is moved to the right end, and the right successor is forwarded to the leader position. Line 32 sorts all solutions except the leader in the flock. Line 34 reduces temperature θ_2^g by θ_3, which denotes the temperature cooling rate. The **while** loop stops if g=$\hat{g}$. Line 36 returns the best solution (leader bird) in the flock, which is converted into decision variables including $[y_\tau^{1,1}, \cdots,$ and $y_\tau^{N_\tau^\dagger, N^0}, N_\tau^\dagger, \mu_\tau]$.

11.3 Performance Evaluation

This chapter evaluates the proposed SMBO with real-life data. SMBO is coded and implemented with MATLAB 2017 in a computer with an Intel Xeon CPU with 2.4 GHz and a 32-GB memory.

11.3.1 Parameter Setting

This chapter uses realistic tasks collected from Google cluster trace[1] to evaluate the proposed method. Figure 11.3 illustrates the task arriving rate in one day. In addition, this chapter uses realistic price of power grid collected from the New York Independent System Operator[2]. Figure 11.4 illustrates the price of power grid in one day. Besides, the length of one time slot is 300 seconds, *i.e.*, L=300 seconds.

In addition, the following parameters are set as follows. $\hat{\varepsilon}$=0.01, $N_\tau^\dagger$=30, $\hat{\hbar}_3$=3000 MB/s, $\hat{\hbar}_2$=1024 MB, and $\hat{\hbar}_1$=2048 MIPS. In addition, ς_3^s, ς_2^s and ς_1^s are

[1]https://github.com/google/cluster-data
[2]https://www.nyiso.com/

Algorithm 9 SMBO (SA-based Migrating Birds Optimization)

Initialize parameters including $|\mathbb{X}|$, $\tilde{M}$, M^0, $\mathcal{N}^*$ and $\hat{g}$
Initialize a population of $|\mathbb{X}|$ initial solutions randomly
Sort the initial flock according to their fitness values obtained with (11.28)
Organize them in a V formation according to Figure 11.2
$g \leftarrow 1$
$\theta_2^g \leftarrow \theta_2^0$
while $g \leq \hat{g}$ **do**
  $i \leftarrow 1$
  while $i \leq \mathcal{N}^*$ **do**
    Create $\tilde{M}$ neighbors for the leader solution and sort them
    $g = g + \tilde{M}$
    for $i \leftarrow 2$ to $|\mathbb{X}|$ **do**
      Create $\tilde{M} - M^0$ neighbors for each solution i and sort them
      $g = g + \tilde{M} - M^0$
    end for
    Move the best neighbor of the leader solution to $\overset{*}{\mathbb{X}}(1)$, and put the second-best and the third-best neighbors of the leader solution into the neighbor sets of two successors of the leader solution
    for $i \leftarrow 2$ to $|\mathbb{X}| - 2$ **do**
      Move the best neighbor of solution i to $\overset{*}{\mathbb{X}}(i)$, and put M^0 unused best neighbors of solution i into the neighbor set of solution $i+2$
    end for
    Move the best neighbor of solution $|\mathbb{X}| - 1$ to $\overset{*}{\mathbb{X}}(|\mathbb{X}| - 1)$, and put M^0 unused best neighbors of solution $|\mathbb{X}| - 1$ into the neighbor set of solution $|\mathbb{X}| - 1$
    Move the best neighbor of solution $|\mathbb{X}|$ to $\overset{*}{\mathbb{X}}(|\mathbb{X}|)$, and put M^0 unused best neighbors of solution $|\mathbb{X}|$ into the neighbor set of solution $|\mathbb{X}|$
    for $i \leftarrow 1$ to $|\mathbb{X}|$ **do**
      if $\overset{*}{\mathbb{X}}(i)$ is better than solution j **then**
        Update solution i with $\overset{*}{\mathbb{X}}(i)$
      else
        Use SA's metropolis rule to update solution i with (11.30)
      end if
    end for
    $i \leftarrow i + 1$
  end while
  Move the leader solution to the end and forward its left or right successor to the leader position
  Sort all solutions except the leader in the flock
  $g \leftarrow g + 1$
  $\theta_2^g \leftarrow \theta_2^{g-1} * \theta_3$
end while
Return the best solution (leader bird) in the flock

randomly produced in the range of (0,0.1), *i.e.*, $\varsigma_3^s \in (0,0.1)$ MB/s, $\varsigma_2^s \in (0,0.1)$ MB and $\varsigma_1^s \in (0,0.1)$ MIPS. In addition, $\dot{\mathbb{N}}$=0.05 tasks, $\hat{N}=2\times10^3$, $\hat{\mathbb{E}}=1\times10^5$ WH, $\hat{\Phi}$=600 W,

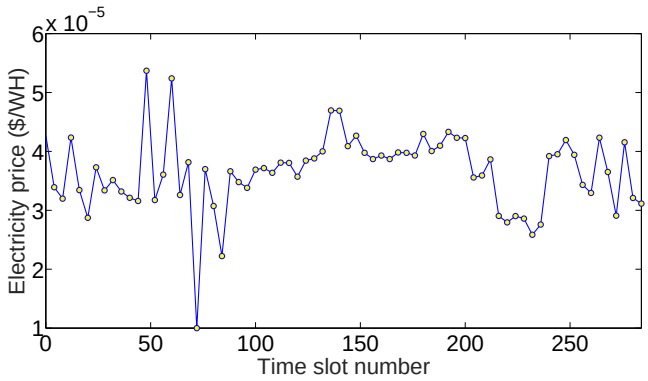


Figure 11.4 Price of power grid in one day.

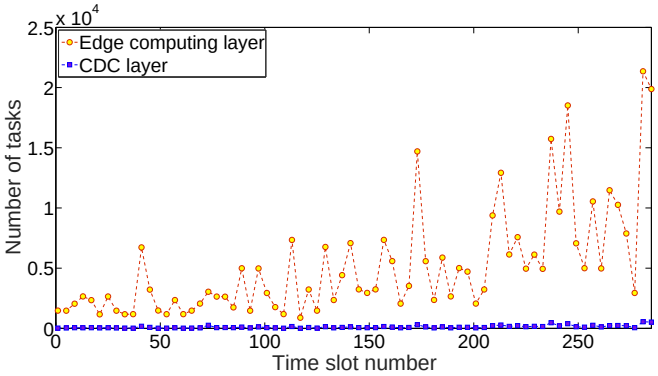


Figure 11.5 Number of tasks scheduled to edge computing and CDC layers.

$\check{\Phi}$=300 W, α=1.6 and $\hat{\partial}$=\$0.01. Besides, υ_τ is set to $\frac{1}{5}\Delta_\tau^{18}/\left(\mu_\tau\left(1-\Delta_\tau^{17}\right)\right)$ seconds. In addition, $\eta_\tau^{i,j}$ is randomly produced in the range of (0,0.01), *i.e.*, $f_{23}^{s,q}\in$\$(0,0.01). $\hat{T}^{\ddagger}$=0.1 seconds, $\hat{T}^{\dagger}$=0.1 seconds, L=300 seconds, $\hat{\mathbb{N}}^{\ddagger}$=30, N_τ^q=1000, $\varkappa_1^0$=0.3333, $\varkappa_2^0$=0.3333 and $\varkappa_3^0$=0.3333.

Besides, $\mathcal{N}_s$ is randomly produced in the range of (1×10^6, 8×10^6) processing operations, *i.e.*, $\mathcal{N}_s\in(1\times10^6,\ 8\times10^6)$. $\hbar_q$ is randomly produced in the range of (1×10^{11}, 2×10^{11}) processing operations per second, *i.e.*, $\mathcal{N}_s\in(1\times10^{11},\ 2\times10^{11})$. The parameters of SMBO are set as follows. $|\mathbb{X}|$=51, $\tilde{M}$=3, M^0=1, m=10, G=51^3, $\mathbb{T}_\tau^{\dagger}$=$5\times10^6$ and θ_3=0.985. In addition, $\overset{\infty}{\mathcal{N}}$=$10^{10}$.

11.3.2 Experimental Results

Figure 11.5 shows the number of tasks scheduled to edge computing, *i.e.*, local computing and CDC layers. It is shown that the number of tasks scheduled to the

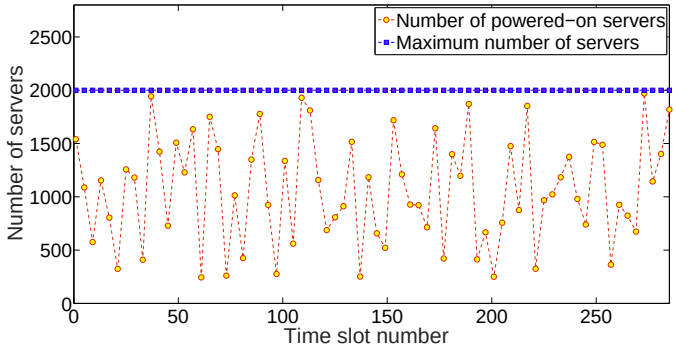


Figure 11.6 Number of switched-on servers in the CDC layer.

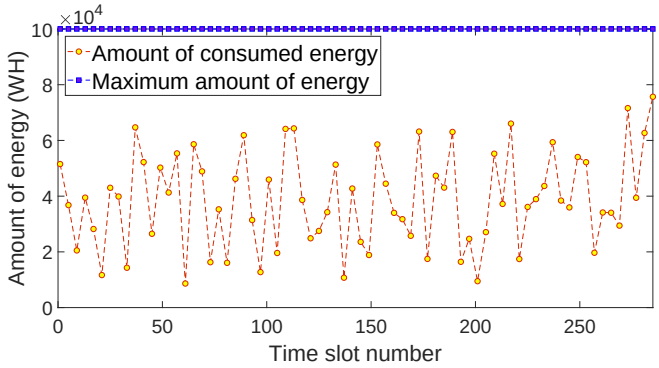


Figure 11.7 Amount of energy consumed in the CDC layer.

CDC layer is much lower than that scheduled to the edge layer in each time slot. The reason is that nodes in the edge layer are much closer to users and can avoid the transmission delay of input/output data to/from the CDC layer through MBS. Since CPU, memory, bandwidth, and storage resources, available energy, and the load balance of all heterogeneous nodes in the edge computing layer are all limited.

Therefore, a few tasks also need to be offloaded to execute in servers in the CDC layer but the number of offloaded tasks is much lower than that executed in nodes in the edge layer. Figure 11.6 shows the number of switched-on servers in the CDC layer. The maximum number of servers in the CDC layer is 2000, *i.e.*, $\hat{N}$=2000. It is shown that the number of switched-on servers in the CDC layer is less than its corresponding upper limit in each time slot.

Figure 11.7 shows the amount of energy consumed in CDC in each time slot. The maximum amount of the total available energy in the CDC layer is 1×10^5 WH,

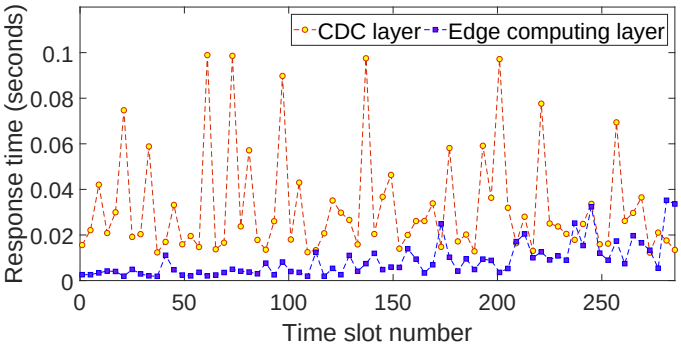


Figure 11.8 Total response time of each task in CDC and edge computing layers.

i.e., Δ=1×10^5 WH. It is shown that the amount of energy consumed in the CDC layer is less than its limit in each time slot. It demonstrates that the proposed SMBO dynamically consumes the energy in CDC and edge computing layers.

Figure 11.8 shows the total response time of each task executed in CDC and edge computing layers. The users' response time limits for each task scheduled to execute in the edge computing and CDC layers are both set to 0.1 seconds, *i.e.*, $\hat{T}^{\dagger}$=$\hat{T}^{\ddagger}$=0.1 seconds. Therefore, it is shown that the total response time of each task executed in CDC and edge computing layers is less than its limit in each time slot, *i.e.*, (11.6) and (11.9) are both met in each time slot. The results demonstrate that the execution results based on the obtained schedule can strictly meet response time limits of tasks.

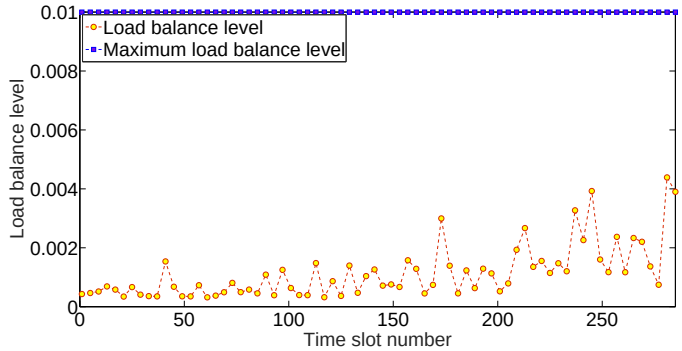


Figure 11.9 Load balance level of all nodes in edge computing layer.

Figure 11.9 shows the load balance level of all nodes in the edge computing layer in each time slot. The specified maximum load balance level in the edge

computing layer is 0.01, *i.e.*, $\hat{\varepsilon}$=0.01. It is shown that the load balance level of all nodes in the edge computing layer is less than its limit in each time slot. This result demonstrates that SMBO provides the load balance among nodes in the edge computing layer. The reason is that SMBO jointly considers CPU, memory, and bandwidth resource limits, the load balance needs of all nodes, and different processing capacities of heterogeneous nodes in the edge computing layer. The result in Figure 11.9 demonstrates that the proposed SMBO realizes the collaborative computation offloading and resource allocation for all tasks by jointly optimizing the computation offloading between CDC and edge computing layers, and specifying resource allocation in CDC layer.

11.3.3 Comparison Results

To validate SMBO, this chapter compares it with two typical meta-heuristic optimization algorithms including firefly algorithm (FA) [302] and Genetic Learning Particle Swarm Optimization (GL-PSO) [289], and two baseline algorithms including local computing and entire offloading. Here SMBO, FA and GL-PSO are repeated independently for 30 times to produce their respective results. The reasons for choosing them as benchmark algorithms are given as follows.

1) FA [302]: As an emerging meta-heuristic algorithm, FA can efficiently find high-quality optima of multimode functions. Each of its fireflies is independent of each other, and it is easy to be implemented in parallel. Its convergence speed is fast, but it suffers from a premature convergence problem. The oscillation in its search process happens repeatedly, and its accuracy is often unsatisfying for high-dimension optimization problems.

2) GL-PSO [289]: GL-PSO applies a genetic learning mechanism of a genetic algorithm (GA) to construct exemplars that are hybridized with particle swarm optimization (PSO) in a cascade manner. Then, particles in PSO are guided by the exemplars produced by GA. Thus, GA and PSO are integrated cohesively to diversify the search of particles thus guiding the population to find high-quality solutions. The search history of particles in PSO provides promising genetic information to GA and helps it reproduce high-quality exemplars.

3) Local computing. All arriving tasks of users are executed by nodes in the edge computing layer.

4) Entire offloading. All arriving tasks of users are offloaded and executed in servers in the CDC layer.

The comparison between SMBO and FA can demonstrate the former's convergence speed. The comparison between it and GL-PSO can demonstrate its solution accuracy. Besides, SMBO, FA and GL-PSO are all sensitive to the setting of their parameters. Therefore, similar to SMBO, several trials are conducted to determine the parameter setting for both FA and GL-PSO with the grid search method [303]. Besides, SMBO, FA and GL-PSO stop their search processes if their solutions are not improved in 10 consecutive iterations.

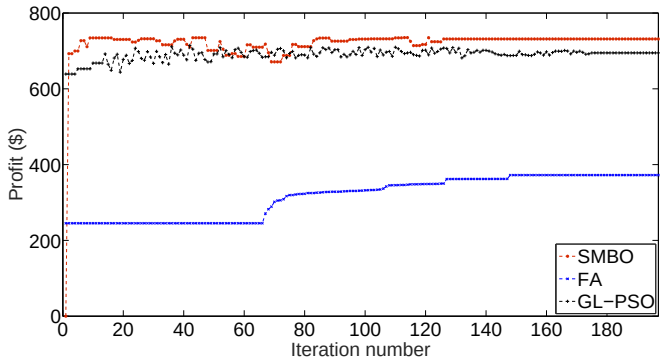


Figure 11.10 Profit of each iteration of SMBO, FA and GL-PSO in time slot 1.

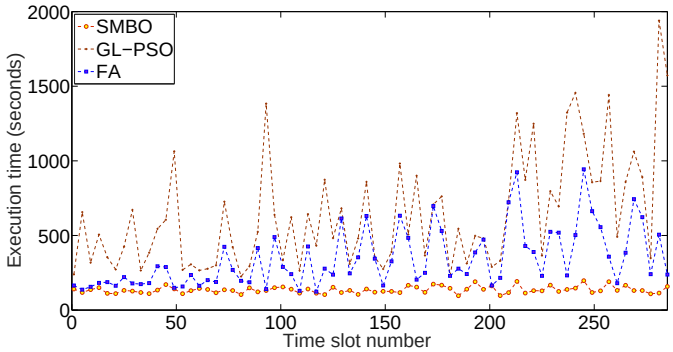


Figure 11.11 Convergence time of SMBO, FA and GL-PSO in each time slot.

Figure 11.10 shows the profit computed at each iteration of SMBO, FA and GL-PSO in time slot 1. Each iteration of SMBO refers to Lines 8–34 in Algorithm

9. The iterations of FA and GL-PSO are similar to those of SMBO. FA and GL-PSO require 148 and 195 iterations to stop their searches, and their final profits are $365.72 and $694.76, respectively. On the other hand, SMBO only requires 126 iterations to stop its search, and its final profit is $734.35. It is observed that compared to FA and GL-PSO, SMBO's average profit of all time slots is increased by 50.20% and 5.39% on average, respectively. Figure 11.11 shows the convergence time comparison of SMBO, FA and GL-PSO in each time slot. It is shown that compared to FA and GL-PSO, SMBO's average convergence time of all time slots is reduced by 49.26% and 72.21% on average, respectively. Therefore, SMBO increases the profit in less time and fewer iterations than FA and GL-PSO. Figures 11.10 and 11.11 prove that the adoption of SA's Metropolis acceptance rule in the proposed method increases the diversity of population and search performance.

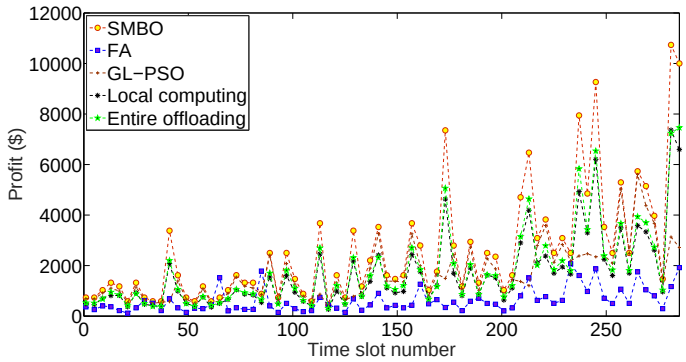


Figure 11.12 Profit of SMBO, FA, GL-PSO, local computing and entire offloading.

Figure 11.12 illustrates the profit comparison of SMBO, FA, GL-PSO, local computing and entire offloading. It is shown that compared with FA, GL-PSO, local computing and entire offloading, the profit of SMBO is increased by 66.83%, 21.32%, 34.81% and 30.22% on average, respectively. The reasons are given as follows. As is shown in Figure 11.10, the solution accuracy of SMBO is higher than those of FA, GL-PSO, and therefore, the profit of SMBO is larger than those of FA and GL-PSO. Local computing schedules all tasks of users to execute in nodes of the edge computing layer. It is worth noting that battery energy, CPU, memory and bandwidth resources

are all limited in nodes in the edge computing layer. Similarly, entire offloading schedules all tasks of users to execute in servers of the CDC layer. On the other hand, the maximum amount of energy, the maximum number of servers, *etc.* in the CDC layer are also limited. Thus, tasks scheduled with local computing and entire offloading need to wait for execution and suffer from higher latency, resulting in bad user experience and low profit. The experimental results validate that the proposed offloading method in SMBO outperforms these four benchmark methods on profit. This is because the proposed method jointly considers computation offloading between CDC and the edge, and CPU, memory and bandwidth allocation to nodes in the edge computing layer.

11.4 Summary

In recent years, edge computing has been gaining enormous attention, and is growingly adopted due to its fast response and close proximity to users when computing tasks are not intensive. However, its computing resources and energy capacity are limited. Cloud data centers (CDCs) have rich computation resources but are usually located in remote sites, and need both energy and time to transmit programs or data to CDCs and retrieve results from CDCs. Consequently, offloading tasks in the edge to CDCs is becoming a promising alternative to maximize the profit while enforcing user-specific response time limits of tasks. Based on an architecture including terminal, edge computing and CDC layers. This chapter proposes a profit-maximized collaborative computation offloading and resource allocation algorithm. It jointly optimizes the computation offloading between CDC and edge computing layers, and specifies resource allocation in CDC layer. In each time slot, a single-objective constrained optimization problem is formulated and solved by a proposed Simulated-annealing-based Migrating Birds Optimization (SMBO) algorithm. Realistic data-driven experimental results demonstrate that SMBO obtains larger profit than its several existing methods.

CHAPTER 12

CONCLUSIONS AND FUTURE WORK

12.1 Summary of Contributions

An increasing number of companies deploy their delay-constrained applications, *e.g.*, big data processing, deep learning, and high-performance computing, in various cloud data centers (CDCs). These applications share infrastructure resources in CDCs and provide services to users in a high-performance and cost-effective way. This inevitably increases the amount of energy consumed by these large-scale data centers. As an emerging architecture, edge computing provides real-time and scalable data processing capacities in the network edge, but it suffers from some challenging problems of limited battery and processing capacities. Therefore, intelligent task scheduling and resource allocation in CDCs and nodes in the edge are critically important in running them energy-efficiently and profit-sensitively. However, it is highly challenging to jointly achieve greenness, cost-effectiveness, energy efficiency and delay bound assurance due to dramatic growth, aperiodic arrival and heterogeneity of tasks. Consequently, this dissertation work proposes a series of intelligent task scheduling and resource allocation algorithms to address these challenges arising from distributed cloud and edge computing systems.

1) Contributions of the background work of this dissertation.

The background work has presented four intelligent optimization algorithms to solve above problems. **First**, Chapter 3 in the background work investigates the spatial diversity of bandwidth prices of Internet service providers (ISPs), power grid prices and the availability of renewable green energy to minimize the total cost of a CDC provider. Specifically, Chapter 3 formulates a nonlinear optimization problem and solves it by a proposed simulated-annealing-based bat algorithm. In this way, Chapter 3 proposes a Spatial Task Scheduling and Resource Optimization

(STSRO) method to minimize the total cost of a CDC provider by exploiting such spatial diversity in CDCs. STSRO cost-effectively schedules all arriving tasks of heterogeneous applications while strictly meeting their delay bound constraints. Real-life data-driven experimental results demonstrate that STSRO drastically increases the throughput and reduces the total cost of the CDC provider in comparison with cheap-electricity-first and renewable energy-first scheduling methods provided that delay bound constraints of all tasks are strictly satisfied.

Second, Chapter 4 in the background work proposes a Geography-Aware Task Scheduling (GATS) approach to maximize total profit of the CDC provider by jointly and optimally determining the allocation of tasks of all applications among multiple ISPs and task service rates of servers in CDCs. GATS considers spatial variations of ISP bandwidth prices, availability of green energy, prices of power grid, and revenue brought by the execution of tasks. Specifically, Chapter 4 formulates a profit maximization problem as a convex optimization one solved by an interior point method. GATS intelligently schedules tasks of all applications in comparison with two typical task scheduling methods. Simulation results show that GATS increases the total profit and throughput of the CDC provider compared to two typical task scheduling approaches.

Third, Chapter 5 in the background work proposes a novel Spatio-Temporal Task Scheduling (STTS) method that exploits the spatial and temporal variations in prices of power grid and availability of green energy in CDCs. STTS jointly investigates the spatial and temporal variations in prices of power grid and availability of renewable energy in CDCs within all tasks' delay bound constraints, and aims to minimize energy cost of CDC providers by cost-effectively scheduling all arriving tasks of heterogeneous applications among multiple CDCs. In each iteration of STTS, the energy cost for the CDC provider is formulated and solved with a hybrid meta-heuristic algorithm named Genetic Simulated-annealing-based Particle Swarm

Optimization (GSPSO). Extensive trace-driven experimental results demonstrate that it achieves higher throughput and lower energy cost for the CDC provider than simulated annealing, particle swarm optimization, bat algorithm and Spatio-Temporal Load Balancing (STLB) methods, respectively while strictly meeting delay bound constraints of all arriving tasks of heterogeneous applications.

Fourth, Chapter 6 in the background work considers the temporal variations in prices of power grid, revenue, solar irradiance, wind speed and prices of public clouds. Chapter 6 presents a Temporal Task Scheduling (TTS) algorithm that investigates such temporal variations, and cost-effectively schedules all tasks to the CDC and public clouds within delay bound constraints. Specifically, Chapter 6 formulates a profit maximization problem for a green hybrid CDC provider, and solves it with the proposed GSPSO algorithm. Extensive simulation experiments are conducted to demonstrate that the proposed TTS outperforms cheap-electricity-first, immediate and renewable energy-first scheduling methods, respectively in terms of profit.

2) Contributions of this dissertation.

First, Chapter 7 in this dissertation work proposes a Profit and Quality of service (QoS)-optimized Task Scheduling (PQTS) method to achieve a beneficial tradeoff between these two objectives including the profit maximization for CDC providers, and the average task loss possibility minimization for users' applications. Specifically, this dissertation work formulates a bi-objective optimization problem and solves it with a Simulated-annealing-based Bi-objective Differential Evolution (SBDE) algorithm. The minimum Manhattan distance is used to specify a knee solution that determines proper task split among ISPs and task service rates at CDCs in each time slot. Real-life data-based simulations are conducted to reveal that the proposed method achieves significantly higher profit and lower average task loss possibility of all applications than three existing task scheduling algorithms including

electricity-cost-aware distributed task scheduling, profit maximization approach and green task balancing, respectively.

Second, Chapter 8 in this dissertation work formulates the joint optimization of energy cost and QoS as a bi-objective constrained optimization problem. It is solved by a Simulated-annealing-based Adaptive Differential Evolution (SADE) algorithm to obtain a close-to-Pareto-optimal set. In this way, it properly allocates arriving tasks among CDCs, and changes task service rates of each CDC in each time slot. Real-life data-based results demonstrate that SADE reduces energy cost and response time of tasks compared with Non-dominated Sorting Genetic Algorithm 2 (NSGA2) and multi-objective evolutionary algorithm based on decomposition.

Third, Chapter 9 in this dissertation work formulates a constrained bi-objective optimization problem, which is solved with an Improved Multiobjective Evolutionary Algorithm based on Decomposition (IMEAD). The obtained result determines a high-quality balance between maximizing the revenue of green CDC providers and minimizing their energy cost of heterogeneous applications. It jointly determines the optimal split of all arriving tasks of each application among multiple CDCs, and specifies the optimal task service rate of each server in each CDC at each time slot. Simulations prove that it increases the profit of CDC providers and reduces the convergence time in comparison with NSGA2 and strength pareto evolutionary algorithm 2.

Fourth, Chapter 10 in this dissertation work proposes a Fine-grained Spatial Task Scheduling (FSTS) algorithm to achieve the energy cost minimization for a provider of distributed CDCs by optimally allocating tasks of heterogeneous applications among multiple CDCs, and specifying the running speed of each server and the number of switched-on servers in each CDC. FSTS uses a $G/G/1$ queuing model to analyze the performance of servers, and further formulates a constrained optimization problem. It is solved by a newly proposed simulated-annealing-based

bees algorithm to find a close-to-optimal solution. Real-life data-driven experiments demonstrate that FSTS can decrease energy cost and ensure the highest throughput in comparison with bees algorithm, Genetic Learning Particle Swarm Optimization (GL-PSO), STLB, and spatial task scheduling, respectively.

Fifth, Chapter 11 in this dissertation work proposes a profit-maximized collaborative computation offloading and resource allocation algorithm. It jointly optimizes the computation offloading between CDC and edge computing layers, and specifies resource allocation in the CDC layer. It jointly considers CPU, memory, and bandwidth resource limits, load balance requirements of all nodes, and different processing capacities of heterogeneous nodes in the edge computing layer. In addition, it jointly considers the maximum amount of energy, maximum number of available servers and task queue stability of servers in the CDC layer. In each time slot, a single-objective constrained optimization problem is formulated and solved by a proposed Simulated-annealing-based Migrating Birds Optimization (SMBO) algorithm. Realistic data-driven experimental results demonstrate that SMBO obtains larger profit than its firefly algorithm, GL-PSO, local computing and entire offloading, respectively while meeting response time limits of tasks. It can be readily implemented and incorporated into large-scale industrial computing systems.

12.2 Limitations

The limitations of the proposed algorithms are discussed here. **First**, this dissertation work assumes that the future information, *e.g.*, arriving rates of tasks, solar irradiance, wind speed, prices of power grid and unit bandwidth prices, are already known at the start of each time slot. However, in real CDCs, the information prediction typically requires time to determine the prediction models based on historical data. This dissertation ignores the information prediction time. Thus, some future information may be unavailable when the proposed algorithms start to run. This may cause the unexpected results that delay bounds of some tasks cannot be guaranteed. The

recently emerging big data technologies, *e.g.*, deep learning-based and other prediction algorithms [304]–[311], and the wide deployment of high-performance clusters in CDCs make it possible that the prediction of above information can be performed in a real-time manner, and negligible. But their actual prediction performance needs to be further evaluated.

Second, this dissertation adopts the Google cluster trace to evaluate the proposed algorithms. Similar to the work in [110], it is assumed that the task service time conforms to the exponential distribution. Many existing studies [312]–[314] adopt the $M/M/1$ queueing model to evaluate the performance of their proposed methods with simulators or a realistic cluster of servers in data centers. Their results demonstrate that the Google trace data can effectively and well approximate the behaviors of real data centers. Nevertheless, the $M/M/1$ queueing system has strong assumption of the exponential task service time. Actually, several more accurate and advanced queueing models, *e.g.*, $G/D/1$, $G/G/c$, $M/G/1$, $G/M/1$ and Pareto queues with the tailed distribution assumption [306], could be adopted for the performance analysis of CDCs and edge computing systems. In addition, some hybrid queueing models combine many different queueing ones together to improve the accuracy of performance modeling for cloud and edge computing systems [114, 307, 308, 309]. They should be adopted to evaluate some of our proposed methods.

Third, similar to the work in [24], this dissertation determines the setting of parameters of each meta-heuristic optimization algorithm with multiple trials based on the parameter setting in previous studies. Nevertheless, all meta-heuristic optimization algorithms are sensitive to the setting of their parameters [23]. It currently lacks the comprehensive parameter sensitivity analysis, and the optimal setting of parameters might not be obtained. Therefore, a thorough sensitivity analysis of parameters of all algorithms would further improve the current scheduling precision and cost-effectiveness of tasks for distributed cloud and edge computing

systems [315, 316, 317]. In this way, the energy cost of CDC and edge providers, and performance of tasks of delay-constrained applications could be further optimized.

12.3 Future Research

The future work should add some extensions to this dissertation work from the following two aspects. **First**, this dissertation work considers simple applications whose tasks are executed independently and does not rely on the execution of other applications. On the contrary, there are many workflow applications in CDCs and edge computing, such as scientific workflows [267, 306, 318, 319, 320] and complex engineering analysis. Each workflow application consists of multiple application subsystems, and its execution often involves the execution of multiple tasks. In addition, the interactions among tasks are complex, and the execution of each task often depends on the execution results of other tasks with which it interacts. Besides, to ensure the quality of service specified by users, the CDC provider and the users often sign service level agreements (SLAs). If the actual execution time of users' tasks of workflow applications exceeds its corresponding threshold specified in SLAs, a CDC provider needs to pay users a high penalty fee. On the other hand, in an actual CDC, each task contains complex conditions, loops and other program structures. In addition, resources, *e.g.*, CPU, memory and I/O, of each server in CDCs are often shared by multiple virtual machines (VMs). This resource sharing mode makes the performance of each VM change with time. Therefore, it is often difficult to effectively predict the execution time of each task in a particular VM for a workflow application. In addition, the size of execution data of tasks in typical CDCs is huge. For example, the data currently processed in an industrial Internet platform, INDICS [321], is nearly 6000TB. Therefore, in the big data environment, to improve the responsiveness of CDCs to users, it is urgent to design an effective big data-driven task execution time prediction method. Specifically, the future work should investigate how to utilize the time autocorrelation in task execution time data

of VMs, and the local spatial dependence between each VM and its nearby VMs with deep learning methods, *e.g.*, rough stacked denoising autoencoder [322, 323] and long short-term memory [324, 325]. According to the information, the valuable temporal and spatial features could be automatically extracted from the task execution time data, and the execution time of tasks in VMs would be effectively predicted.

Second, there are many types of complex workflow applications in CDCs [326], *e.g.*, complex cloud simulation workflows and cloud manufacturing workflows [318, 319], [327]–[329]. These workflows require various resources, *e.g.*, software, data, information, models, knowledge and VMs in CDCs. Each workflow application often requires collaborative interaction among multiple tasks to achieve users' objectives. CDCs adopt a pay-as-you-go method to dynamically provide various types of resources for users around the world through networks. The economies of scale of cloud computing have attracted more and more users to deploy their different types of complex workflow applications in CDCs, *e.g.*, Google, Amazon and Microsoft. Therefore, for each CDC provider, as types and number of workflow applications in CDCs grow rapidly, it becomes a very important and challenging problem of how to maximize the profit of a CDC provider while ensuring the latency requirements of users' tasks of workflow applications. The future work will should investigate how to utilize execution patterns of tasks of workflow applications in VMs, dependency among tasks, and the profit relation between workflow applications and CDC providers. Then, the profit of CDC providers would be maximized by intelligently scheduling and dispatching users' tasks to VMs. Its applications to other optimization problems [322, 323], [330]–[341] should also be pursued.

Third, all the proposed solution algorithms are based on the novel combinations of mature intelligent optimization methods, *i.e.*, bat algorithm, bees algorithm, simulated annealing, differential evolution, genetic algorithm, particle swarm optimization, multi-objective evolutionary algorithm and migrating birds optimization.

A common issue for them is that their performance tends to be sensitive to many user-defined parameters and the optimal parameter settings are problem-dependent. Hence, their further analysis and investigation are required. Some newly developed meta-heuristics, *e.g.*, [342]–[350] should be tried. It is critically important to determine the best parameter setting for each meta-heuristic optimization algorithm. There are many classic, proven, and highly useful methods, *e.g.*, the Taguchi's experimental design method [195], which could be tried to find an optimal reasonable combination of the user-defined parameters. In addition, there are some grid-based parameter adaptation methods, *e.g.*, a systematic parameter-search method called support vector machine-Grid [287] and a grid-based approach [351]. The grid-based methods can investigate the parameter selection toward the optimal direction while keeping an extensive distribution of different parameters in the evolutionary process. In addition, learning-based methods, *e.g.*, stacked autoencoder deep neural network [352] should be also adopted to search better parameter setting. Stacked autoencoder is efficient to reduce the dimension of parameter values especially for large-scale optimization problems, and keep their important features. In this way, it can specify the optimal parameter setting and speed up the search process of each meta-heuristic optimization algorithm.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D.Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia., "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers," in *Proc. of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, 2010, pp. 577-578.

[3] Q. Fan and N. Ansari, "On Cost Aware Cloudlet Placement for Mobile Edge Computing," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 926–937, Jul. 2019.

[4] P. Zhang, M. Zhou and G. Fortino, "Security and Trust Issues in Fog Computing: A Survey," *Future Generation Computer Systems*, vol. 88, pp. 16–27, Nov. 2018.

[5] K. Hwang and D. Li, "Trusted Cloud Computing with Secure Resources and Data Coloring," *IEEE Internet Computing*, vol. 14, no. 5, pp. 14–22, Oct. 2010.

[6] M. H. Ghahramani, M. C. Zhou and C. T. Hon, "Toward Cloud Computing QoS Architecture: Analysis of Cloud Systems and Cloud Services," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 5–17, Jan. 2017.

[7] X. Zhu, J. Li, X. Chen and M. C. Zhou, "Minimum Cost Deployment of Heterogeneous Directional Sensor Networks for Differentiated Target Coverage," *IEEE Sensor Journal*, vol. 17, no. 15, pp. 4938–4952, Aug. 2017.

[8] D. A. Chekired and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3312–3321, Dec. 2017.

[9] D. Yu and J. Li. "Recent Progresses in Deep Learning Based Acoustic Models," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 396–409, Jul. 2017.

[10] Z. Ren, K. Qian, Z. X. Zhang, V. Pandit, A. Baird and B. Schuller, "Deep Scalogram Representations for Acoustic Scene Classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 662–669, May 2018.

[11] Data Center Energy Use in U.S., U.S. Energy Information Administration (EIA), Washington, DC, Available: `http://www.eia.gov/` (accessed on May 10, 2019).

[12] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu and H. Tenhunen, "Energy-Aware VM Consolidation in Cloud Data Centers Using Utilization Prediction Model," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 524–536, Apr. 2019.

[13] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, "Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 377–391, Sept. 2015.

[14] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, Sept. 2009.

[15] A. Greenberg, J. Hamilton, D. A. Maltz and P. Patel,"The Cost of a Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, Jan. 2009.

[16] Z. A. Mann, "Allocation of Virtual Machines in Cloud Data Centers-A Survey of Problem Models and Optimization Algorithms," *ACM Computing Surveys*, vol. 48, no. 1, pp. 11:1–11:34, Aug. 2015.

[17] C. Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill and M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-Driven WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 15–26, Oct. 2013.

[18] H. Yuan, J. Bi, W. Tan and B. H. Li, "CAWSAC: Cost-Aware Workload Scheduling and Admission Control for Distributed Cloud Data Centers," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 976–985, Apr. 2015.

[19] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan and B. Christian, "Optimizing Cost and Performance in Online Service Provider Networks," in *Proceedings of 7th USENIX Symposium on Networked Systems Design and Implementation*, 2010, pp. 33–48.

[20] U. Franke and M. Buschle, "Experimental Evidence on Decision-Making in Availability Service Level Agreements," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 58–70, Mar. 2016.

[21] J. Luo, L. Rao and X. Liu, "Temporal Load Balancing with Service Delay Guarantees for Data Center Energy Cost Optimization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 775–784, Mar. 2014.

[22] A. Kiani and N. Ansari, "A Fundamental Tradeoff Between Total and Brown Power Consumption in Geographically Dispersed Data Centers," *IEEE Communications Letters*, vol. 20, no. 10, pp. 1955–1958, Oct. 2016.

[23] A. Kiani and N. Ansari, "Profit Maximization for Geographically Dispersed Green Data Centers," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 703–711, Mar. 2018.

[24] M. Ghamkhari and H. Mohsenian-Rad, "Energy and Performance Management of Green Data Centers: A Profit Maximization Approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, Jun. 2013.

[25] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[26] B. Pravallika and B. Shirisha, "A Survey on Green Cloud Computing," *International Journal of Research*, vol. 5, no. 12, pp. 3802–3817, Oct. 2015.

[27] M. Fazio, A. Celesti, R. Ranjan, C. Liu, L. Chen and M. Villari, "Open Issues in Scheduling Microservices in the Cloud," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 81–88, Oct. 2016.

[28] E. Baccarelli, N. Cordeschi, A. Mei, M. Panella, M. Shojafar and J. Stefa, "Energy-efficient Dynamic Traffic Offloading and Reconfiguration of Networked Data Centers for Big Data Stream Mobile Computing: Review, Challenges, and a Case Study," *IEEE Network*, vol. 30, no. 2, pp. 54–61, Apr. 2016.

[29] P. Pierleoni, R. Concetti, A. Belli and L. Palma, "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison," *IEEE Access*, vol. 8, pp. 5455–5470, Dec. 2019.

[30] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres and H. Tenhunen, " Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187–198, Mar. 2015.

[31] J. Bi, H. Yuan, W. Tan and B. H. Li, "TRS: Temporal Request Scheduling with Bounded Delay Assurance in a Green Cloud Data Center," *Information Sciences*, vol. 360, no. 1, pp. 57–72, Sept. 2016.

[32] R. V. den Bossche, K. Vanmechelen and J. Broeckhove, "Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 973–985, Jun. 2013.

[33] Y. Niu, F. Liu, X. Fei and B. Li, "Handling Flash Deals with Soft Guarantee in Hybrid Cloud," in *Proceedings of IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[34] M. Al-Ayyoub, M. Al-Quraan, Y. Jararweh and E. BENKHELIFA, "Resilient Service Provisioning in Cloud Computing Data Centers," *Future Generation Computer Systems*, vol. 86, pp. 765–774, Sept. 2018.

[35] J. Shuja, K. Bilal, S. A. Madani, M. Othman, R. Ranjan, P. Balaji and S. U. Khan, "Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers," *IEEE Systems Journal*, vol. 10, no. 2, pp. 507–519, Jun. 2016.

[36] L. Gu, D. Zeng, A. Barnawi, S. Guo and I. Stojmenovic, "Optimal Task Placement with QoS Constraints in Geo-Distributed Data Centers Using DVFS," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 2049–2059, Jul. 2015.

[37] L. Wei, X. Li, R. Fan, H. Sun and Z. Hu, "A Hybrid Multiobjective Particle Swarm Optimization Algorithm Based on R2 Indicator," *IEEE Access*, vol. 6, pp. 14710–14721, Mar. 2018.

[38] M. Saez, F. P. Maturana, K. Barton and D. M. Tilbury, "Real-Time Manufacturing Machine and System Performance Monitoring Using Internet of Things," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1735–1748, Oct. 2018.

[39] A. Botta, W. De Donato, V. Persico and A. Pescapé, "Integration of Cloud Computing and Internet of Things: A Survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, Mar. 2016.

[40] L. Wang and R. Ranjan, "Processing Distributed Internet of Things Data in Clouds," *IEEE Cloud Computing*, vol. 2, no. 1, pp. 76–80, Feb. 2015.

[41] M. Ghahramani, M. Zhou and C. T. Hon, "Mobile Phone Data Analysis: A Spatial Exploration Toward Hotspot Detection," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 351–362, Jan. 2019.

[42] Y. Deng, Z. Chen, X. Yao, S. Hassan and A. M. A. Ibrahim, "Parallel Offloading in Green and Sustainable Mobile Edge Computing for Delay-Constrained IoT System," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12202–12214, Dec. 2019.

[43] P. Wang, C. Yao, Z. Zheng, G. Sun and L. Song, "Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2872–2884, Apr. 2019.

[44] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[45] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[46] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing and Y. Zhang, "Selective Offloading in Mobile Edge Computing for the Green Internet of Things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, Feb. 2018.

[47] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin and W. Dai, "Energy Efficient Task Allocation and Energy Scheduling in Green Energy Powered Edge Computing," *Future Generation Computer Systems*, vol. 95, pp. 89–99, Jun. 2019.

[48] X. Sun and N. Ansari, "Green Cloudlet Network: A Sustainable Platform for Mobile Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 180–192, Jan. 2020.

[49] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[50] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach and F. Giust, "Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[51] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, "Multiobjective Optimization for Computation Offloading in Fog Computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[52] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[53] K. Gai, M. Qiu, H. Zhao and X. Sun, "Resource Management in Sustainable Cyber-Physical Systems Using Heterogeneous Cloud Computing," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 2, pp. 60–72, Apr. 2018.

[54] Y. Lin, E. T. -. Chu, Y. Lai and T. Huang, "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds," *IEEE Systems Journal*, vol. 9, no. 2, pp. 393–405, Jun. 2015.

[55] M. Gaggero and L. Caviglione, "Model Predictive Control for Energy-Efficient, Quality-Aware, and Secure Virtual Machine Placement," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 420–432, Jan. 2019.

[56] X. T. R. Kong, S. X. Xu, M. Cheng and G. Q. Huang, "IoT-Enabled Parking Space Sharing and Allocation Mechanisms," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1654–1664, Oct. 2018.

[57] J. Zhao, Q. Li, Y. Gong and K. Zhang, "Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[58] T. Mareda, L. Gaudard and F. Romerio, "A Parametric Genetic Algorithm Approach to Assess Complementary Options of Large Scale Wind-solar Coupling,"

IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 2, pp. 260–272, Apr. 2017.

[59] S. Lyden and M. E. Haque, "A Simulated Annealing Global Maximum Power Point Tracking Approach for PV Modules Under Partial Shading Conditions," *IEEE Transactions on Power Electronics*, vol. 31, no. 6, pp. 4171–4181, Jun. 2016.

[60] M. Campos, R. A. Krohling and I. Enriquez, "Bare Bones Particle Swarm Optimization With Scale Matrix Adaptation," *IEEE Transactions on Cybernetics*, vol. 44, no. 9, pp. 1567–1578, Sept. 2014.

[61] H. Shah-Mansouri, V. W. S. Wong and R. Schober, "Joint Optimal Pricing and Task Scheduling in Mobile Cloud Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5218–5232, Aug. 2017.

[62] T. Maqsood, N. Tziritas, T. Loukopoulos, S. A. Madani, S. U. Khan and C. Z. Xu, "Leveraging on Deep Memory Hierarchies to Minimize Energy Consumption and Data Access Latency on Single-Chip Cloud Computers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 154–166, Apr. 2017.

[63] M. Nir, A. Matrawy and M. St-Hilaire, "Economic and Energy Considerations for Resource Augmentation in Mobile Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 99–113, Jan. 2018.

[64] Y. W. Chen and J. M. Chang, "EMaaS: Cloud-Based Energy Management Service for Distributed Renewable Energy Integration," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2816–2824, Nov. 2015.

[65] C. Canali, L. Chiaraviglio, R. Lancellotti and M. Shojafar, "Joint Minimization of the Energy Costs from Computing, Data Transmission, and Migrations in Cloud Data Centers," *IEEE Transactions on Green Communications and Networking*, vol. PP, no. 99, pp. 1-16, Jan. 2018.

[66] W. Chen, I. Paik and Z. Li, "Cost-Aware Streaming Workflow Allocation on Geo-Distributed Data Centers," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 256-271, Feb. 2017.

[67] Z. Hu, B. Li and J. Luo, "Time- and Cost-Efficient Task Scheduling across Geo-Distributed Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 705–718, Mar. 2018.

[68] L. Yu, T. Jiang, Y. Zou and Z. Sun, "Joint Energy Management Strategy for Geo-Distributed Data Centers and Electric Vehicles in Smart Grid Environment," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2378–2392, Sept. 2016.

[69] K. Wu, P. Lu and Z. Zhu, "Distributed Online Scheduling and Routing of Multicast-Oriented Tasks for Profit-Driven Cloud Computing," *IEEE Communications Letters*, vol. 20, no. 4, pp. 684–687, Apr. 2016.

[70] L. Ismail and H. Materwala, "Energy-Aware VM Placement and Task Scheduling in Cloud-IoT Computing: Classification and Performance Evaluation," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5166–5176, Dec. 2018.

[71] P. Varshney and Y. Simmhan, "AutoBoT: Resilient and Cost-Effective Scheduling of a Bag of Tasks on Spot VMs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1512–1527, Jul. 2019.

[72] N. Kumar, G. S. Aujla, S. Garg, K. Kaur, R. Ranjan and S. K. Garg, "Renewable Energy-Based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2947–2957, May 2019.

[73] S. Hsieh, C. Chen, C. Chen, T. Yen, H. Hsiao and R. Buyya, "Novel Scheduling Algorithms for Efficient Deployment of MapReduce Applications in Heterogeneous Computing Environments," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1080–1095, Dec. 2018.

[74] K. K. Nguyen and M. Cheriet, "Environment-Aware Virtual Slice Provisioning in Green Cloud Environment," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 507–519, May 2015.

[75] M. Qiu, Z. Ming, J. Li, K. Gai and Z. Zong, "Phase-Change Memory Optimization for Green Cloud with Genetic Algorithm," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3528–3540, Dec. 2015.

[76] X. Deng, D. Wu, J. Shen and J. He, "Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters," *IEEE Systems Journal*, vol. 10, no. 1, pp. 78–87, Mar. 2016.

[77] M. S. Hasan, Y. Kouki, T. Ledoux and J. Pazat, "Exploiting Renewable Sources: When Green SLA Becomes a Possible Reality in Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 249–262, Apr. 2017.

[78] Y. Yang, X. Chang, J. Liu and L. Li, "Towards Robust Green Virtual Cloud Data Center Provisioning," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 168–181, Apr. 2017.

[79] A. Khosravi, L. L. H. Andrew and R. Buyya, "Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183–196, Apr. 2017.

[80] R. Tripathi, S. Vignesh and V. Tamarapalli, "Optimizing Green Energy, Cost, and Availability in Distributed Data Centers," *IEEE Communications Letters*, vol. 21, no. 3, pp. 500–503, Mar. 2017.

[81] M. Anastasopoulos, A. Tzanakaki and D. Simeonidou, "Stochastic Energy Efficient Cloud Service Provisioning Deploying Renewable Energy Sources," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3927–3940, Dec. 2016.

[82] D. Cheng, J. Rao, C. Jiang and X. Zhou, "Elastic Power-Aware Resource Provisioning of Heterogeneous Workloads in Self-Sustainable Datacenters," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 508–521, Feb. 2016.

[83] R. Rahmani, I. Moser and A. L. Cricenti, "Modelling and Optimization of Microgrid Configuration for Green Data Centres: A Metaheuristic Approach," *Future Generation Computer Systems*, vol. 108, pp. 742–750, Jul. 2020.

[84] Y. Wu, M. Tornatore, S. Ferdousi and B. Mukherjee, "Green Data Center Placement in Optical Cloud Networks," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 3, pp. 347–357, Sept. 2017.

[85] J. Yang, W. Xiao, C. Jiang, M. S. Hossain, G. Muhammad and S. U. Amin, "AI-Powered Green Cloud and Data Center," *IEEE Access*, vol. 7, pp. 4195–4203, Dec. 2018.

[86] L. Zhang, T. Han and N. Ansari, "Energy-Aware Virtual Machine Management in Inter-Datacenter Networks Over Elastic Optical Infrastructure," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 1, pp. 305–315, Mar. 2018.

[87] J. Mei, K. Li, Z. Tong, Q. Li and K. Li, "Profit Maximization for Cloud Brokers in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 190–203, Jan. 2019.

[88] Y. Ma, W. Liang, Z. Xu and S. Guo, "Profit Maximization for Admitting Requests with Network Function Services in Distributed Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1143–1157, May 2019.

[89] X. Zhang, Z. Huang, C. Wu, Z. Li and F. C. M. Lau, "Online Auctions in IaaS Clouds: Welfare and Profit Maximization With Server Costs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1034–1047, Apr. 2017.

[90] J. Wan, R. Zhang, X. Gui and B. Xu, "Reactive Pricing: An Adaptive Pricing Policy for Cloud Providers to Maximize Profit," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 941–953, Dec. 2016.

[91] Y. Wang, X. Lin and M. Pedram, "A Stackelberg Game-Based Optimization Framework of the Smart Grid With Distributed PV Power Generations and Data Centers," *IEEE Transactions on Energy Conversion*, vol. 29, no. 4, pp. 978–987, Dec. 2014.

[92] S. Ren and M. van der Schaar, "Dynamic Scheduling and Pricing in Wireless Cloud Computing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, pp. 2283–2292, Oct. 2014.

[93] K. Patel, M. Annavaram and M. Pedram, "NFRA: Generalized Network Flow-Based Resource Allocation for Hosting Centers," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1772–1785, Sept. 2013.

[94] A. Hammoud, H. Otrok, A. Mourad, O. Abdel Wahab and J. Bentahar, "On the Detection of Passive Malicious Providers in Cloud Federations," *IEEE Communications Letters*, vol. 23, no. 1, pp. 64–67, Jan. 2019.

[95] H. Deng, L. Huang, H. Xu, X. Liu, P. Wang and X. Fang, "Revenue Maximization for Dynamic Expansion of Geo-distributed Cloud Data Centers," *IEEE Transactions on Cloud Computing*. doi: 10.1109/TCC.2018.2808351, Feb. 2018.

[96] W. Wu, J. Wang, K. Lu, W. Qi, F. Shan and J. Luo, "Providing Service Continuity in Clouds under Power Outage," *IEEE Transactions on Services Computing*. doi: 10.1109/TSC.2017.2728795.

[97] S. Benbrahim, A. Quintero and M. Bellaïche, "Live Placement of Interdependent Virtual Machines to Optimize Cloud Service Profits and Penalties on SLAs," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 237–249, Jan. 2019.

[98] X. Mao, C. Li, W. Yan and S. Du, "Optimal Scheduling Algorithm of MapReduce Tasks Based on QoS in the Hybrid Cloud," in *Proceedings of 17th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2016, pp. 119–124.

[99] X. Zuo, G. Zhang and W. Tan, "Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, Apr. 2014.

[100] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea and J. Kołodziej, "Resource-Aware Hybrid Scheduling Algorithm in Heterogeneous Distributed Computing," *Future Generation Computer Systems*, vol. 51, no. 1, pp. 61–71, Oct. 2015.

[101] T. A. L. Genez, L. F. Bittencourt, N. L. S. d. Fonseca and E. R. M. Madeira, "Estimation of the Available Bandwidth in Inter-Cloud Links for Task Scheduling in Hybrid Clouds," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 62–74, Jan. 2019.

[102] J. Zhu, X. Li, R. Ruiz and X. Xu, "Scheduling Stochastic Multi-Stage Jobs to Elastic Hybrid Cloud Resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1401–1415, Jun. 2018.

[103] G. V. Prasad, A. S. Prasad and S. Rao, "A Combinatorial Auction Mechanism for Multiple Resource Procurement in Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 904–914, Oct. 2018.

[104] J. Hwang, "Toward Beneficial Transformation of Enterprise Workloads to Hybrid Clouds," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 295–307, Jun. 2016.

[105] P. Lu, Q. Sun, K. Wu and Z. Zhu, "Distributed Online Hybrid Cloud Management for Profit-Driven Multimedia Cloud Computing," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1297–1308, Aug. 2015.

[106] F. B. Charrada and S. Tata, "An Efficient Algorithm for the Bursting of Service-Based Applications in Hybrid Clouds," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 357–367, Jun. 2016.

[107] X. Qiu, H. Li, C. Wu, Z. Li and F. C. M. Lau, "Cost-Minimizing Dynamic Migration of Content Distribution Services into Hybrid Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3330–3345, Dec. 2015.

[108] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang and M. R. Lyu, "Towards Operational Cost Minimization in Hybrid Clouds for Dynamic Resource Provisioning with Delay-Aware Optimization," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 398–409, Jun. 2015.

[109] P. Zhang and M. Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 772–783, Apr. 2018.

[110] J. Yao, H. Guan, J. Luo, L. Rao and X. Liu, "Adaptive Power Management through Thermal Aware Workload Balancing in Internet Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2400–2409, Sept. 2015.

[111] H. Shao, L. Rao, Z. Wang, X. Liu, Z. Wang and K. Ren, "Optimal Load Balancing and Energy Cost Management for Internet Data Centers in Deregulated Electricity Markets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2659–2669, Oct. 2014.

[112] C. Liu, K. Li, K. Li and R. Buyya, "A New Cloud Service Mechanism for Profit Optimizations of a Cloud Provider and Its Users," *IEEE Transactions on Cloud Computing*, Jun. 2017.

[113] J. Cao, K. Li and I. Stojmenovic, "Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 45–58, Jan. 2014.

[114] J. Bi, H. Yuan, M. Tie and W. Tan, "SLA-Based Optimisation of Virtualised Resource for Multi-Tier Web Applications in Cloud Data Centres," *Enterprise Information Systems*, vol. 9, no. 7, pp. 743–767, Sept. 2015.

[115] J. Mei, K. Li, A. Ouyang and K. Li, "A Profit Maximization Scheme with Guaranteed Quality of Service in Cloud Computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3064–3078, Nov. 2015.

[116] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Workload-Aware Provisioning in Public Clouds," *IEEE Internet Computing*, vol. 18, no. 4, pp. 15–21, Jul. 2014.

[117] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang and J. Li, "Application-Aware Dynamic Fine-Grained Resource Provisioning in a Virtualized Cloud Data Center," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1172–1184, Apr. 2017.

[118] S. El Kafhali and K. Salah, "Stochastic Modelling and Analysis of Cloud Computing Data Center," in *Proc. of 20th Conference on Innovations in Clouds, Internet and Networks*, New York, NY, USA, 2017, pp. 153–167.

[119] A. Satpathy, S. K. Addya, A. K. Turuk, B. Majhi and G. Sahoo, "Crow Search Based Virtual Machine Placement Strategy in Cloud Data Centers with Live Migration," *Computers & Electrical Engineering*, vol. 69, pp. 334–350, Dec. 2018.

[120] A. Ponraj, "Optimistic Virtual Machine Placement in Cloud Data Centers Using Queuing Approach," *Future Generation Computer Systems*, vol. 93, pp. 338–344, Nov. 2019.

[121] G. Darzanos, I. Koutsopoulos and G. D. Stamoulis, "Cloud Federations: Economics, Games and Benefits," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2111–2124, Oct. 2019.

[122] L. Li, Q. Guan, L. Jin and M. Guo, "Resource Allocation and Task Offloading for Heterogeneous Real-Time Tasks With Uncertain Duration Time in a Fog Queueing System," *IEEE Access*, vol. 7, pp. 9912–9925, Jan. 2019.

[123] G. Fang, X. Li and Z. Cai, "Optimal Pricing for Service Provision in IaaS Cloud Markets," in *Proc. International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Xi'an, China, 2017, pp. 205-209.

[124] K. Santhi and R. Saravanan, "Performance Analysis of Cloud Computing Using Batch Queueing Models in Healthcare Systems," *Research Journal of Pharmacy and Technology*, vol. 10, no. 10, pp. pp. 3331–3336, Mar. 2018.

[125] Q. Fang, J. Wang and Q. Gong, "QoS-Driven Power Management of Data Centers via Model Predictive Control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1557–1566, Oct. 2016.

[126] L. Yu, T. Jiang and Y. Zou, "Distributed Real-Time Energy Management in Data Center Microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3748–3762, Jul. 2018.

[127] Q. Fang, J. Wang, Q. Gong and M. Song, "Thermal-Aware Energy Management of an HPC Data Center via Two-Time-Scale Control," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2260–2269, Oct. 2017.

[128] H. Rong, H. Zhang, S. Xiao, C. Li and C. Hu, "Optimizing Energy Consumption for Data Centers," *IEEE Transactions on Industrial Informatics*, vol. 58, pp. 674–691, May 2016.

[129] M. Vasudevan, Y. Tiana, M. Tang, E. Kozan and X. Zhang, "Energy-efficient Application Assignment in Profile-based Data Center Management through a Repairing Genetic Algorithm," *Applied Soft Computing*, vol. 67, pp. 399–408, Jun. 2018.

[130] X. Hu. P. Li. K. Wang. Y. Sun. D. Zeng. X. Wang and S. Guo, "Joint Workload Scheduling and Energy Management for Green Data Centers Powered by Fuel Cells," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 397–406, Jun. 2019.

[131] K. Kaur, S. Garg, G. Kaddoum, E. Bou-Harb and K. R. Choo, "A Big Data-Enabled Consolidated Framework for Energy Efficient Software Defined Data Centers in IoT Setups," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2687–2697, Apr. 2020.

[132] L. Shi, Y. Shi, X. Wei, X. Ding and Z. Wei, "Cost Minimization Algorithms for Data Center Management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 60–71, Jan. 2017.

[133] J. Yang, S. Zhang, X. Wu, Y. Ran and H. Xi, "Online Learning-Based Server Provisioning for Electricity Cost Reduction in Data Center," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1044–1051, May 2017.

[134] N. Hogade, S. Pasricha, H. J. Siegel, A. A. Maciejewski, M. A. Oxley and E. Jonardi, "Minimizing Energy Costs for Geographically Distributed Heterogeneous Data Centers," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 318–331, Oct. 2018.

[135] C. Canali, L. Chiaraviglio, R. Lancellotti and M. Shojafar, "Joint Minimization of the Energy Costs From Computing, Data Transmission, and Migrations in Cloud Data Centers," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 2, pp. 580–595, Jun. 2018.

[136] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy and Y. Tian, "Adaptive Energy-Aware Algorithms for Minimizing Energy Consumption and

SLA Violation in Cloud Computing," *IEEE Access*, vol. 6, pp. 55923–55936, Oct. 2018.

[137] X. Hu, P. Li, K. Wang, Y. Sun, D. Zeng, X. Wang and S. Guo, "Joint Workload Scheduling and Energy Management for Green Data Centers Powered by Fuel Cells," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 397–406, Jun. 2019.

[138] A. Al-Dulaimi, S. Al-Rubaye and Q. Ni, "Energy Efficiency Using Cloud Management of LTE Networks Employing Fronthaul and Virtualized Baseband Processing Pool," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 403–414, Jun. 2019.

[139] M. Erol-Kantarci and H. T. Mouftah, "Energy-Efficient Information and Communication Infrastructures in the Smart Grid: A Survey on Interactions and Open Issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 179–197, Jul. 2015.

[140] H. Chen, P. Lu, P. Xiong, C. Z. Xu and Z. Wang, "Energy-Aware Application Performance Management in Virtualized Data Centers," *Frontiers of Computer Science*, vol. 6, no. 4, pp. 373–387, Aug., 2012.

[141] T. Chen, Y. Zhang, X. Wang and G. B. Giannakis, "Robust Workload and Energy Management for Sustainable Data Centers," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 651–664, Mar. 2016.

[142] X. Lyu, H. Tian, C. Sengul and P. Zhang,"Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[143] D. Li, P. Hong, K. Xue and J. Pei, "Virtual Network Function Placement Considering Resource Optimization and SFC Requests in Cloud Datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[144] J. Li, M. Peng, Y. Yu and Z. Ding, "Energy-Efficient Joint Congestion Control and Resource Optimization in Heterogeneous Cloud Radio Access Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9873–9887, Dec. 2016.

[145] J. M. Luna, C. T. Abdallah and G. L. Heileman, "Probabilistic Optimization of Resource Distribution and Encryption for Data Storage in the Cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 428–439, Jun. 2018.

[146] S. Mireslami, L. Rakai, B. H. Far and M. Wang, "Simultaneous Cost and QoS Optimization for Cloud Resource Allocation," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 676–689, Sept. 2017.

[147] A. C. Zhou, B. He, X. Cheng and C. T. Lau, "A Declarative Optimization Engine for Resource Provisioning of Scientific Workflows in Geo-Distributed Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 647–661, Mar. 2017.

[148] L. Chou, H. Chen, F. Tseng, H. Chao and Y. Chang, "DPRA: Dynamic Power-Saving Resource Allocation for Cloud Data Center Using Particle Swarm Optimization," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1554–1565, Jun. 2018.

[149] S. G. Domanal, R. M. R. Guddeti and R. Buyya, "A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 3–15, Feb. 2020.

[150] A. S. Alrawahi, K. Lee and A. Lotfi, "A Multiobjective QoS Model for Trading Cloud of Things Resources," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9447–9463, Dec. 2019.

[151] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin and A. Sala, "Smoothed Online Resource Allocation in Multi-Tier Distributed Cloud Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2556–2570, Aug. 2017.

[152] Y. Li, J. Liu, B. Cao and C. Wang, "Joint Optimization of Radio and Virtual Machine Resources With Uncertain User Demands in Mobile Cloud Computing," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2427–2438, Sept. 2018.

[153] X. Qiu, Y. Dai, Y. Xiang and L. Xing, "Correlation Modeling and Resource Optimization for Cloud Service With Fault Recovery," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 693–704, Jul. 2019.

[154] M. Liu and Y. Liu, "Price-Based Distributed Offloading for Mobile-Edge Computing With Computation Capacity Constraints," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 420–423, Jun. 2018.

[155] Z. Wei, B. Zhao, J. Su and X. Lu, "Dynamic Edge Computation Offloading for Internet of Things With Energy Harvesting: A Learning Method," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4436–4447, Jun. 2019.

[156] W. Li, Y. Zhao, S. Lu and D. Chen, "Mechanisms and Challenges on Mobility-Augmented Service Provisioning for Mobile Cloud Computing," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 89–97, Mar. 2015.

[157] H. Guo and J. Liu, "Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber-Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[158] Z. Ning, P. Dong, X. Kong and F. Xia, "A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[159] S. Bi and Y. J. Zhang, "Computation Rate Maximization for Wireless Powered Mobile-Edge Computing With Binary Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[160] L. Lei, H. Xu, X. Xiong, K. Zheng and W. Xiang, "Joint Computation Offloading and Multiuser Scheduling Using Approximate Dynamic Programming in NB-IoT Edge Computing System," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5345–5362, Jun. 2019.

[161] S. Yu, R. Langar, X. Fu, L. Wang and Z. Han, "Computation Offloading With Data Caching Enhancement for Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.

[162] L. Chen, S. Zhou and J. Xu, "Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[163] T. Q. Dinh, Q. D. La, T. Q. S. Quek and H. Shin, "Learning for Computation Offloading in Mobile Edge Computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[164] Z. Hong, W. Chen, H. Huang, S. Guo and Z. Zheng, "Multi-Hop Cooperative Computation Offloading for Industrial IoT-Edge-Cloud Computing Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.

[165] J. Zheng, Y. Cai, Y. Wu and X. Shen, "Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[166] X. Tao, K. Ota, M. Dong, H. Qi and K. Li, "Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774–777, Dec. 2017.

[167] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li and A. Ylä-Jääski, "Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

[168] J. Ren, G. Yu, Y. He and G. Y. Li, "Collaborative Cloud and Edge Computing for Latency Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[169] S. Han, H. Su, C. Yang and A. F. Molisch, "Proactive Edge Caching for Video on Demand with Quality Adaptation," *IEEE Transactios on Wireless Communications*, vol. 19, no. 1, pp. 218–234, Oct. 2019.

[170] A. Mehrabi, M. Siekkinen and A. Ylä-Jääski, "Edge Computing Assisted Adaptive Mobile Video Streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 787–800, Apr. 2019.

[171] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang and S. Ou, "Dynamic Resource Scheduling in Mobile Edge Cloud with Cloud Radio Access Network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429–2445, Nov. 2018.

[172] S. Hu and G. Li, "Dynamic Request Scheduling Optimization in Mobile Edge Computing for IoT Applications," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1426-1437, Feb. 2020.

[173] J. Ren, G. Yu, Y. He and G. Y. Li, "Collaborative Cloud and Edge Computing for Latency Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[174] C. Wu, Y. Zhang and Y. Deng, "Toward Fast and Distributed Computation Migration System for Edge Computing in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10041–10052, Dec. 2019.

[175] T. Ouyang, Z. Zhou and X. Chen, "Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[176] X. Tao, K. Ota, M. Dong, H. Qi and K. Li, "Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774–777, Dec. 2017.

[177] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[178] D. Xu, Q. Li and H. Zhu, "Energy-Saving Computation Offloading by Joint Data Compression and Resource Allocation for Mobile-Edge Computing," *IEEE Communications Letters*, vol. 23, no. 4, pp. 704–707, Apr. 2019.

[179] M. Li, F. R. Yu, P. Si and Y. Zhang, "Energy-Efficient Machine-to-Machine (M2M) Communications in Virtualized Cellular Networks with Mobile Edge Computing (MEC)," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1541–1555, Jul. 2019.

[180] G. Zhang, Y. Chen, Z. Shen and L. Wang, "Distributed Energy Management for Multiuser Mobile-Edge Computing Systems With Energy Harvesting Devices and QoS Constraints," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4035–4048, Jun. 2019.

[181] G. Zhang, W. Zhang, Y. Cao, D. Li and L. Wang, "Energy-Delay Tradeoff for Dynamic Offloading in Mobile-Edge Computing System With Energy Harvesting Devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.

[182] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming and N. Lu, "Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing for Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4791–4803, Jun. 2019.

[183] L. Ji and S. Guo, "Energy-Efficient Cooperative Resource Allocation in Wireless Powered Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4744–4754, Jun. 2019.

[184] Y. Sun, S. Zhou and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[185] Z. Zhou, J. Feng, Z. Chang and X. Shen, "Energy-Efficient Edge Computing Service Provisioning for Vehicular Networks: A Consensus ADMM Approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[186] L. Pu, X. Chen, G. Mao, Q. Xie and J. Xu, "Chimera: An Energy-Efficient and Deadline-Aware Hybrid Edge Computing Framework for Vehicular Crowdsensing Applications," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 84–99, Feb. 2019.

[187] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou and A. Y. Zomaya, "Adaptive Energy-Aware Computation Offloading for Cloud of Things Systems," *IEEE Access*, vol. 5, pp. 23947–23957, Oct. 2017.

[188] Y. Dong, S. Guo, J. Liu and Y. Yang, "Energy-Efficient Fair Cooperation Fog Computing in Mobile Edge Networks for Smart City," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7543–7554, Oct. 2019.

[189] Z. Zhou, J. Feng, Z. Chang and X. Shen, "Energy-Efficient Edge Computing Service Provisioning for Vehicular Networks: A Consensus ADMM Approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[190] S. Garg, J. Aryal, H. Wang, T. Shah, G. Kecskemeti and R. Ranjan, "Cloud Computing Based Bushfire Prediction for Cyber–Physical Emergency

Applications," *Future Generation Computer Systems*, vol. 79, pp. 354–363, Mar. 2017.

[191] M.-F. Hsieh, F.-S. Hsu and D. G. Dorrell, "Winding Changeover Permanent-Magnet Generators for Renewable Energy Applications," *IEEE Transactions on Magnetics*, vol. 48, no. 11, pp. 4168–4171, Nov. 2012.

[192] M. Khodayar, O. Kaynak and M. E. Khodayar, "Rough Deep Neural Architecture for Short-Term Wind Speed Forecasting," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2770–2779, Dec. 2017.

[193] D. Wu, X. Luo, M. Shang, Y. He, G. Wang and M. Zhou, "A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, doi: 10.1109/TSMC.2019.2931393, online, Aug. 2019.

[194] S. Li, M. Zhou and X. Luo, "Modified Primal-Dual Neural Networks for Motion Control of Redundant Manipulators with Dynamic Rejection of Harmonic Noises," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4791–4801, Oct. 2018.

[195] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi and J. Wang, "Dendritic Neuron Model With Effective Learning Algorithms for Classification, Approximation, and Prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[196] Z. Ding, Y. Zhou, G. Pu and M. C. Zhou, "Online Failure Prediction for Railway Transportation Systems Based on Fuzzy Rules and Data Analysis", *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1143–1158, Sept. 2018.

[197] X. Li, K. Xing, M. C. Zhou, X. Wang and Y. Wu, "Modified Dynamic Programming Algorithm for Optimization of Total Energy Consumption in Flexible Manufacturing Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 691–705, Apr. 2019.

[198] M. Shokrian and K. A. High, "Application of a Multi Objective Multi-Leader Particle Swarm Optimization Algorithm on NLP and MINLP Problems," *Computers and Chemical Engineering*, vol. 60, no. 1, pp. 57–75, Jan. 2014.

[199] C. B. Khadse, M. A. Chaudhari and V. B. Borghate, "Electromagnetic Compatibility Estimator Using Scaled Conjugate Gradient Backpropagation Based Artificial Neural Network," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1036–1045, Jun. 2017.

[200] W. Sheng, K. y. Liu, S. Cheng, X. Meng and W. Dai, "A Trust Region SQP Method for Coordinated Voltage Control in Smart Distribution Grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 381–391, Jan. 2016.

[201] W. Tian, H. Zhou and W. Deng, "A Class of Second Order Difference Approximations for Solving Space Fractional Diffusion Equations," *Mathematics of Computation*, vol. 84, no. 294, pp. 1703–1727, Jan. 2015.

[202] A. H. Gandomi and X. Yang, "Chaotic Bat Algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp. 224–232, Mar. 2014.

[203] X. Zuo, S. Gao, M. Zhou, X. Yang and X. Zhao, "A Three-Stage Approach to a Multirow Parallel Machine Layout Problem," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 433–447, Jan. 2019.

[204] H. Hallawi, J. Mehnen and H. He, "Multi-Capacity Combinatorial Ordering GA in Application to Cloud Resources Allocation and Efficient Virtual Machines Consolidation," *Future Generation Computer Systems*, vol. 69, pp. 1–10, Apr. 2017.

[205] D. Ardagna, B. Panicucci and M. Passacantando, "Generalized Nash Equilibria for the Service Provisioning Problem in Cloud Systems," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 429–442, Oct. 2013.

[206] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A.A. Alelaiwi and F. Li, "Minimizing SLA Violation and Power Consumption in Cloud Data Centers Using Adaptive Energy-aware Algorithms," *Future Generation Computer Systems*, vol. 86, pp. 836–850, Sept. 2018.

[207] H. Yuan, J. Bi, W. Tan and B. H. Li, "Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[208] S. Diamond and S. Boyd, "CVXPY: A Python-embedded Modeling Language for Convex Optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, Apr. 2016.

[209] H. S. Kim and N. B. Shroff, "Loss Probability Calculations and Asymptotic Analysis for Finite Buffer Multiplexers," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 755–768, Dec. 2001.

[210] Y. L. Luke, "Mathematical Functions and Their Approximations", *Academic Press*, Cambridge, MA, USA, 2014.

[211] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang and D. Tao, "Stacked Convolutional Denoising Auto-Encoders for Feature Representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[212] W. Samek, A. Binder, G. Montavon, S. Lapuschkin and K. R. M´uller, "Evaluating the Visualization of What a Deep Neural Network Has Learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, Nov. 2017.

[213] A. N. Toosi, K. Vanmechelen, F. Khodadadi and R. Buyya, "An Auction Mechanism for Cloud Spot Markets," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 1, pp. 1–33, Apr. 2016.

[214] A. Panda and S. Pani, "A Symbiotic Organisms Search Algorithm with Adaptive Penalty Function to Solve Multi-Objective Constrained Optimization Problems," *Applied Soft Computing*, vol. 46, pp. 344–360, Sept. 2016.

[215] H. Zhang, J. Zhang, G. H. Yang and Y. Luo, "Leader-Based Optimal Coordination Control for the Consensus Problem of Multiagent Differential Games via Fuzzy Adaptive Dynamic Programming," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 1, pp. 152–163, Feb. 2015.

[216] B. OD́onoghue and E. Candès, "Adaptive Restart for Accelerated Gradient Schemes," *Foundations of Computational Mathematics*, vol. 15, no. 3, pp. 715–732, Jun. 2015.

[217] X. Guo, S. Liu, G. Tian and M. C. Zhou, "Disassembly Sequence Optimization for Large-scale Products with Multi-resource Constraints Using Scatter Search and Petri Nets," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2435–2446, Nov. 2016.

[218] M. R. Tanweer, S. Suresh and N. Sundararajan, "Self Regulating Particle Swarm Optimization Algorithm," *Information Sciences*, vol. 294, pp. 182–202, Feb. 2015.

[219] W. Dong and M. C. Zhou, "A Supervised Learning and Control Method to Improve Particle Swarm Optimization Algorithms," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, vol. 47, no. 7, pp. 1149–1159, Jul. 2017.

[220] S. Amaran, N. V. Sahinidis, B. Sharda and S. J. Bury, "Simulation Optimization: A Review of Algorithms and Applications," *Annals of Operations Research*, vol. 240, no. 1, pp. 351–380, May. 2016.

[221] F. Ahmadizar, K.Soltanian, F. AkhlaghianTab and I. Tsoulos, "Artificial Neural Network Development by Means of a Novel Combination of Grammatical Evolution and Genetic Algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, Mar. 2015.

[222] J. Luo, L. Rao and X. Liu, "Spatio-Temporal Load Balancing for Energy Cost Optimization in Distributed Internet Data Centers," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 387–397, Jul. 2015.

[223] J. Diaz-Montes, M. Diaz-Granados, M. Zou, S. Tao and M. Parashar, "Supporting Data-Intensive Workflows in Software-Defined Federated Multi-Clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 250–263, Jan. 2018.

[224] F. Yao, J. Wu, S. Subramaniam and G. Venkataramani, "WASP: Workload Adaptive Energy-Latency Optimization in Server Farms Using Server Low-Power States," in *Proceedings of IEEE 10th International Conference on Cloud Computing*, 2017, pp. 75–84.

[225] A. Heidari, Z. Dong, D. Zhang, P. Siano and J. Aghaei, "Mixed-Integer Nonlinear Programming Formulation for Distribution Networks Reliability Optimization," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, Nov. 2017.

[226] Z. Zhu, J. Bi, H. Yuan and Y. Chen, "SLA Based Dynamic Virtualized Resources Provisioning for Shared Cloud Data Centers," in *Proceedings of IEEE International Conference on Cloud Computing*, 2011, pp. 630–637.

[227] G. Tian, Y. Ren and M. C. Zhou, "Dual-Objective Scheduling of Rescue Vehicles to Distinguish Forest Fires via Differential Evolution and Particle Swarm Optimization Combined Algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3009–3021, Nov. 2016.

[228] Z. Zhao, M. Peng, Z. Ding, W. Wang and H. V. Poor, "Cluster Content Caching: An Energy-Efficient Approach to Improve Quality of Service in Cloud Radio Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1207–1221, May 2016.

[229] K. Thirugnanam, S. K. Kerk, C. Yuen, N. Liu and M. Zhang, "Energy Management for Renewable Microgrid in Reducing Diesel Generators Usage With Multiple Types of Battery," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6772–6786, Aug. 2018.

[230] A. Rubio-Solis, P. Melin, U. Martinez-Hernandez and G. Panoutsos, "General Type-2 Radial Basis Function Neural Network: A Data-Driven Fuzzy Model," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 2, pp. 333–347, Feb. 2019.

[231] T. Trzciński and P. Rokita, "Predicting Popularity of Online Videos Using Support Vector Regression," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2561–2570, Nov. 2017.

[232] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura and R. Bianchini, "Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms," in *Proc. of 26th Symposium on Operating Systems Principles*, New York, NY, USA, 2017, pp. 153–167.

[233] X. Zuo, B. Li, X. Huang, M. Zhou, C. Cheng, X. Zhao and Z. Liu, "Optimizing Hospital Emergency Department Layout via Multiobjective Tabu Search," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1137–1147, Jul. 2019.

[234] Q. Peng, M. Zhou, Q. He, Y. Xia, C. Wu and S. Deng, "Multi-Objective Optimization for Location Prediction of Mobile Devices in Sensor-based Applications," *IEEE Access*, vol. 6, pp. 77123–77132, Dec. 2018.

[235] S. Gao, J. Cheng, Y. Todo and M. Zhou, "Incorporation of Solvent Effect into Multi-Objective Evolutionary Algorithm for Improved Protein Structure Prediction", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 4, pp. 1365–1378, Jul. 2018.

[236] N. Taran, D. M. Ionel and D. G. Dorrell, "Two-Level Surrogate-Assisted Differential Evolution Multi-Objective Optimization of Electric Machines Using 3-D FEA," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, Nov. 2018.

[237] H. Ishibuchi, H. Masuda and Y. Nojima, "Pareto Fronts of Many-Objective Degenerate Test Problems," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 807–813, Oct. 2016.

[238] W. Chiu, G. G. Yen and T. Juan, "Minimum Manhattan Distance Approach to Multiple Criteria Decision Making in Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 972–985, Dec. 2016.

[239] U. Singh and S. N. Singh, "Optimal Feature Selection via NSGA-II for Power Quality Disturbances Classification," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 2994–3002, Jul. 2018.

[240] A. Wahid, X. Gao and P. Andreae, "Multi-Objective Clustering Ensemble for High-Dimensional Data Based on Strength Pareto Evolutionary Algorithm (SPEA-II)," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–9.

[241] A. K. Kiani and N. Ansari, "On The Fundamental Energy Trade-offs of Geographical Load Balancing," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 170–175, May 2017.

[242] K. Wang, H. Li, S. Maharjan, Y. Zhang and S. Guo, "Green Energy Scheduling for Demand Side Management in the Smart Grid," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 2, pp. 596–611, Jun. 2018.

[243] P. Agrawal and S. Rao, "Energy-Aware Scheduling of Distributed Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1163–1175, Oct. 2014.

[244] B. Zhai, D. Blaauw, D. Sylvester and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," in *Proc. 41st Design Automation Conference*, San Diego, CA, USA, 2004, pp. 868–873.

[245] J. Cao, K. Hwang, K. Li and A. Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[246] H. Yuan, J. Bi and M. Zhou, "Spatiotemporal Task Scheduling for Heterogeneous Delay-Tolerant Applications in Distributed Green Data Centers," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1686–1697, Oct. 2019.

[247] S. Jiang and S. Yang, "A Strength Pareto Evolutionary Algorithm Based on Reference Direction for Multiobjective and Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, Jun. 2017.

[248] Z. Fan, H. Li, C. Wei, W. Li, H. Huang, X. Cai and Z. Cai, "An Improved Epsilon Constraint Handling Method Embedded in MOEA/D for Constrained Multi-objective Optimization Problems," in *Proc. of IEEE Symposium Series on Computational Intelligence*, Athens, Greece, 2016, pp. 1–8.

[249] Y. Zhang, D. Gong and J. Cheng, "Multi-Objective Particle Swarm Optimization Approach for Cost-Based Feature Selection in Classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 64–75, Feb. 2017.

[250] A. Kamjoo, A. Maheri, A. M. Dizqah and G. A. Putrus, "Multi-objective Design under Uncertainties of Hybrid Renewable Energy System using NSGA-II and Chance Constrained Programming," *International Journal of Electrical Power & Energy Systems*, vol. 74, pp. 187–194, Jan. 2016.

[251] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou and J. Wang, "Bi-objective Elite Differential Evolution Algorithm for Multivalued Logic Networks", *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 233–246, Jan. 2020.

[252] Y. Yu, S. C. Gao, Y. R. Wang and Y. Todo, "Global Optimum-based Search Differential Evolution," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 379–394, Mar. 2019.

[253] S. Das, S. S. Mullick and P. N. Suganthan, "Recent Advances in Differential Evolution–An Updated Survey," *Swarm and Evolutionary Computation*, vol. 27, pp.1–30, Apr. 2016.

[254] U. Mlakar, I. Fister, J. Brest and B. Potočnik, "Multi-objective Differential Evolution for Feature Selection in Facial Expression Recognition Systems," *Expert Systems with Applications*, vol. 15, no. 89, pp. 129–137, Dec. 2017.

[255] C. Lin, L. Shu and D. Deng, "Router Node Placement With Service Priority in Wireless Mesh Networks Using Simulated Annealing With Momentum Terms," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1402–1411, Dec. 2016.

[256] M. Zhang and H. Li. "A Reference Direction and Entropy based Evolutionary Algorithm for Many-objective Optimization," *Applied Soft Computing*, vol. 70, pp. 108–130, Sept. 2018.

[257] Y. Shen and Y. Wang, "Operating Point Optimization of Auxiliary Power Unit Using Adaptive Multi-Objective Differential Evolution Algorithm," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 115–124, Jan. 2017.

[258] Q. Kang, X. Song, M. Zhou and L. Li, "A Collaborative Resource Allocation Strategy for Decomposition-Based Multiobjective Evolutionary Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2416–2423, Dec. 2019.

[259] A. Ghassami and N. Kiyavash, "A Covert Queueing Channel in FCFS Schedulers," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1551–1563, Jun. 2018.

[260] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag and B. Maggs, "Cutting the Electric Bill for Internet-scale Systems," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 123–134, Oct. 2009.

[261] X. Fan, W. D. Weber and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, May 2007.

[262] P. Torcellini, S. Pless, M. Deru and D. Crawley, "Zero Energy Buildings: A Critical Look at the Definition," in *Proc. of the ACEEE Summer Study on Energy Efficiency in Buildings*, August Pacific Grove, CA, USA, 2006, pp. 1–12.

[263] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan and X. Chen, "Deep Transfer Learning Based on Sparse Autoencoder for Remaining Useful Life Prediction of Tool in Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2416–2425, Apr. 2019.

[264] W. Zhang, Y. Wen, Y. Wah Wong, K. Chuan Toh and C. Chen, "Towards Joint Optimization Over ICT and Cooling Systems in Data Centre: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1596–1616, Mar. 2016.

[265] Q. Kang, S. W. Feng, M. C. Zhou, A. C. Ammari and K. Sedraoui, "Optimal Load Scheduling of Plug-in Hybrid Electric Vehicles via Weight-Aggregation Multi-objective Evolutionary Algorithms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2557–2568, Sept. 2017.

[266] Y. Feng, M. C. Zhou, G. Tian, Z. Li, Z. Zhang, Q. Zhang and J. Tan, "Target Disassembly Sequencing and Scheme Evaluation for CNC Machine Tools Using Improved Multiobjective Ant Colony Algorithm and Fuzzy Integral," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2438–2451, Dec. 2019.

[267] Q. Wu, M. Zhou, Q. Zhu, Y. Xia and J. Wen, "MOELS: Multiobjective Evolutionary List Scheduling for Cloud Workflows," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 166–176, Jan. 2020.

[268] Y. Hou, N. Wu, M. C. Zhou and Z. Li, "Pareto-optimization for Scheduling of Crude Oil Operations in Refinery via Genetic Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[269] X. Ma, Q. Zhang, G. Tian, J. Yang and Z. Zhu, "On Tchebycheff Decomposition Approaches for Multiobjective Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 226–244, Apr. 2018.

[270] J. Luo, A. Gupta, Y. Ong and Z. Wang, "Evolutionary Optimization of Expensive Multiobjective Problems With Co-Sub-Pareto Front Gaussian Process Surrogates," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1708–1721, May 2019.

[271] F. Tseng, X. Wang, L. Chou, H. Chao and V. C. M. Leung, "Dynamic Resource Prediction and Allocation for Cloud Data Center Using the Multiobjective Genetic Algorithm," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1688–1699, Jun. 2018.

[272] J. Zhang and L. Xing, "A Survey of Multiobjective Evolutionary Algorithms," in *Proc. IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Guangzhou, China, 2017, pp. 93–100.

[273] Z. He, G. G. Yen and J. Zhang, "Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 269–285, Apr. 2014.

[274] B. Zhang, A. Wang and M. Doppelbauer, "Multi-Objective Optimization of a Transverse Flux Machine With Claw-Pole and Flux-Concentrating Structure," *IEEE Transactions on Magnetics*, vol. 52, no. 8, pp. 1–10, Aug. 2016.

[275] W. Dou, X. Xu, S. Meng, X. Zhang, C. Hu, S. Yu and J. Yang, "An Energy–aware Virtual Machine Scheduling Method for Service QoS Enhancement in Clouds over Big Data," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 14, 1–20, Jul. 2017.

[276] M. N. Hjelmeland, J. Zou, A. Helseth and S. Ahmed, "Nonconvex Medium-Term Hydropower Scheduling by Stochastic Dual Dynamic Integer Programming," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 1, pp. 481–490, Jan. 2019.

[277] F. Zhang and D. R. Bull,"Rate-Distortion Optimization Using Adaptive Lagrange Multipliers," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 10, pp. 3121–3131, Oct. 2019.

[278] A. A. Augusto, M. B. Do Coutto Filho, J. C. S. de Souza and M. A. R. Guimaraens, "Branch-and-Bound Guided Search for Critical Elements in State Estimation," *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 2292–2301, May 2019.

[279] F. Bistaffa, N. Bombieri and A. Farinelli, "An Efficient Approach for Accelerating Bucket Elimination on GPUs," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3967–3979, Nov. 2017.

[280] S. Pare, A. Kumar, V. Bajaj and G. K. Singh, "A Context Sensitive Multilevel Thresholding Using Swarm Based Algorithms," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1471–1486, Nov. 2019.

[281] E. Yigit and H. Duysak, "Determination of Optimal Layer Sequence and Thickness for Broadband Multilayer Absorber Design Using Double-Stage Artificial Bee Colony Algorithm," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 8, pp. 3306–3317, Aug. 2019.

[282] Q. Duan, J. Zeng, K. Chakrabarty and G. Dispoto, "Real-Time Production Scheduler for Digital-Print-Service Providers Based on a Dynamic Incremental Evolutionary Algorithm," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 701–715, Apr. 2015.

[283] L. Zhu, F. R. Yu, Y. Wang, B. Ning and T. Tang, "Big Data Analytics in Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, Jan. 2019.

[284] B. Wang, H. Xie, X. Xia and X. Zhang, "A NSGA-II Algorithm Hybridizing Local Simulated-Annealing Operators for a Bi-Criteria Robust Job-Shop Scheduling Problem Under Scenarios," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 5, pp. 1075–1084, May 2019.

[285] M. Castellani, S. Otri and D.T. Pham, "Printed Circuit Board Assembly Time Minimisation Using a Novel Bees Algorithm," *Computers & Industrial Engineering*, vol. 133, pp.186–194, May 2019.

[286] Z. J. Huang, Z. S. Wang and H. G. Zhang, "Multilevel Feature Moving Average Ratio Method for Fault Diagnosis of the Microgrid Inverter Switch," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 177–185, Apr. 2017.

[287] P. Zhang, S. Shu and M. C. Zhou, "An Online Fault Detection Method based on SVM-Grid for Cloud Computing Systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.

[288] A. Kavousi, B. Vahidi, R. Salehi, M. K. Bakhshizadeh, N. Farokhnia and S. H. Fathi, "Application of the Bee Algorithm for Selective Harmonic Elimination Strategy in Multilevel Inverters," *IEEE Transactions on Power Electronics*, vol. 27, no. 4, pp. 1689–1696, Apr. 2012.

[289] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[290] H. Yuan, J. Bi and M. Zhou, "Spatial Task Scheduling for Cost Minimization in Distributed Green Cloud Data Centers," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 729–740, Apr. 2019.

[291] S. Shenoy, D. Gorinevsky and N. Laptev, "Probabilistic Modeling of Computing Demand for Service Level Agreement," *IEEE Transactions on Services Computing*, vol. 12, no. 6, pp. 987–993, Nov. 2019.

[292] A. Barri and A. Dooms, "Data-Driven Modules for Objective Visual Quality Assessment Focusing on Benchmarking and SLAs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 196–205, Feb. 2017.

[293] S. Mubeen, S. A. Asadollah, A. V. Papadopoulos, M. Ashjaei, H. Pei-Breivold and M. Behnam, "Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study," *IEEE Access*, vol. 6, pp. 30184–30207, Aug. 2018.

[294] O. Muñoz, A. Pascual-Iserte and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[295] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[296] J. Conejero, O. Rana, P. Burnap, J. Morgan, B. Caminero and C. Carrión, "Analyzing Hadoop Power Consumption and Impact on Application QoS," *Future Generation Computer Systems*, vol. 55, pp. 213–223, Feb. 2016.

[297] A. Jyoti, M. Shrimali and R. Mishra, "Cloud Computing and Load Balancing in Cloud Computing -Survey," in *Proc. of 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2019, pp. 51–55.

[298] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen and L. Xiao, "Learning Driven Computation Offloading for Asymmetrically Informed Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1802–1815, Aug. 2019.

[299] Y. Du, C. Xu and D. Tao, “Matrix Factorization for Collaborative Budget Allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1471–1482, Oct. 2018.

[300] E. Duman, M. Uysal and A. F. Alkaya, “Migrating Birds Optimization: A New Metaheuristic Approach and Its Performance on Quadratic Assignment Problem," *Information Sciences*, vol. 217, pp. 65–77, Dec. 2012.

[301] F. Javidrad and M. Nazari, “A New Hybrid Particle Swarm and Simulated Annealing Stochastic Optimization Method," *Applied Soft Computing*, vol. 60, pp. 634–654, Nov. 2017.

[302] K. Jagatheesan, B. Anand, S. Samanta, N. Dey, A. S. Ashour and V. E. Balas, “Design of a Proportional-integral-derivative Controller for an Automatic Generation Control of Multi-area Power Thermal Systems Using Firefly Algorithm," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 503–515, Mar. 2019.

[303] F. J. Pontes, G. F. Amorim, P. P. Balestrassi, A. P. Paiva and J. R. Ferreira, “Design of Experiments and Focused Grid Search for Neural Network Parameter Optimization," *Neurocomputing*, vol. 186, pp. 22–34, Apr. 2016.

[304] X. S. Lu, M. Zhou, L. Qi and H. Liu, “Clustering Algorithm-based Analysis of Rare Event Evolution via Social Media Data," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 301–310, Apr. 2019.

[305] M. Ghahramani, M.C. Zhou and C.T. Hon, “Extracting Significant Mobile Phone Interaction Patterns based on Community Structures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1031–1041, Mar. 2019.

[306] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo and H. Xie, “Multi-Objective Workflow Scheduling With Deep-Q-Network-Based Multi-Agent Reinforcement Learning", *IEEE Access*, vol. 7, pp. 39974–39982, Mar. 2019.

[307] J. Bi, Z. Zhu, R. Tian and Q. Wang, “Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center," in *Proc. of IEEE 3rd International Conference on Cloud Computing*, Miami, FL, 2010, pp. 370–377.

[308] X. Qiu, W. L. Yeow, C. Wu and F. C. M. Lau, “Cost-minimizing Preemptive Scheduling of MapReduce Workloads on Hybrid Clouds," in *Proc. of IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*, Montreal, QC, 2013, pp. 1–6.

[309] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin and S. Kounev, “Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 800–813, Apr. 2019.

[310] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, S. Zhu and H. Leung, "Incorporation of Efficient Second-Order Solvers Into Latent Factor Models for Accurate Prediction of Missing QoS Data," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1216–1228, Apr. 2018.

[311] X. Luo, M. C. Zhou, Y. Xia, Q. Zhu, A. C. Ammari and A. Alabdulwahab, "Generating Highly Accurate Predictions for Missing QoS Data via Aggregating Nonnegative Latent Factor Models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 524–537, Mar. 2016.

[312] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin and X. S. Shen, "Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2019.2903240, Mar. 2019.

[313] M.S. Yoon, A.E. Kamal and Z. Zhu, "Adaptive Data Center Activation with User Request Prediction," *Computer Networks*, vol. 122, pp.191–204, Jul. 2017.

[314] M. Hähnel, J. Martinovic, G. Scheithauer, A. Fischer, A. Schill and W. Dargie, "Extending the Cutting Stock Problem for Consolidating Services with Stochastic Workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2478–2488, Nov. 2018.

[315] X. Zhu, J. Huang, L. Quan, Z. Xiang and B. Shi, "Comprehensive Sensitivity Analysis and Multiobjective Optimization Research of Permanent Magnet Flux-Intensifying Motors," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 2613–2627, Apr. 2019.

[316] B. Liu, X. Chang, Z. Han, K. Trivedi and R. J. Rodríguez, "Model-based Sensitivity Analysis of IaaS Cloud Availability," *Future Generation Computer Systems*, vol. 83, pp. 1–13, Jun. 2018.

[317] R. Matos, J. Dantas, J. Araujo, K. S. Trivedi and P. Maciel, "Redundant Eucalyptus Private Clouds: Availability Modeling and Sensitivity Analysis," *Journal of Grid Computing*, vol. 15, pp. 1–22, Nov. 2016.

[318] W. Tan and M. Zhou, "Service Oriented Workflow Systems," *Contemporary Issues in Systems Science and Engineering*, Edited by M. C. Zhou, H.-X. Li and M. Weijnen, Wiley/IEEE Press, Hoboken, NJ, pp. 645–660, 2015.

[319] W. Tan and M. C. Zhou, "Business and Scientific Workflows: A Service-Oriented Approach", *IEEE Press/Wiley*, Hoboken, NJ, 2013.

[320] W. Li, Y. Xia, M. Zhou, X. Sun and Q. Zhu, "Fluctuation-Aware and Predictive Workflow Scheduling in Cost-Effective Infrastructure-as-a-Service Clouds," *IEEE Access*, vol. 6, pp. 61488–61502, Sept. 2018.

[321] *INDICS Platform*. China Aerospace Science and Industry Corporation (CASIC), Beijing, China. Available: `http://intl.indics.com/` (accessed on February 1, 2019).

[322] K. Janod, M. Morchid, R. Dufour, G. Linarés and R. De Mori, "Denoised Bottleneck Features From Deep Autoencoders for Telephone Conversation Analysis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, pp. 1809–1820, Sept. 2017.

[323] E. Principi, D. Rossetti, S. Squartini and F. Piazza, "Unsupervised Electric Motor Fault Detection by Using Deep Autoencoders," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 441–451, Mar. 2019.

[324] W. Chien, C. Lai and H. Chao, "Dynamic Resource Prediction and Allocation in C-RAN With Edge Artificial Intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4306–4314, Jul. 2019.

[325] T. Hussain, K. Muhammad, A. Ullah, Z. Cao, S. W. Baik and V. H. C. de Albuquerque, "Cloud-Assisted Multiview Video Summarization Using CNN and Bidirectional LSTM," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 77–86, Jan. 2020.

[326] Q. Wu, M. Zhou, Q. Zhu and Y. Xia, "A VCG Auction-based Approach for Multi-granularity Service Composition," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 796–805, Apr. 2018.

[327] C. Pan, M. C. Zhou, Y. Qiao and N. Q. Wu, "Scheduling Cluster Tools in Semiconductor Manufacturing: Recent Advances and Challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 896–899, Apr. 2018.

[328] X. Q. Shang, F.-Y. Wang, G. Xiong, T. R. Nyberg, Y. Yuan, S. Liu, C. Guo and S. Bao, "Social Manufacturing for High-end Apparel Customization," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 489–500, Mar. 2018.

[329] G. Xiong, F-Y. Wang, T. R. Nyberg, X. Q. Shang, M. C. Zhou, Z. Shen, S. S. Li and C. Guo, "From Mind to Products: Towards Social Manufacturing and Service," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 47–57, Jan. 2018.

[330] L. Li, Y. Lin, N. Zheng and F.-Y. Wang, "Parallel Learning: A Perspective and a Framework," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 389–395, Jul. 2017.

[331] T. Liu, B. Tian, Y. F. Ai, L. Li, D. P. Cao and F.-Y. Wang, "Parallel Reinforcement Learning: A Framework and Case Study," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 4, pp. 827–835, Jul. 2018.

[332] S. Li, M. C. Zhou, X. Luo and Z.-H. You, "Distributed Winner-take-all in Dynamic Networks", *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 577–589, Feb. 2017.

[333] K. Z. Gao, Z. G. Cao, L. Zhang, Z. H. Chen, Y. Y. Han and Q. K. Pan, "A Review on Swarm Intelligence and Evolutionary Algorithms for Solving Flexible Job Shop Scheduling Problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 875–887, Jul. 2019.

[334] W. Gu, Y. Yu and W. Hu, "Artificial Bee Colony Algorithm-based Parameter Estimation of Fractional-order Chaotic System with Time Delay," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 107–113, Jan. 2017.

[335] Z. Han, J. Zhao and W. Wang, "An Optimized Oxygen System Scheduling with Electricity Cost Consideration in Steel Industry," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 216–222, Apr. 2017.

[336] Z. Lv, L. Wang, Z. Han, J. Zhao and W. Wang, "Surrogate-assisted Particle Swarm Optimization Algorithm with Pareto Active Learning for Expensive Multi-objective Optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, May 2019.

[337] Y. Tang, C. Luo, J. Yang and H. He, "A Chance Constrained Optimal Reserve Scheduling Approach for Economic Dispatch Considering Wind Penetration," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 186–194, Apr. 2017.

[338] B. Wang, X. Xia, H. Meng and T. Li, "Bad-scenario-set Robust Optimization Framework with Two Objectives for Uncertain Scheduling Systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 143–153, Jan. 2017.

[339] F. Wang, T. Xu, T. Tang, M. C. Zhou and H. Wang, "Bilevel Feature Extraction-based Text Mining for Fault Diagnosis of Railway Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 49–58, Jan. 2017.

[340] L. Wang and J. W. Lu, "A Memetic Algorithm with Competition for the Capacitated Green Vehicle Routing Problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 516–526, Mar. 2019.

[341] L. Wang, S. Wang and X. Zheng, "A Hybrid Estimation of Distribution Algorithm for Unrelated Parallel Machine Scheduling with Sequence-Dependent Setup Times," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 235–246, Jul. 2016.

[342] J. Li, J. Zhang, C. Jiang and M. Zhou, "Composite Particle Swarm Optimizer with Historical Memory for Function Optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2350–2363, Oct. 2015.

[343] J. Zhao, S. X. Liu, M. Zhou, X. W. Guo and L. Qi, "Modified Cuckoo Search Algorithm to Solve Economic Power Dispatch Optimization Problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 4, pp. 794–806, Jul. 2018.

[344] X. Meng, J. Li, M. Zhou, X. Dai and J. Dou, "Population-Based Incremental Learning Algorithm for a Serial Colored Traveling Salesman Problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 2, pp. 277–288, Feb. 2018.

[345] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang and W. A. Chaovalitwongse, "Comprehensive Learning Particle Swarm Optimization Algorithm with Local Search for Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 718–731, Aug. 2019.

[346] H. Zhu, G. Liu, M. Zhou, Y. Xie and Q. Kang, "Dandelion Algorithm with Probability-based Mutation," *IEEE Access*, vol. 7, pp. 97974–97985, Jul. 2019.

[347] W. Dong, Y. Wang and M. Zhou, "A Latent Space-Based Estimation of Distribution Algorithm for Large-scale Global Optimization," *Soft Computing*, vol. 23, no. 13, pp. 4593–4615, Jul. 2019.

[348] K. Gao, F. Yang, M. Zhou, Q. Pan and P. N. Suganthan, "Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1944–1955, May 2019.

[349] L. Huang, M. Zhou and K. Hao, "Non-dominated Immune-endocrine Short Feedback Algorithm for Multi-robot Maritime Patrolling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 362–373, Jan. 2020.

[350] M. Ghahramani, Y. Qiao, M. Zhou, A. O Hagan and J. Sweeney, "AI-based Modeling and Data-driven Evaluation for Smart Manufacturing Processes," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 948–959, Jul. 2020.

[351] S. Yang, M. Li, X. Liu and J. Zheng, "A Grid-Based Evolutionary Algorithm for Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, Oct. 2013.

[352] Y. Sun, B. Xue, M. Zhang and G. G. Yen, "A Particle Swarm Optimization-Based Flexible Convolutional Autoencoder for Image Classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2295–2309, Aug. 2019.