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

GLOBAL OPTIMIZATION ALGORITHMS FOR IMAGE REGISTRATION AND CLUSTERING

by
Cuicui Zheng

Global optimization is a classical problem of finding the minimum or maximum value of an objective function. It has applications in many areas, such as biological image analysis, chemistry, mechanical engineering, financial analysis, deep learning and image processing. For practical applications, it is important to understand the efficiency of global optimization algorithms. This dissertation develops and analyzes some new global optimization algorithms and applies them to practical problems, mainly for image registration and data clustering.

First, the dissertation presents a new global optimization algorithm which approximates the optimum using only function values. The basic idea is to use the points at which the function has been evaluated to decompose the domain into a collection of hyper-rectangles. At each step of the algorithm, it chooses a hyper-rectangle according to a certain criterion and the next function evaluation is at the center of the hyper-rectangle. The dissertation includes a proof that the algorithm converges to the global optimum as the number of function evaluations goes to infinity, and also establishes the convergence rate. Standard test functions are used to experimentally evaluate the algorithm.

The second part focuses on applying algorithms from the first part to solve some practical problems. Image processing tasks often require optimizing over some set of parameters. In the image registration problem, one attempts to determine the best transformation for aligning similar images. Such problems typically require minimizing a dissimilarity measure with multiple local minima. The dissertation

describes a global optimization algorithm and applies it to the problem of identifying the best transformation for aligning two images.

Global optimization algorithms can also be applied to the data clustering problem. The basic purpose of clustering is to categorize data into different groups by their similarity. The objective cost functions for clustering usually are non-convex. k-means is a popular algorithm which can find local optima quickly but may not obtain global optima. The different starting points for k-means can output different local optima. This dissertation describes a global optimization algorithm for approximating the global minimum of the clustering problem.

The third part of the dissertation presents variations of the proposed algorithm that work with different assumptions on the available information, including a version that uses derivatives.

GLOBAL OPTIMIZATION ALGORITHMS FOR IMAGE REGISTRATION AND CLUSTERING

by
Cuicui Zheng

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

Department of Computer Science

August 2020

APPROVAL PAGE

GLOBAL OPTIMIZATION ALGORITHMS FOR IMAGE REGISTRATION AND CLUSTERING

Cuicui Zheng

Dr. James Calvin, Dissertation Advisor — Date
Professor of Computer Science, NJIT

Dr. Craig Gotsman, Committee Member — Date
Distinguished Professor of Computer Science, NJIT

Dr. Marvin Nakayama, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Zhi Wei, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Mengchu Zhou, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Cuicui Zheng

Degree: Doctor of Philosophy

Date: August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, New Jersey, 2020
- Bachelor of Science in Information Security,
  Northeastern University, Shenyang, China, 2015

Major: Computer Science

Presentations and Publications:

Cuicui Zheng, James Calvin, Craig Gotsman "A DIRECT-type global optimization algorithm for image registration" *Journal of Global Optimization*, 2020.

Cuicui Zheng, James Calvin, "Using simulation to approximate the minimum cost of a finite set of alternatives" *Winter Simulation Conference*, Maryland, 12/2019.

James Calvin, Craig Gotsman, Cuicui Zheng, "Global Optimization for Image Registration" *Global Optimization Workshop*, Netherlands, 09/21/2018.

It does not matter how slowly you go as long as you do not stop.

Confucius

ACKNOWLEDGMENT

When I start the journey of PhD, I didn't know what the choice of starting PhD would mean. Lots of words in my heart come out in the end of this journey. With hundreds and thousands of days struggle and discussion, the dissertation was finally completed. I cannot forget the help from those who encouraged me, supported me and exchanged ideas with me.

I would like to express my sincere gratitude to my advisor, Prof. James Calvin for the continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me through out the time of research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my PhD study. Every time when I lost myself in self-doubt and questioned a lot about the research, discussion with Prof. James Calvin help me out. Thanks to the words of my advisor, it's been like lights in my PhD life.

I would also like to thank the rest of my committee: Prof. Craig Gotsman, Prof. Zhi Wei, Prof. Marvin Nakayama, Prof. Mengchu Zhou. For their encouragement, insightful comments, and hard questions. I would like to thank the National Science Foundation for financial support under Grant No. CMMI-1562466.

I thank my fellow labmates and friends in NJIT: Yanan Yang, Yajuan Li, Tian Tian, Jie Zhang, Xin Gao, Ling Zheng, Junyi Ye, Cheng Zhong for their help with sharing ideas and information with me. I also thank Prof. Ali Mili, Prof. Reza Curtmola, and Prof. Cristian Borcea for all suggestions in my first year of PhD life.

Last, but not the least, I would like to thank my family: My parents, Meiqiang Zheng and Shuizhen Li, for giving birth to me at the first place and supporting me spiritually throughout my life and my brothers, Chao Zheng and Yuchao Zheng who always bring happiness to my life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Global optimization is the task of finding minima or maxima of an objective function. Many different engineering areas need optimization algorithms. Heuristic methods, for example genetic algorithms, have been applied in machine learning [51, 10]. Local optimization methods, such as gradient descent, are used in many cases because of convenience and speed. Such local optimization methods alone cannot be relied on when there are many local optima.

Neural network objective functions can be non-linear, non-convex and also not smooth [39]. Variations of gradient descent algorithms are used to update parameters that minimize the value of a loss function in neural networks. Gradient descent works well in neural networks and finds global minima of deep neural networks [15] under some conditions. It is confusing why local search works well in a non-convex objective function. Convex functions can be minimized with local optimization algorithms. We are interested in global optimization algorithms for non-convex objective functions. We develop global optimization algorithms which are easy to use and also efficient to find the best points which are at or near the global optima points and also analyze their convergence rate. They approximate the optimal value without requiring inputs, such as a good "starting point" to avoid getting trapped in a local optimum.

The theoretical foundations of commonly used global optimization algorithms such as simulated annealing or genetic algorithms are weak. These global optimization algorithms are applied to many different areas for solving practical problems but it is difficult to characterize the performance of those algorithms.

How do we measure the performance of a global optimization algorithm? Let us suppose that the unknown objective function f is a member of some class of objective

functions F. If the class F is sufficiently restricted, then local optimization methods have a fast convergence rate. For example, if a univariate function f has a unique local minimum, then there exist optimization algorithms that bracket the minimizer inside a subinterval of width $O(\exp(-cn))$ after n observations for some positive number c (for example, the golden section search method [36]).

In many applications it is not reasonable to assume that f has a unique local minimum, but it may be reasonable to assume continuity or some degree of smoothness. If we assume only continuity, for example, then the class F is convex and symmetric; that is, if $f_1, f_2 \in F$ and $0 \leq \lambda \leq 1$, then $-f_1 \in F$ and $\lambda f_1 + (1-\lambda) f_2 \in F$. If the class F is convex and symmetric, then adaptive methods are not essentially better than nonadaptive methods in the worst-case; there exists a nonadaptive method that achieves the same worst-case error bound with at most $n+1$ function evaluations as an adaptive method that uses n function evaluations [79, 83].

Our interest is in problems with objective functions from a convex and symmetric class, and so we do not consider the worst-case model. Two alternatives to the worst-case complexity criterion are average-case and asymptotic-case. The average-case complexity of an algorithm is based on the assumption of a probability measure on the set of possible inputs. In this dissertation, we adopt the asymptotic setting. As the number of function evaluations n becomes large, we examine asymptotic bounds on the approximation error. We are interested in the following question: As the number of function evaluations increases, at what asymptotic rate does the error converge to zero?

Convergence rates are not known for many global optimization algorithms [20]. The DIRECT algorithm [19] applies when the objective function is Lipschtz continuous. To the best of our knowledge, there is no asymptotic analysis of the DIRECT algorithm beyond the error coverging to 0.

In this dissertation, we discuss the convergence rate of a new global optimization algorithm. The argument is based on the asymptotic analysis of [5]. Theoretical work on the convergence of global optimization to the global minimum appears in [53, 78]. The authors provide asymptotic analyses of a simultaneous optimistic optimization approach, which can be considered as a generalization of the DIRECT algorithm [53], and compare its convergence rate to that of DIRECT algorithm. Some researchers [53] have presented global optimization algorithms with errors of order $\exp(-c\sqrt{n})$ after n function evaluations. We conjecture that the convergence rate of the DIRECT algorithm is also of the same order, and supply some experimental evidence in Section 2.4. Our main contribution is to provide the asymptotic convergence rate for a new DIRECT type global optimization algorithm. We establish that the error is of order $\exp(-cn/\log(n))$ as $n \to \infty$. In this expression, the number c depends on the dimension of the domain and the objective function f, but not on n. This result is given by Theorem 2 in Section 2.3.

In addition, we use classical benchmark functions such as quadratic, Rastrigin, and Branin functions to test the new global optimization algorithm. We also use the GKLS [46] software package which generates objective functions with known optima. The experiments we performed indicated that the new global optimization algorithm works well on those functions.

We are interested in using the new global optimization algorithm on the image registration and clustering problems. Image registration is the process of identifying a transformation that best aligns a *moving image* with a *fixed image*. Given an objective function that measures dissimilarity of images, registration becomes a global optimization problem. Many registration methods [8, 63] require the specification of a "starting point" in parameter space, from which an iterative local optimization method proceeds. Most of these methods are sensitive to this starting point, and if

this point is not close enough to the global minimum, the method can get "stuck" in a local minimum [25].

Data clustering is another problem which can be solved by global optimization algorithms. The task of clustering is to divide the input points into different groups by their similarities. We can use different objective cost functions, for example defined by squared Euclidean distance [1]. With the cluster problem, we can regard the classical k-means algorithm as a local optimization algorithm. To get a better solution for clustering problems, we apply the proposed algorithm to find a good starting point for the local search.

The proposed algorithm works with continuous decision variables. Similar optimization problems exists with discrete decision variables and with noise corrupted function evaluations. We describe an algorithm for that setting in Chapter 5.

The structure of this dissertation is as follow. In Chapter 2, we describe the new global optimization algorithm which approximates an optimal point after a fixed number of iterations. We also prove that the global optimization algorithm converges to the global optimum and the convergence rate of the proposed algorithm is provided in Chapter 2. In Chapter 3, we apply the new global optimization algorithm to the image registration problem and compare the algorithm with other global optimization algorithms such as exhaustive search and simulated annealing. In addition, we also apply it to clustering. Chapter 4 provides a discussion of a new 1-D optimization algorithm which uses function values and derivatives and provides a proof of the convergence rate of the algorithm. In Chapter 5, we provide an algorithm to choose a parameter from a finite set that optimizes the long-run average performance of a stochastic system. Chapter 6 contains a short summary and outlines the future problems we want to solve.

CHAPTER 2

A NEW DIRECT TYPE GLOBAL OPTIMIZATION ALGORITHM

2.1 Background and Related Work

"Consider everything. Keep the good. Avoid evil whenever you notice it.[56]" This quote shows the core idea about optimization. When people try to solve problems, it's usually with a process of optimizing a performance measure for a model of the problem. Optimization algorithms include local optimization algorithms and global optimization algorithms. Local optimization algorithms such as gradient descent are easy and efficient for finding local minima but limited in their ability to find the global optimal solution. A global optimization algorithm seeks global solutions of a constrained or unconstrained optimization model. In contrast with local optimization algorithms, global optimization algorithms must avoid being trapped at local optimum. We hope to spend less resources such as running time and gain more which is the main motivation for global optimization. There are several different ways to organize the optimization problem. To formulate the problem of global optimization, assume that the objective function f and the constraints are continuous functions and the feasible set is nonempty. In a simpler version of the problem, we begin by considering optimization with "box" constraints.

The problem itself is easy to describe: approximate the minimum of a cost function over a feasible region. The mathematical way to present a minimization problem is as follows. We are given a compact feasible region $K \subset \mathbb{R}^d$ for some $d \geq 1$, and a function $f : K \to \mathbb{R}$. We assume that $f \in F(K) = F$, for some class of functions F. We define the global optimization problem as:
Find the couple t^*, f^* such that:

$$f^* = f(t^*) \leq f(x) \quad \forall x \in K. \tag{2.1}$$

We can sequentially choose points $t_1, t_2, \cdots, t_i, \cdots \in K$ at which to observe some information $f[t_i]$ about f. We mostly (for now) take $f[t_i] = f(t_i)$.

A global optimization algorithm comprises:

1. A sequence of maps $t_j = t_j(f[t_1], f[t_2], \ldots, f[t_{j-1}])$, taking values in K, which give the j^{th} point to evaluate $f[t_j]$.

2. A stopping rule that determines the time T when the algorithm halts.

3. A function $A(f[t_1], f[t_2], \cdots, f[t_T])$ that gives our approximation of f^*.

An algorithm is non-adaptive (passive) if the mappings t_j are the same for all $f \in F$; otherwise the algorithm is adaptive. The algorithm is deterministic if the t_j are deterministic functions of f.

In the problem we defined above, the main purpose of the global optimization algorithm is to approximate the best solution or the best point t^* which satisfies (2.1).

There are many different approaches to solve global optimization problems. If we use traditional local-scope search methods to solve this problem, then depending on the starting point of the search, we will often find locally optimal solutions of varying quality (the "valleys" in Figure 2.1 below could easily trap local search methods).

In order to find the globally optimal solution, a global-scope search effort is needed. We can divide global optimization algorithms into two different tracks. One of them is deterministic optimization and another is stochastic optimization. Some methods to solve the global optimization problem are listed in following.

The simplest approach for approximating the optimal solution is evaluating the function value over the whole feasible region. These include the most well-known passive or direct sequential global optimization strategies such as uniform grid exhaustive search [91], space covering [28] exhaustive search [61], and pure random search [90]. They are not adaptive. These methods converge under mild assumptions

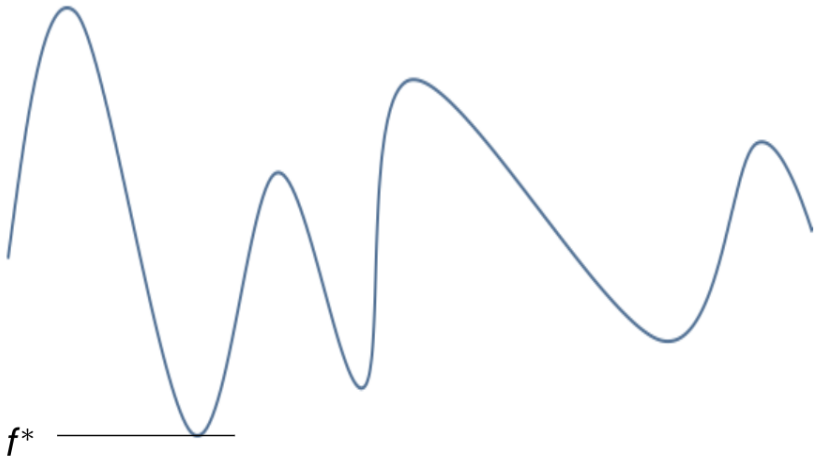

Figure 2.1 A 1-D continuous function with several local optima.

but, as a rule, are not practical in higher dimensional problems because of their inefficiency.

Pure random search is a simple global random search algorithm which takes several random independent points and evaluates the objective functions at those points [90].

The trajectory method is based on the construction of a path along which the gradient of the function points is in a constant direction [4]. The tunneling approach usually includes two phases. The first phase is the minimization phase which searches for a local minimum. The second tunneling phase finds a different point which has the same function value as the local minima point in the first phase. Then do the next minimization phase. The new stationary point will have a function value no greater than the previous minimum found [34].

These methods have the "ambitious" objective of visiting all stationary points of the objective function: this, in turn, leads to the list of all (global as well as local) optima. This general approach includes differential equation model based, path-following search strategies, as well as fixed-point methods and pivoting algorithms [12].

These are heuristic methods, for which conditions for convergence are difficult to determine.

Branch and bound algorithms are designed for discrete and combinational problems [27]. It subsumes many specific approaches, and allows for a variety of implementations. Branch and bound methods typically rely on some a priori structural knowledge about the problem. This information may relate, for instance, to how rapidly each function can vary, or to the availability of an analytic formulation and guaranteed smoothness of all functions (for instance, in interval arithmetic-based methods).

The general branch and bound methodology is applicable to broad classes of global optimization problems, e.g., in combinatorial optimization [57], concave minimization [26], reverse convex programs [35], and Lipschitz optimization [65]. There is another broad class of methods, based upon "exhaustive" random sampling in the feasible set. In its basic form, it includes various random search strategies that are convergent, with probability one. Search strategy adjustments, clustering and deterministic solution refinement options, statistical stopping rules, etc. can also be added as enhancements. The methodology is applicable to both discrete and continuous global optimization problems under very mild conditions [3, 91].

Simulated annealing is based upon the physical analogy of cooling crystal structures that spontaneously attempt to arrive at some stable (globally or locally minimal potential energy) equilibrium. This general principle is applicable to both discrete [40] and continuous global optimization problems under mild structural requirements [58].

Genetic algorithms (GAs) are meta-heuristics inspired by the process of natural selection that belong to the larger class of evolutionary algorithms (EA). In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each

candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0's and 1's, but other encodings are also possible [85].

2.2 The Algorithm

There are many global optimization algorithms listed in the last section but some of them require a good starting point and convergence rates are not always known. We design a global optimization algorithm in this section that doesn't requires a starting point and only uses function values to compute the next point at which to evaluate the function. In addition, we prove that the algorithm converges and establish its convergence rate.

The algorithm subdivides the feasible region into subrectangles and sequentially chooses the next subrectangle to subdivide according to a certain numerical value. We use the rectangular subdivision introduced in [33] which is called the DIRECT algorithm. The criteria for selecting the next subrectangle to subdivide is similar to the criteria used in [5].

Suppose the domain $K = [0,1]^d$ and the objective function is a continuous but not necessarily convex unknown function $f(x)$ where $x \in K$. The algorithm operates by decomposing $[0,1]^d$ into hyper-rectangles as follows. The first operation is to evaluate f at the center of the unit hyper-cube $(1/2, 1/2, \ldots, 1/2)$. Given a current decomposition, choose one of the hyper-rectangles (according to the maximal value of a criterion to be defined below) and trisect it along the longest axis. The central sub-hyperrectangle retains its central function value, while the function is evaluated at the centers of the two other hyperrectangles.

After k iterations of the algorithm, f will have been evaluated $2k+1$ times and the unit hyperrectangle will have been decomposed into $2k+1$ sub-hyperrectangles.

It will be convenient to index quantities to be defined by the iteration number of the algorithm, which we denote by n, instead of the number of function evaluations, $N = 2n + 1$.

After n iterations of the algorithm, let $M_n = \min_{1 \le i \le N} f(c_i)$ where c_i is the center of R_i and denote the error by $\Delta_n = M_n - f(t^*)$ where t^* is a global optimizer of f.

Define

$$g_n(x) \equiv d\,(x \log(n))^{2/d},$$

for $0 < x \le 1/2$ and $g(1) = 1$. Note that g_n is increasing and $g_n(x) \downarrow 0$ as $x \downarrow 0$.

The term $g_n(v_n)$ where v_n is the volume of smallest hyper-rectangles serves the role of an approximate upper bound on the error. That is, eventually the global minimum of the function should be above $M_n - g_n(v_n)$. If the function is twice continuously differentiable near the global minimizer, then the difference between M_n and the global minimum should be proportional to the square of the diameter of the smallest rectangle, which is approximately its volume raised to the power $2/d$. The logarithmic term ensures convergence.

Our algorithm will assign a numerical value to each hyper-rectangle in the subdivision. After n iterations of the algorithm, for each hyper-rectangle R_i, $1 \le i \le 2n + 1$, note that $|R_i|$ is the volume of hyper-rectangle R_i, set

$$\rho_i^n \equiv \frac{|R_i|^{2/d}}{f(c_i) - M_n + g_n(v_n)} \tag{2.2}$$

.

If we are about to subdivide the smallest hyper-rectangle, then

$$\rho_i^n \le \frac{|R_i|^{2/d}}{g_n(v_n)} = \frac{v_n^{2/d}}{d(v_n \log(n))^{2/d}} = \frac{1}{d \log(n)^{2/d}}. \tag{2.3}$$

The choice of g_n and ρ_i^n is based on efficiency considerations; g_n must be large enough to ensure convergence yet small enough to lead to an efficient search.

The form of the criteria ρ_i^n is motivated by the Bayesian approach to optimization, for which choosing large ρ_i^n implies a high likelihood of the minimum over the rectangle lying below $M_n - g_n(v_n)$ [5]. Let us suppose that f is a Gaussian random function on $[0,1]^d$ with three-times continuously differentiable sample paths. Taylor's formula gives

$$f(x+y) \approx f(x) + \nabla f(x) \cdot y + \frac{1}{2} y' \nabla^2 f(x) y.$$

Then if the gradient is near 0, as is the case near a minimum, then a lower bound on the minimum of f over a ball of radius h centered at x is

$$f(x) - \frac{1}{2} \nabla^2 f \cdot h^2,$$

where h is the maximum distance from the center to any other point in the rectangle. Because of the partitioning scheme, which bounds the aspect ratio of the rectangles, the maximum distance from the center to an extreme point of a rectangle with volume V is $(3/2)\sqrt{d} \cdot V^{1/d}$. Thus, the lower bound is

$$f(x) - \frac{1}{2} \nabla^2 f (3/2)^2 d V^{2/d}.$$

The algorithm described chooses the rectangle with the lowest such bound. If the lower bound falls below $M_n - g_n$, that would entail (assuming the gradient is zero) that

$$f(x) - \frac{1}{2} \nabla^2 f (3/2)^2 d V^{2/d} < M_n - g_n,$$

or

$$\nabla^2 f > \frac{f(x) - M_n + g_n}{c V^{2/d}} = \frac{1}{c (\hat{\rho}_i^n)^{d/2}}$$

for a constant $c > 0$. We choose the rectangle with the minimum of the right-hand side, or maximum $\hat{\rho}_i^n$. This way, as $\max_i \hat{\rho}_i^n \downarrow 0$, the minimum over all rectangles lies above $M_n - g_n$ unless $\nabla^2 f$ is increasingly large.

A more formal description of the algorithm follows. In the description, N is the total number of function evaluations to make and j is the number of function evaluations that have been made. Let v, the volume of the smallest hyperrectangle, take initial value 1 and let M, the minimum of the observed function values, take initial value $f(1/2, \ldots, 1/2)$.

The algorithm comprises the following steps.

1. We start with the hyperrectangle $[0,1]^d$ and with the function value at the center. Set j, the number of function evaluations (=number of hyperrectangles), to 1.
2. For each hyperrectangle R_i in the current collection $\{R_1, R_2, \ldots, R_j\}$, compute ρ_i^j, keeping track of the rectangle with the lowest index γ that has the maximal value of ρ_i^j.
3. From the hyperrectangle with maximal value ρ_γ^j form three new hyperrectangles as follows. Suppose that

$$R_\gamma = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d],$$

and that k is the smallest index with $b_k - a_k \geq b_i - a_i$ for $1 \leq i \leq d$. The three new hyperrectangles are

   (a) $R'_\gamma \equiv [a_1, b_1] \times \cdots \times [a_k, a_k + (a_k + b_k)/3] \times \cdots \times [a_d, b_d]$

   (b) $R''_\gamma \equiv [a_1, b_1] \times \cdots \times [a_k + (a_k + b_k)/3, a_k + 2(a_k + b_k)/3] \times \cdots \times [a_d, b_d]$

   (c) $R'''_\gamma \equiv [a_1, b_1] \times \cdots \times [a_k + 2(a_k + b_k)/3, b_k] \times \cdots \times [a_d, b_d]$.

   Evaluate f at the centers of R' and R''. If $|R_\gamma| = v$, then set $v \leftarrow v/3$, and if the smallest of the new function values is less than the previously observed

minimum M, then set M to that new smallest value. Increment j by the number of new function evaluations $j \leftarrow j + 2$.

4. If $j < N$, return to step 2.

The only stopping rule that we consider is to initially specify a fixed number of function evaluations after which the algorithm terminates. For running time, the hyperrectangles can be stored in a heap-based priority queue keyed on ρ values. If either M_i or v_i changes on iteration i, then the ρ values must be recomputed and the heap re-formed, taking time $\Theta(i)$. In the worst-case, this occurs on each iteration, resulting in a quadratic run time. If M_i or v_i changes on only K_n of the iterations, then the run time is $O(nK_n + n \lg n)$. In our experiments it appeared that K_n grew logarithmically.

The proposed global optimization algorithm is aimed at a broader class of optimization problems and image registration is just one of the applications. The required property of the objective function is continuity, though the algorithm should be adjusted for the application at hand and the local search may not be appropriate for some applications. Applications such as clustering can use global optimization algorithms to get better clusters [29].

In the next section, we will prove that the algorithm convergences to the optimal point when $N \to \infty$.

2.3 Convergence

Theorem 1. *If f is continuous, then the error $\Delta_n = M_n - f(t^*)$ converges to 0 where $M_n = \min_{1 \le i \le n} f(c_i)$.*

Proof. To prove that $\Delta_n = M_n - f(t^*)$ converges to 0, we need to prove that for any $x \in [0,1]^d$, there is a subsequence of $\{c_i\}$ converging to x.

To obtain a contradiction, assume that there exists a point $x \in [0,1]^d$ with no subsequence of c_i converging to x.

There will be an infinite sequence of times $n_1, n_2, \ldots$ at which a new smallest rectangle is formed. Let $\rho_s^{n_k}$ be the ρ value for a smallest rectangle that is about to be split at time n_k. Then $\rho_s^{n_k} \to 0$ as $k \to \infty$.

From the definition of ρ, we know

$$\rho_s^{n_k} \equiv \frac{|R_s^{n_k}|^{2/d}}{f(c_s) - M_{n_k} + g(v_{n_k})}, \tag{2.4}$$

where c_s is the center of $R_s^{n_k}$ and $R_s^{n_k} = v_{n_k}$. Then

$$\rho_s^{n_k} = \frac{|R_s^{n_k}|^{2/d}}{f(c_s) - M_{n_k} + g(v_{n_k})} = \frac{|v_n|^{2/d}}{f(c_s) - M_{n_k} + g(v_{n_k})} = \frac{|v_{n_k}|^{2/d}}{f(c_s) - M_{n_k} + d\left(v_{n_k}\log(n)\right)^{2/d}}. \tag{2.5}$$

As we already know $f(c_s) \geq M_n$, we can get

$$\rho_s^{n_k} \leq \frac{|v_{n_k}|^{2/d}}{d\left(v_{n_k}\log(v_{n_k})\right)^{2/d}} \leq \frac{1}{d\left(\log(v_{n_k})\right)^{2/d}}. \tag{2.6}$$

As $n_k \to \infty$, we have $d\left(v_{n_k}\log(v_{n_k})\right)^{2/d} \to \infty$, and So $\frac{1}{d\left(\log(v_{n_k})\right)^{2/d}} \to 0$. By (3.1), we know that $\rho_s^{n_k} \to 0$ when $n_k \to 0$.

The ρ of smallest rectangles which are split converges to 0 when $n_k \to \infty$.

For other rectangles which are not smallest, their ρ value is positive:

$$\rho_i^{n_k} = \frac{|R_i^{n_k}|^{2/d}}{f(c_i) - M_{n_k} + g(v_{n_k})} = \frac{|R_i^{n_k}|^{2/d}}{f(c_i) - M_{n_{n_k}} + d\left(v_{n_{n_k}}\log(n_{n_k})\right)^{2/d}} \geq \frac{|R_i^{n_k}|^{2/d}}{f(c_i) - M_{n_k} + d\left(\log(n_k)/n_k\right)^{2/d}}, \tag{2.7}$$

When $n_k \to \infty$, $d\left(\log(n_k)/n_k\right)^{2/d} \to 0$, so we have

$$\liminf_{n_k\to\infty} \rho_i^{n_k} = \liminf_{n_k\to\infty} \frac{|R_i^{n_k}|^{2/d}}{f(c_i) - M_{n_k} + d\left(v_{n_k}\log(n_k)\right)^{2/d}} \geq \liminf_{n_k\to\infty} \frac{|R_i^{n_k}|^{2/d}}{f(c_i) - f(t^*)} > 0, \tag{2.8}$$

which means at some point, the algorithm is going to split these rectangles which are not smallest. It means all rectangles are being split for $\rho_i^{n_k} \leq \rho_s^{n_k}$ when it's the

moment to split the smallest rectangle. It is a contradiction to there are some center points in rectangles x_i which are never split. This means our assumption $\exists x \in [0,1]^d$, there is no $\{c_i\} \to x$ is false. Then we know $\forall x \in [0,1]^d$, there exists a subsequence of $\{c_i\}$ that converges to x. We conclude that $\{c_i\}$ is dense in $[0,1]^d$.

We already have $M_n = \min_{1 \le i \le n} f(c_i)$ and let c_n^* be the point of first n that is closest to an optimizer. We have that $M_n \le f(c_n^*)$, so $\Delta_n = M_n - f(t^*) \le f(c_n^*) - f(t^*)$. We already know that for any $x \in [0,1]^d$, there is a subsequence of c_i converging to x and f is continuous, so $f(c_n^*) - f(t^*)$ converges to 0 when $n \to \infty$.

□

2.3.1 Convergence rate

This algorithm has the property that $\Delta_n \to 0$ as $n \to \infty$ for any continuous function f. In order to determine the rate at which the error converges to 0 we need to place some assumptions on f beyond continuity.

Assume that the minimizer t^* is unique and lies in the interior of $[0,1]^d$. Assume that f is twice continuously differentiable on $[0,1]^d$. Let c_i^n denote the center of rectangle R_i^n after n iterations. Let $M = f(t^*)$ and denote the smallest function value after n evaluations by $M_n = \min_{1 \le i \le n}\{f(c_i^n)\}$ and the error by $\Delta_n = M_n - M$. Let $v_n = \min_{1 \le i \le n} |R_i^n|$ denote the smallest rectangle volume where $|R_i^n|$ is the volume of the ith rectangle.

Since $f \in C^2$ and t^* is in the interior of $[0,1]^d$, there exists a positive number β such that the ball of radius β is contained in $[0,1]^d$ and f is convex on the ball $B_\beta(t^*) = \{t \in [0,1]^d : \|t - t^*\| \le \beta\}$.

Let $\nabla^2 f(t^*)$ denote the matrix of second partial derivatives at the minimizer t^*. We assume $\nabla^2 f(t^*)$ is positive definite, with eigenvalues $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_d$. Because t^* is the minimizer, $\nabla f(t^*) = 0$. By Taylor's Theorem, we have $f(s) \le f(t^*) + (s - t^*)^T \nabla^2 f(t^*)(s - t^*)$ for $s \in B_\beta(t^*)$ for sufficiently small β.

Our main result is:

Theorem 2. *Let f be twice continuously differentiable with unique global minimizer in the interior of $[0,1]^d$. Then the normalized error after n evaluations is bounded asymptotically as:*

$$\liminf_{n\to\infty} \frac{\log(n)}{n} \log\left(\frac{1}{\Delta_n}\right) \geq \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}$$

where $\gamma = \frac{d(2\pi)^{d/2}}{2\Gamma(1+d/2)}(\det\nabla^2 f(t^*))^{-1/2}$ *and* $\alpha = \max_{1\leq i,j\leq d}\max_{c\in[0,1]^d}\left(\left|\nabla^2 f(c)_{i,j}\right|, \left|\nabla f(c)_j\right|\right)$.

To prove Theorem 2, we need a technical lemma.

Lemma 3. *There exists a postive number β such that for sufficiently large n,*

$$\int_{B_\beta(t^*)} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}} \geq \int_{B_\beta(t^*)} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(2+\frac{9d^{3/2}\alpha}{\lambda_1}\right)^{d/2}}.$$

Proof. For each rectangle R_i^n which is in $B_\beta(t^*)$, we have

$$\begin{aligned}
&\int_{R_i^n} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}} \\
&\quad= \int_{R_i^n} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(\frac{f(s)-f(c_i^n)+f(c_i^n)-M+M_n-M_n+g_n(v_n)}{f(c_i^n)-M_n+g_n(v_n)}\right)^{d/2}} \\
&\quad\geq \int_{R_i^n} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(1+\frac{\left|f(s)-f(c_i^n)\right|+M_n-M}{f(c_i^n)-M_n+g_n(v_n)}\right)^{d/2}}. \qquad (2.9)
\end{aligned}$$

By Taylor's theorem with Peano's form of remainder, we get

$$\begin{aligned}
&\frac{|f(s)-f(c_i^n)|+M_n-M}{f(c_i^n)-M_n+g_n(v_n)} \\
&\quad= \frac{\nabla f(c_i^n)(s-c_i^n)+\frac{1}{2}(s-c_i^n)^T\nabla^2 f(c_i^n)(s-c_i^n)+o(s-c_i^{n2})+M_n-M}{f(c_i^n)-M_n+g_n(v_n)}. \qquad (2.10)
\end{aligned}$$

Suppose that w_i^j is the jth dimensional width of the ith rectangle. For any $s\in R_i^n\subset B_\beta(t^*)$, we have

$$\frac{\nabla f(c_i^n)(s-c_i^n)+\frac{1}{2}(s-c_i^n)^T\nabla^2 f(c_i^n)(s-c_i^n)+o(s-c_i^{n2})+M_n-M}{f(c_i^n)-M_n+g_n(v_n)}$$
$$=\frac{\nabla f(c_i^n)(s-c_i^n)}{f(c_i^n)-M_n+g_n(v_n)}+\frac{1}{2}\frac{(s-c_i^n)\nabla^2 f(c_i^n)(s-c_i^n)}{f(c_i^n)-M_n+g_n(v_n)}$$
$$+\frac{o(s-c_i^{n2})}{f(c_i^n)-M_n+g_n(v_n)}+\frac{M_n-M}{f(c_i^n)-M_n+g_n(v_n)}$$
$$=\frac{\nabla f(c_i^n)(s-c_i^n)}{f(c_i^n)-M_n+g_n(v_n)}+\frac{1}{2}(s-c_i^n)^T\nabla^2 f(c_i^n)(s-c_i^n)\frac{\rho_i^{2/d}}{(\prod_{j=1}^d w_i^j)^{2/d}}+$$
$$\frac{o(s-c_i^{n2})}{f(c_i^n)-M_n+g_n(v_n)}+\frac{M_n-M}{f(c_i^n)-M_n+g_n(v_n)}. \quad (2.11)$$

We will discuss these terms separately.

If $R_i^n \subset B_\beta(t^*)$, then

$$\frac{1}{3}(c_i^n-t^*)^T\lambda_1(c_i^n-t^*)\le f(c_i^n)-f(t^*)\le\frac{2}{3}(c_i^n-t^*)^T\lambda_d(c_i^n-t^*), \quad (2.12)$$

and

$$\max_{j,k}\nabla f(c_i^n)_{j,k}\le\alpha c_i^n-t^*. \quad (2.13)$$

Then we have

$$\nabla f(c_i^n)(s-c_i^n)\le\alpha c_i^n-t^*s-c_i^n \quad (2.14)$$

and

$$f(c_i^n)-M_n+g_n(v_n)\ge\frac{1}{3}c_i^n-t^{*2}\lambda_1+M-M_n+g_n(v_n). \quad (2.15)$$

Suppose that $v_n=\prod_{j=1}^d\tau_j$ and $\tau=\min_{1\le j\le d}\tau_j$, θ of widths are τ, and the rest of the widths are 3τ. We have $1\le\theta\le d$. Then $v_n^{2/d}=\tau^2*3^{(d-\theta)\frac{2}{d}}$, so we get that $\tau^2=3^{-(d-\theta)*\frac{2}{d}}v_n^{\frac{2}{d}}$. By our previous proof of convergence, the rectangle algorithm

will keep splitting rectangles which have larger ρ_i^n and when $n \to \infty$, all rectangles' volume will tend to 0. It means the c_{i*}^n will eventually be contained in one of the smallest rectangles. Then if c_{i*}^n is contained in the one of the smallest rectangles, we get

$$\begin{aligned} M_n - M &\leq \frac{1}{2}\lambda_d \frac{1}{4}\sum_{j=1}^{d}\tau_j^2 \leq \frac{1}{8}\lambda_d[\theta\tau^2 + (d-\theta)9\tau^2] \\ &= \frac{1}{8}\lambda_d\tau^2[\theta + (d-\theta)9] \leq \frac{1}{8}\lambda_d v_n^{2/d}[\theta + (d-\theta)9]. \end{aligned}$$

For $\theta = 1$, $\theta + 9(d-\theta)$ achieves the maximum $9d-8$. Then we have

$$\frac{1}{8}\lambda_d v_n^{2/d}[\theta + (d-\theta)9] \leq \frac{1}{8}\lambda_d v_n^{2/d}(9d-8) = \frac{9d-8}{8}\lambda_d v_n^{2/d}. \tag{2.16}$$

By definition of $g_n(v_n)$, we know that

$$g_n(v_n) = d(v_n \log(n))^{2/d} = d(\log n)^{2/d} v_n^{2/d}.$$

Then

$$\begin{aligned} \frac{1}{8}\lambda_d v_n^{2/d}[\theta + (d-\theta)9] &= g_n(v_n)\frac{\frac{1}{8}\lambda_d[\theta + (d-\theta)9]}{d(\log n)^{2/d}} \\ &= \frac{9d-8}{8d} g_n(v_n)\frac{\lambda_d}{(\log n)^{2/d}} \end{aligned}$$

if

$$\frac{9d-8}{8d}\frac{\lambda_d}{(\log n)^{2/d}} \leq 1.$$

If $\log(n) \geq \left(\frac{(9d-8)\lambda_d}{8d}\right)^{d/2}$, then

$$\frac{9d-8}{8d} g_n(v_n)\frac{\lambda_d}{(\log n)^{2/d}} \leq g_n(v_n). \tag{2.17}$$

Therefore, we conclude that $M_n - M \leq g_n(v_n)$ and combine with (2.12) to get $f(c_i^n) - M_n + g_n(v_n) \geq \frac{1}{3}c_i^n - t^{*2}\lambda_1$.

We will show that

$$c_i^n - t^* s - c_i^n \leq 3\sqrt{d}c_i^n - t^{*2}. \tag{2.18}$$

As for $s - c_i^n$,

$$s - c_i^n \leq \sqrt{\sum_{j=1}^{d}\left(\frac{1}{2}w_i^j\right)} \leq \frac{1}{2}\sqrt{d}\max w_i^j \tag{2.19}$$

and

$$c_i^n - t^* \geq \frac{1}{6}\max_j w_i^j. \tag{2.20}$$

Then

$$s - c_i^n \leq \frac{1}{2}\sqrt{d}\max_j w_i^j \leq 3\sqrt{d}c_i^n - t^*. \tag{2.21}$$

Therefore,

$$\frac{\nabla f(c_i^n)(s - c_i^n)}{f(c_i^n) - M_n + g_n(v_n)} \leq \frac{9d^{3/2}\alpha c_i^n - t^{*2}}{\lambda_1 c_i^n - t^{*2}} = \frac{9d^{3/2}\alpha}{\lambda_1}.$$

Suppose $w = \min_j w_i^j$ and $\max_j w_i^j \leq 3w$. As for the second term in the formula (2.11), we get

$$\begin{aligned}
\frac{1}{2}\frac{(s - c_i^n)^T\nabla^2 f(c_i^n)(s - c_i^n)\rho_i^{2/d}}{(\prod_{j=1}^d w_i^j)^{2/d}} &= \frac{1}{2}\frac{(s - c_i^n)^T\lambda_i(s - c_i^n)\rho_i^{2/d}}{(\prod_{j=1}^d w_i^j)^{2/d}} \\
&\leq \frac{1}{2}\frac{(s - c_i^n)^T\lambda_d(s - c_i^n)\rho_i^{2/d}}{(\prod_{j=1}^d w_i^j)^{2/d}} \leq \frac{1}{2}\frac{\lambda_d\sum_{j=1}^d (w_i^j)^2\rho_i^{2/d}}{(\prod_{j=1}^d w_i^j)^{2/d}} \\
&\leq \frac{1}{18}\frac{\lambda_d d w^2\rho_i^{2/d}}{w^2} = \frac{d\lambda_d\rho_i^{2/d}}{18}.
\end{aligned}$$

For large enough n, the term $o(s - c_i^{n2}) \leq \frac{(s-c_i^n)\nabla^2 f(c_i^n)(s-c_i^n)}{2}$. As for the third term in formula (2.11), for sufficiently large n,

$$\frac{o(s-c_i^{n2})}{f(c_i^n)-M_n+g_n(v_n))} \leq \frac{\frac{1}{2}(s-c_i^n)^T\nabla^2 f(c_i^n)(s-c_i^n)}{f(c_i^n)-M_n+g_n(v_n)} \leq \frac{d\lambda_d\rho_i^{2/d}}{18}. \tag{2.22}$$

The last term in formula (2.11) can be bounded as follows.

We know that $M_n - M \leq g_n(v_n)$ for sufficiently large n. Then

$$\frac{M_n-M}{f(c_i^n)-M_n+g_n(v_n)} \leq 1. \tag{2.23}$$

We conclude that for large enough n

$$\int_{R_i^n} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}} \geq \int_{R_i^n} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(2+\frac{9d^{3/2}\alpha}{\lambda_1}\right)^{d/2}}$$

holds for every rectangle contained in $B_\beta(t^*)$.

Therefore we cancluded that for sufficiently large n,

$$\int_{B_{\beta/2}(t^*)} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}} \geq \int_{B_{\beta/2}(t^*)} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(2+\frac{9d^{3/2}\alpha}{\lambda_1}\right)^{d/2}}$$

also holds.

Then, we get

$$\int_{B_\beta(t^*)} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}} \geq \int_{B_\beta(t^*)} \frac{ds}{(f(c_i^n)-M_n+g_n(v_n))^{d/2}\left(2+\frac{9d^{3/2}\alpha}{\lambda_1}\right)^{d/2}}.$$

□

The following proof is for Theorem 2.

Proof. From [5], we know

$$\lim_{\epsilon\to 0} \frac{\int_{[0,1]^d} \frac{ds}{(f(s)-M+\epsilon)^{d/2}}}{\log(1/\epsilon)} = \frac{d(2\pi)^{d/2}}{2\Gamma(1+d/2)}(\det\nabla^2 f(t^*))^{-1/2} = \gamma.$$

Then,

$$\lim_{g_n(v_n)\to 0} \frac{\int_{[0,1]^d} \frac{ds}{(f(s)-M+g_n(v_n))^{d/2}}}{\log(1/g_n(v_n))} = \gamma. \tag{2.24}$$

Using the fact that

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} \rho_i^n &= \frac{1}{n}\int_{[0,1]^d} \frac{ds}{\sqrt{f(c_i^n) - M_n + g_n(v_n)}} \\
&= \frac{1}{n}\int_{B_\beta(t^*)} \frac{ds}{\sqrt{f(c_i^n) - M_n + g_n(v_n)}} + O(1/n) \\
&\le \left(\frac{2+3d\alpha}{a}\right)^{d/2} \frac{1}{n}\int_{B_\beta(t^*)} \frac{ds}{\sqrt{f(s) - M + g_n(v_n)}} + O(1/n) \\
&\le \left(\frac{2+3d\alpha}{a}\right)^{d/2} \frac{1}{n}\int_{[0,1]^d} \frac{ds}{\sqrt{f(s) - M + g_n(v_n)}} + O(1/n) \\
&= \log(1/g_n(v_n)) \left(\frac{2+3d\alpha}{a}\right)^{d/2} \frac{1}{n} \frac{\int_{[0,1]^d} \frac{ds}{\sqrt{f(s)-M+g_n(v_n)}}}{\log(1/g_n(v_n))} + O(1/n).
\end{aligned}$$

Therefore, for large enough n,

$$\frac{1}{n}\sum_{i=1}^{n} \rho_i^n \le \left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\log(1/g_n(v_n))\gamma + O(1/n). \tag{2.25}$$

Suppose that s denotes the index of the subrectangle containing the minimizer. Since this rectangle eventually has volume at most $3^d v_n$,

$$\rho_s^n \le \frac{3^d v_n}{2g_n(v_n)^{d/2}} = \frac{3^d}{2\log n}.$$

In the neighborhood of t^* where f is convex, the ρ-values for each child will be at least a third the value of the parent's. For subrectangles outside the neighborhood of t^*, the children of split rectangles will have ρ values about a third of the parent's. Thus for all sufficiently large n,

$$\rho_s^n \ge \frac{3^d}{6\log n}$$

and

$$\frac{1}{3} \leq \liminf_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \leq \limsup_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \leq 3. \tag{2.26}$$

Therefore,

$$\begin{aligned}
3 &\geq \limsup_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \\
&\geq \limsup_{n\to\infty} \frac{\frac{3^d}{6\log n}}{\left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\log(1/g_n(v_n))\gamma + O(1/n)} \quad \text{(using 2.25)} \\
&\geq \limsup_{n\to\infty} \frac{3^{d-1}}{2\log n \left\{\left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\log(1/g_n(v_n))\gamma + O(1/n)\right\}},
\end{aligned}$$

Then

$$\liminf_{n\to\infty} \log n \left\{\left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\log(1/g_n(v_n))\gamma + O(1/n)\right\} \geq 3^{d-2}/2. \tag{2.27}$$

Therefore,

$$\begin{aligned}
3^{d-2}/2 &\leq \liminf_{n\to\infty} \log n \left\{\left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\log(1/g_n(v_n))\gamma\right\} \\
&= \liminf_{n\to\infty} \log n \left\{\left(\frac{2+3d\alpha}{\lambda_1}\right)^{d/2} \frac{1}{n}\left(-\log d + \frac{2}{d}\log\left(\frac{1}{v_n}\right) - \frac{2}{d}\log\log(n)\right)\gamma\right\}.
\end{aligned}$$

Then

$$\liminf_{n\to\infty} \frac{\log n}{n}\left\{-\log d + \frac{2}{d}\log(\frac{1}{v_n}) - \frac{2}{d}\log\log(n)\right\} \geq \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}.$$

Since $v_n \leq \frac{1}{n}$, $\log(\frac{1}{v_n}) \geq \log(n) \geq \log\log(n) \geq 0$. This implies that

$$\liminf_{n\to\infty} \frac{\log(n)}{n}\frac{2}{d}\log\left(\frac{1}{v_n}\right) \geq \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}. \tag{2.28}$$

Then

$$\liminf_{n\to\infty} \frac{\log(n)}{n}\log\left(\frac{1}{v_n}\right) \geq \frac{d3^{d-2}}{4\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}. \tag{2.29}$$

Recall that $\Delta_n = M_n - M$, we have for sufficiently large n that $M_n - M \leq g_n(v_n)$ by formula (2.17).

By the definition of $g_n(v_n)$, we have

$$\log \frac{1}{g_n(v_n)} = \log \frac{1}{d(v_n \log(n))^{2/d}}. \tag{2.30}$$

Then

$$\begin{aligned}\frac{\log(n)}{n} \log\left(\frac{1}{\Delta_n}\right) &\geq \frac{\log(n)}{n} \log\left(\frac{1}{g_n(v_n)}\right) \\ &= \frac{\log(n)}{n} \log \frac{1}{d(v_n \log(n))^{2/d}} \\ &= \frac{\log(n)}{n}\left(-\log(d) + \frac{2}{d}\log\left(\frac{1}{v_n}\right) - \frac{2}{d}\log\log(n)\right). \end{aligned} \tag{2.31}$$

Therefore

$$\begin{aligned}\liminf_{n\to\infty} \frac{\log(n)}{n} \log\left(\frac{1}{\Delta_n}\right) &\geq \liminf_{n\to\infty} \frac{\log(n)}{n}\left[-\log(d) + \frac{2}{d}\log\left(\frac{1}{v_n}\right) - \frac{2}{d}\log\log(n)\right] \\ &\geq \liminf_{n\to\infty}\left[\frac{-\log(d)\log(n)}{n} + \frac{2}{d}\frac{\log(n)}{n}\log\left(\frac{1}{v_n}\right) - \frac{2\log(n)\log\log(n)}{dn}\right]. \end{aligned} \tag{2.32}$$

When $n \to \infty$, $\frac{\log(n)}{n} \to 0$. By formula (2.32), we get

$$\begin{aligned}\liminf_{n\to\infty} \frac{\log(n)}{n} \log\left(\frac{1}{\Delta_n}\right) &\geq \liminf_{n\to\infty}\left[\frac{2}{d}\frac{\log(n)}{n}\log\left(\frac{1}{v_n}\right)\right] \\ &\geq \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}. \end{aligned}$$

□

We tested the convergence rate bound claimed in this section. By Theorem 2, the normalized error is bounded as:

$$\liminf_{n\to\infty} \frac{\log(n)}{n} \log\left(\frac{1}{\Delta_n}\right) \geq \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}$$

where $\gamma = \frac{d(2\pi)^{d/2}}{2\Gamma(1+d/2)}(\det\nabla^2 f(t^*))^{-1/2}$.

We calculated the theoretical bound with $d = 2$ by Matlab, which gave $\frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2} = 0.6721$. By running the rectangle algorithm for $1,000$ iterations, the value $\frac{\log n}{n}\log(\frac{1}{\Delta_n})$ is shown in Figure 2.2.

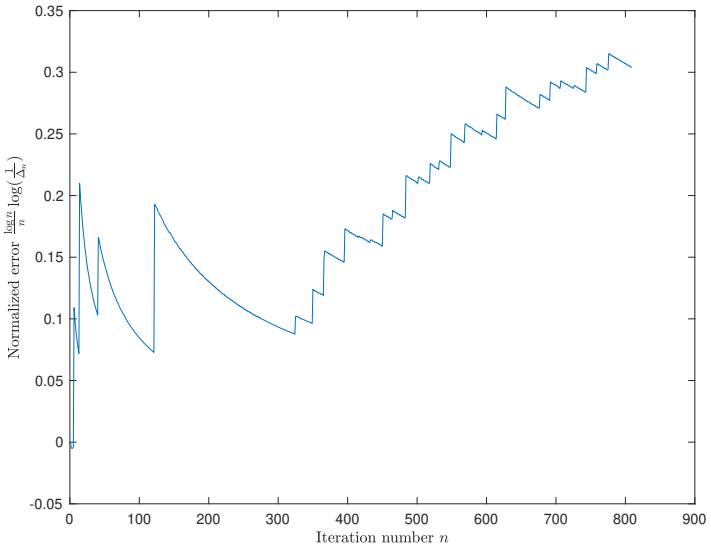

Figure 2.2 Normalized error for rectangle algorithm.

2.3.2 Discussions and comparison with DIRECT algorithm

In this subsection, we compare the convergence rates of the proposed algorithm and other alternatives which can be used to solve global optimization problems.

There is an asymptotic analyse of deterministic optimistic optimization algorithm provided in [53]. We are not aware of other asymptotic analyses of similar global optimization algorithms. In the paper [53], they provide asymptotic analyse of a simultaneous optimistic optimization approach which can be considered as a generalization of the DIRECT algorithm. In their studies, their simultaneous optimistic optimization approach has errors of order $\exp(-c\sqrt{n})$ after n function evaluations. We can see from Figure 2.3 that the DIRECT algorithm appears to

have normalized error $\frac{1}{\sqrt{n}} \log\left(\frac{1}{\Delta_n}\right) \to c_1$ as $n \to \infty$, where c_1 is a positive constant number. This gives experimental evidence that the DIRECT algorithm also has error of order $\exp(-c_1\sqrt{n})$. As we mentioned before, the rectangle algorithm has error of order $\exp(-c_2 \frac{n}{\log(n)})$ as $n \to \infty$ where $c_2 = \frac{3^{d-2}}{2\gamma}\left(\frac{\lambda_1}{2+3d\alpha}\right)^{d/2}$.

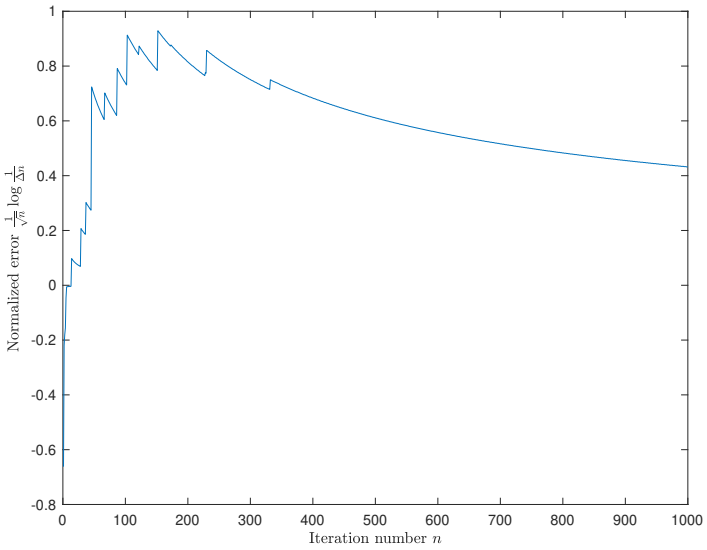

Figure 2.3 Normalized error of DIRECT algorithm.

2.4 Experiments

We tested our optimization algorithm on quadratic functions, Rastrigin functions, Branin functions and GKLS functions in Matlab and compared the algorithm we developed with the DIRECT algorithm on GKLS functions.

Quadratic function We use the quadratic function $f(x) = \sum_{i=1}^{d}(x_i - 0.5)^2$ to test the rectangle algorithm. The algorithm produced the following results shown in Figure 2.4.

Rastrigin function The Rastrigin function is a non-convex function used as a performance test problem for global optimization algorithms. It is a typical example

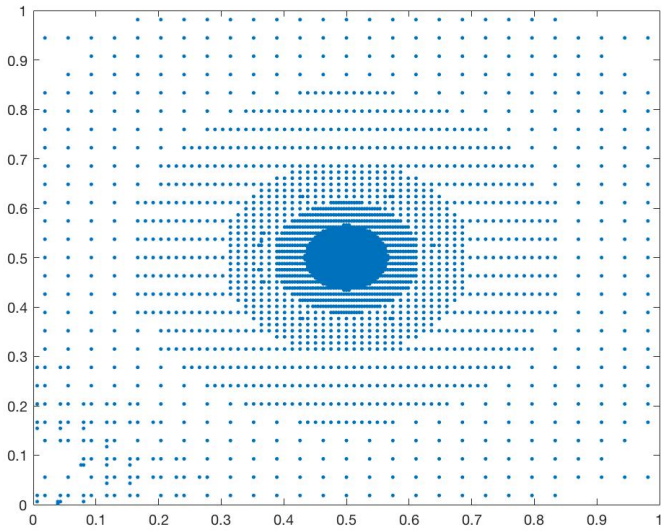

Figure 2.4 Evaluation points for quadratic 2-D objective function.

of a non-linear multimodal function. It was first proposed by Rastrigin [64] as a 2-D function and has been generalized by Mühlenbein et al [52]. Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima.

On a d-dimensional domain it is defined by:

$$f(\mathbf{x}) = Ad + \sum_{i=1}^{d} \left[x_i^2 - A\cos(2\pi x_i)\right] \tag{2.33}$$

where $A = 10$ and $x_i \in [-5.12, 5.12]$. It has a global minimum at $\mathbf{x} = 0$ where $f(\mathbf{x}) = 0$.

We use the Rastrigin function with the domain rescaled to $[0,1]^d$ which normalized $\bar{x}_i$ as $\bar{x}_i = (x_i + 5.12)/10.24$ to test the rectangle algorithm. This makes $\bar{x}_i \in [0,1]$ which is consistent with the algorithm as presented. For the 2-D Rastrigin function, the algorithm produced the results shown in Figure 2.5.

Branin function The Branin function [13] is another good test function for global optimization algorithms because there are three global minima. The Branin function

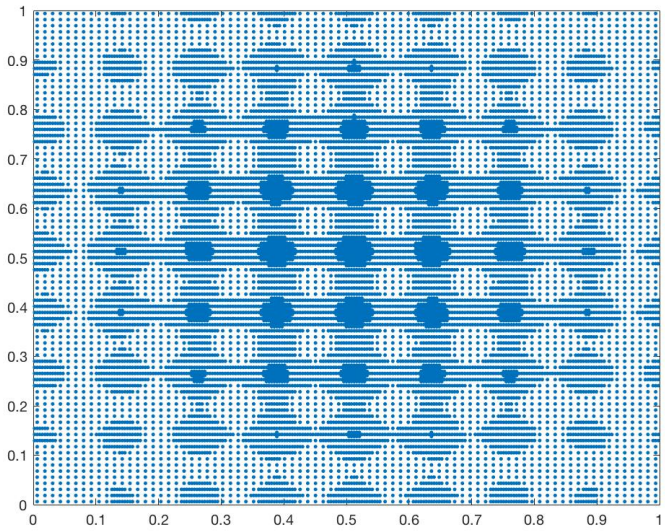

Figure 2.5 Evaluation points of Rastrigin 2-D function.

is defined by $f(x) = a(x_2 - b \cdot x_1^2 + c \cdot x_1 - r)^2 + s(1-t)\cos(x_1) + s$. The recommended values of a, b, c, r, s and t are: $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$. This function is usually evaluated on the square $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$.

We define the Branin function with $\bar{x_1} = (x_1 + 5)/15$ and $\bar{x_2} = x_2/15$ to test the rectangle algorithm. The algorithm evaluated at the points shown in Figure 2.6.

GKLS To further test the algorithm, we use the GKLS generator described in [46] to generate other test cases. This allows the generation of smooth objective functions, and so we use the version of the algorithm without the logarithmic term in the numerator in the definition of ρ_i^n.

For our better understanding of GKLS objective functions, we plot a differentiable function of dimension $d = 2$ in Figure 2.7 which has several different local minima and a unique global minimum value of -1.

In order to obtain comparable results, the six GKLS test classes of continuously differentiable functions of dimensions $d = 2$, 3, and 4 defined by the similar five

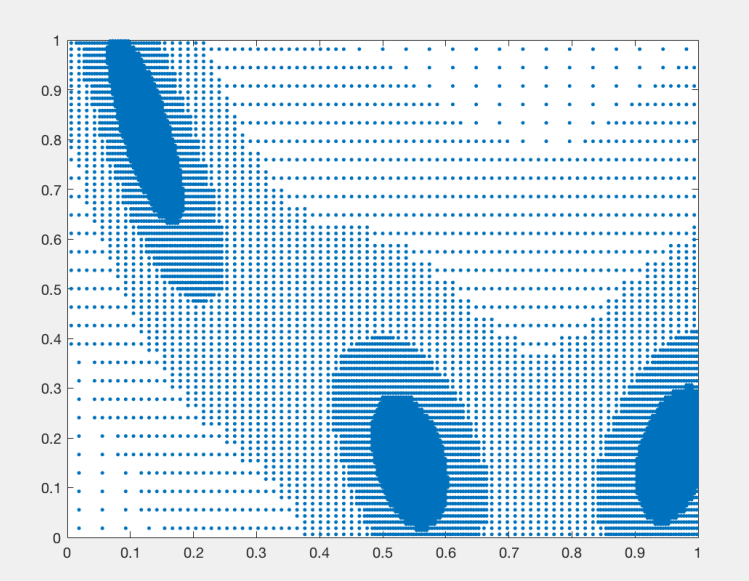

Figure 2.6 Branin 2-D objective function.

parameters as in [70, 71] were used (see Table 1). The GKLS generator produces different classes of test functions with known local and global minima. We set the number of local minima m equal to 10. The global minimum value f^* was set equal to -1. We used the fixed function evaluation numbers $N = 51$ to track the error of the DIRECT algorithm and the rectangle algorithm. In the following tables, D is the distance parameter and R is the radius parameter required for generating objective functions in GKLS. ADE is the average error for the DIRECT algorithm and ARE is the average error for the rectangle algorithm. The error was calculated as the minimum value obtained by the algorithm minus the known global minimum value which is defined by the GKLS generator.

Results of numerical experiments with six GKLS tests classes appear in Table 2.1 and Table 2.2. For these experiments the average error of the rectangle algorithm is smaller than the average error of the DIRECT algorithm.

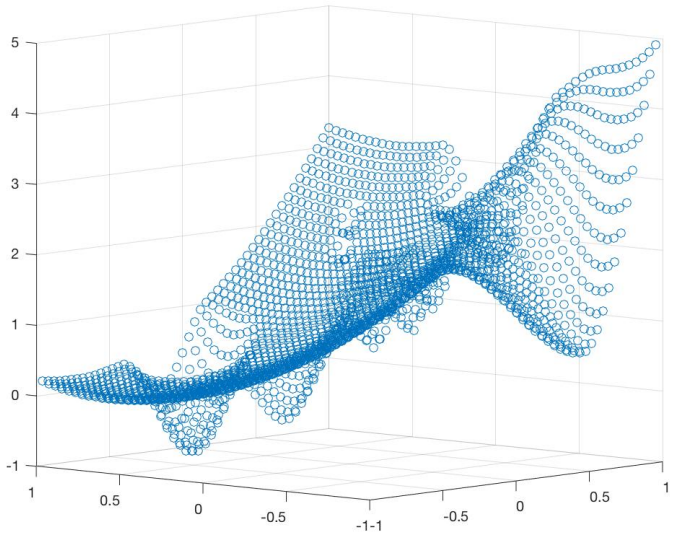

Figure 2.7 GKLS 2-D objective function.

Table 2.1 Description of the GKLS Test Classes Used in Numerical Experiments

Class	d	m	f^*	D	R
1	2	10	-1.0	0.66	0.33
2	2	10	-1.0	0.66	0.2
3	3	10	-1.0	0.66	0.33
4	3	10	-1.0	0.66	0.2
5	4	10	-1.0	0.66	0.33
6	4	10	-1.0	0.66	0.2

Table 2.2 Error of Two Algorithms for 600 GKLS Test Functions

Class	d	f^*	D	R	N	ADE	ARE
1	2	-1.0	0.66	0.33	51	0.1114	0.1101
2	2	-1.0	0.66	0.2	51	0.417	0.3256
3	3	-1.0	0.66	0.33	51	0.6278	0.6240
4	3	-1.0	0.66	0.2	51	0.8133	0.8048
5	4	-1.0	0.66	0.33	51	0.8827	0.8366
6	4	-1.0	0.66	0.2	51	0.8952	0.8416

CHAPTER 3

A GLOBAL OPTIMIZATION ALGORITHM FOR IMAGE REGISTRATION AND CLUSTERING

3.1 Image Registration

How to segment and how to register images are important tasks in image processing. Several algorithms [82, 49, 75] are used in those areas. Image registration aligns two or more images of the same scene taken at different times, from various sensors and/or from multiple views [92]. Obtaining the change of the original image and referenced image over time or from different views play a significant role in solving many important problems. For instance, comparing a patient's magnetic resonance image (MRI) with the MRI which contains a kind of cancer can help determine if the patient has the cancer or not. There are many useful applications of image registration, for example in medical imaging, geometrical sensor image comparison and map updating. The benefits gained from image registration and the rapid development of image acquisition devices motivate us to explore automatic image registration. We give a brief introduction to the main steps of image registration and summarize the various methods to solve image registration problems. We present a modified global optimization algorithm for image registration.

3.1.1 Background

The general steps of image registration illustrate how image registration works [18, 75, 49]. The first step is feature detection which won't happen in each method of image registration. Feature detection mainly includes point detection and extraction: identify and extract the points that carry critical information about the scene structure through a number of control points [18]. The second step is feature matching which includes image description, similarity measure and point selection.

The third step is transform model estimation. Point pattern matching establishes correspondence between the selected points to determine the correct and incorrect correspondences. And the last step is image resampling and transformation. There are typical problems in each registration step.

There are multiple methods to register images. We can divide all these methods into two classes based on image matching approaches. One of them is area-based methods which automatically compare two or more images pixel by pixel. One branch of these methods is based on minimizing the similarity difference. The similarity difference of image registration tells us how much these images match. The transformation which obtains the minimal similarity difference is the goal. The second class is feature-based methods which mainly extract obvious features from the image and then map the images based on the features. In feature-based algorithms, we take advantage of identifiable landmarks to aid the alignment [16].

The methods we presented in this section belongs to area-based method. Area-based methods, also called correlation methods or template matching methods, usually merge the feature detection part with the matching part [32]. They match images with detected static or salient objects such as a house or road in the image. They use a rectangular window area to estimate the correspondence of the images. The following are several frequently used matching methods.

One of the first articles proposing mutual information (MI) based image registration is Viola and Wells [81]. The paper registered MR-CT and MR-PET images of a human brain and optimized the MI using Brent's method and the Powell's multi-dimensional algorithm [48]. There are comparisons of different matching methods in [66].

The image registration problem can be viewed as the problem of maximizing the similarity or minimizing the dissimilarity measure [75]. Exhaustive search over the entire image is a basic and easy but inefficient way to optimize similarity. The

Gauss-Newton numerical minimization algorithm for minimizing the sum of squared differences was used with projective geometric deformation in [74]. Maximization of mutual information using gradient descent is used in [81]. Levenberg-Marquardt optimization was used to minimize the variance in intensities of corresponding pixels in [68]. Combination Levenbery-Marquardt method and the sum of squared differences of the metric is described in [59]. In [76], a dissimilarity measure defined on point pairs was minimized by means of simulated annealing.

Correlation-based methods compute a similarity measure by pixel comparisons. A region from one image can be translated or rotated aroud the second one to get the best alignment by optimizing the simiarity measure. The main simiarity measure we use is the sum of squared pixel differences.

The use of mutual information as a similarity measure is surveyed in [62]. We still have efficiency, accuracy and robustness problems when applying mutual information in image registration because the optimization algorithm we are choosing cannot exactly get the global optimizer. Many researchers attempt to improve mutual information with other information-theoretic measures, for example, see in [16].

We take as input two matrices of pixel intensities, possibly of different sizes. After interpolating the pixel values, we can view the fixed and moving images as mappings $I_f : \Omega_f \to \mathbb{R}$ and $I_m : \Omega_m \to \mathbb{R}$, respectively, where Ω_f and Ω_m are suitable rectangles in $\mathbb{R}^2$. (For simplicity we assume gray-scale images.) We seek a transformation $T : \Omega_f \to \Omega_m$ such that $I_m \circ T$ is "close to" I_f. To quantify the notion of closeness we adopt a dissimilarity measure C and choose T to minimize $C(T; I_f, I_m)$. To obtain a tractable optimization problem, we restrict the class of transformations. We consider rigid transformations involving translation and rotation of the moving image.

By suitable scaling, we consider the registration problem to be a global optimization problem over a parameter space $[0, 1]^d$, where we take $d = 2$ (two

translation parameters) or $d = 3$ (two translation parameters and one rotation parameter). We are interested in methods that are automatic in the sense that no starting point or other parameters are required from the user.

Local optimization algorithms are often used in image registration. In [31] the authors examined the application of global optimization approaches, showing that local optimization schemes often did not find the optimum in their test medical imagery. Also in large-deformation image registration, there is the issue of large-deformation causing methods to be trapped at local minima. In [84], the authors propose a structural Tensor and Driving force-based Log-Demons algorithm for avoiding it and also speed up the registration process. Global optimization algorithms such as simulated annealing or genetic algorithms are commonly-used methods for image registration [63]. There are other derivative-free global optimization algorithms that treat the objective function as a black box [23, 43]. To improve efficiency, some researchers also developed global optimization algorithms that work with Lipschitz constants of gradients [38, 71]. Some proposed partition-based deterministic algorithms for global optimization of Lipschitz-continuous functions work without requiring knowledge of the Lipschitz constant [33, 42, 44]. Advanced deterministic global optimization algorithms [21, 60, 72, 73] are potential tools for image registration.

Other approaches to image registration include soft computing-based methods such as artificial neural networks, fuzzy sets and optimization heuristics [54]. Some recent work explores the use of deep learning frameworks for image registration [11].

We do not address the problems of the choice of similarity measure or permissible transformations. Rather, we focus on the problem of optimizing the similarity measure over the chosen set of transformations. We specifically address the issue of *automatic* image registration, where no special domain knowledge is required, for example to identify landmarks, and no tuning of input parameters is required.

3.1.2 The algorithm

The algorithm subdivides the feasible region into subrectangles and sequentially chooses the next subrectangle to subdivide according to a certain numerical value. We use the rectangular subdivision introduced in [33]. The criteria for selecting the next subrectangle to subdivide is similar to the criteria used in [5].

The algorithm operates by decomposing $[0,1]^d$ into hyper-rectangles as follows. The first operation is to evaluate f at the center of the unit hyper-cube $(1/2, 1/2, \ldots, 1/2)$. Given a current decomposition, choose one of the hyper-rectangles (according to the maximal value of a criterion to be defined below) and trisect it along the longest axis. The central sub-hyperrectangle retains its central function value, while the function is evaluated at the centers of the two other hyperrectangles.

After k iterations of the algorithm, f will have been evaluated $2k+1$ times and the unit hyperrectangle will have been decomposed into $2k+1$ sub-hyperrectangles.

In image registration problem, we modified the algorithm in Chapter 2 for approximating a global optimal point. Use the following ρ and $g_n(x)$.

$$\rho_i^n \equiv \frac{|R_i|^{r/d} * \log(1 + \frac{f(c_i) - M_n}{g(v_n)})^{r/d}}{(f(c_i) - M_n + g(v_n))}, \tag{3.1}$$

where c_i is the center of R_i and r is the smooth rate of the objective function. We define $g_n(x)$ as

$$g_n(x) = d(x \log n)^{r/d}.$$

The algorithm with the logarithmic factor in the numerator causes the evaluation points to concentrate at a slower rate. The reason for adding this factor is uncertainty in the smoothness of f.

The processing of the algorithm is the same with the algorithm in Chapter 2. The only stopping rule that we consider is to initially specify a fixed number

of function evaluations after which the algorithm terminates. The input to our problem is discrete, but by interpolating the pixel intensities we obtain a continuous optimization problem.

3.1.3 Local search

The algorithm described in Section 2.2 uses only function values and no local information, such as derivatives. After the algorithm terminates, it is natural to use a local optimization method to improve on the terminal value. The reason we use a local optimization method is that the algorithm described in Section 2.2 approaches the optimal value in the limit, but the solution will typically be sub-optimal after a finite number of iterations. To improve the final solution, we will stop the algorithm with a reasonable N and start to use a local optimization method such as gradient descent to improve the results. We continue to assume that only function values are available, and so finite-difference gradient approximations are used in the local search methods. We used the following differences for our local descent method:

$$G(x,y,r) \equiv$$
$$\left(\frac{f(x+\epsilon_x,y,r)-f(x,y,r)}{\epsilon_x}, \frac{f(x,y+\epsilon_y,r)-f(x,y,r)}{\epsilon_y}, \frac{f(x,y,r+\epsilon_r)-f(x,y,r)}{\epsilon_r} \right),$$

where ϵ_x, ϵ_y, and ϵ_r are chosen to be comparable to the distance between pixels.

3.1.4 Numerical experiments

We implemented the optimization algorithm for the 2-D and 3-D image registration problems in Matlab. The 2-D image registration problem is to find the best translation (left to right and up-down), while for the 3-D registration problem we consider rotation in addition to translation of the image.

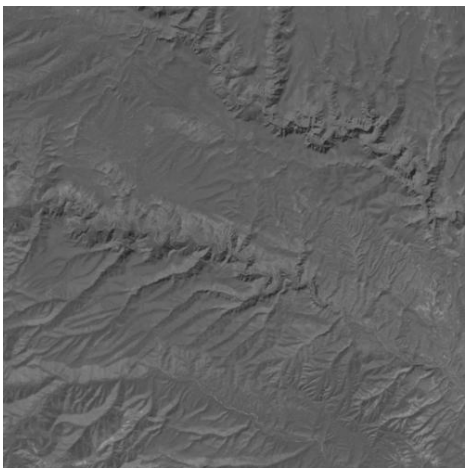

Figure 3.1 Desert image.

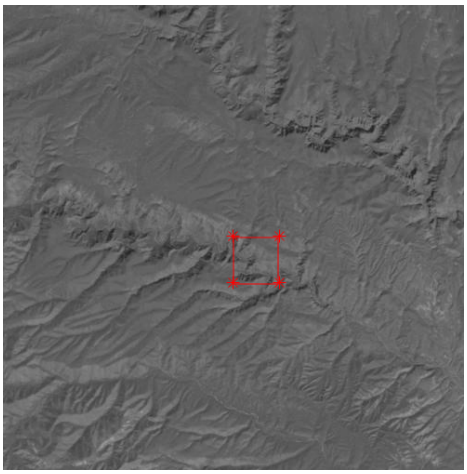

Figure 3.2 Target subimage (enclosed in red rectangle).

In the 2-D image registration problem, we take the original image to have $s_m \times s_n$ pixels. We use the example desert image shown in Figure 3.1, which has 500×500 pixels. The target size is $t_m \times t_n = 50 \times 50$ and the optimal location is $(250, 250)$.

The objective function we used is the sum of squared image pixel differences. Suppose the pixel value of the target image at location (i, j) is $p_1(i, j)$, and the pixel value of the original image at the same place is $p_2(i, j)$. Let $f(x, y)$ be the sum of squared image pixel difference if x is the translation along the x-axis and y is the translation of the y-axis. The objective function of 2-D image registration is then

$$f(x, y) = \sum_{i=1}^{i=t_m} \sum_{j=1}^{j=t_n} (p_1(i, j) - p_2(i + x, j + y))^2 \tag{3.2}$$

where $1 \leq x \leq s_m - t_m$ and $1 \leq y \leq s_n - t_n$. The objective function is shown in Figure 3.3.

We plot all points chosen by the algorithm described in Section 5.3 with local search in Figure 3.4. The last position we got after 4,000 iterations is exactly the optimal position with minimum function value 0.

In the 3-D image registration problem, the size of original image is 600×600 which is the dog picture in Figure 3.5.

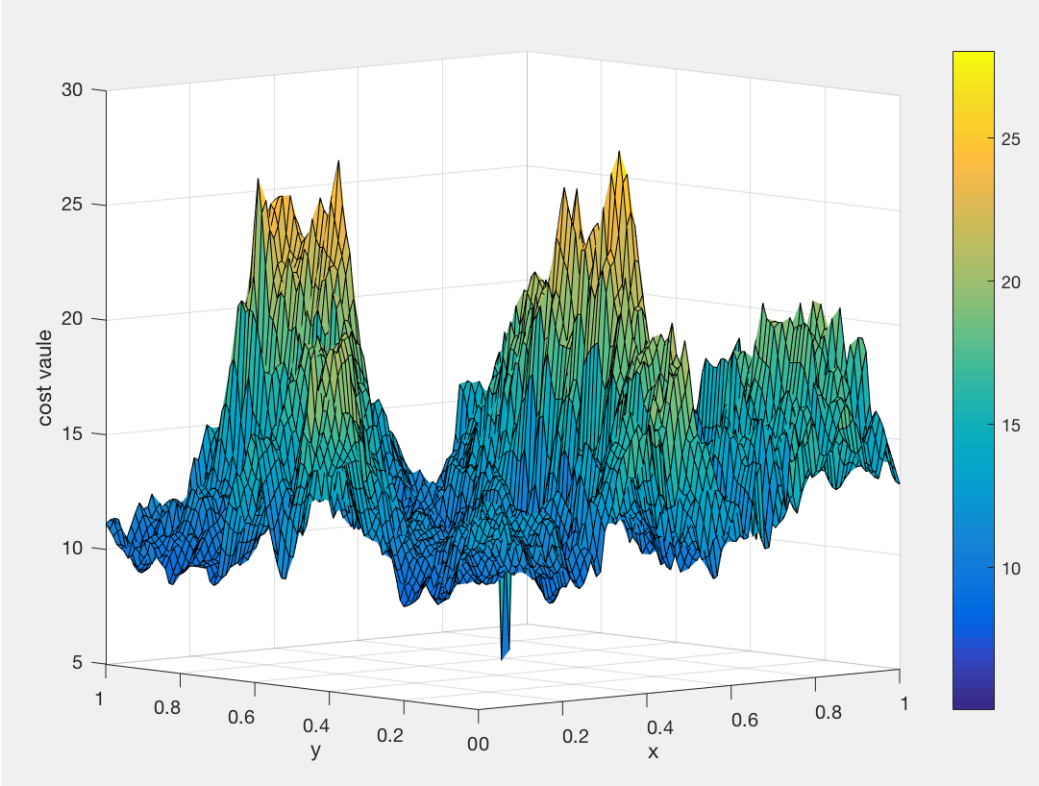

Figure 3.3 2-D image objective function.

The objective function of three-dimensional image registration is as follow.

$$f(x,y,r) = \sum_{i=1}^{i=t_m} \sum_{j=1}^{j=t_n} (p_1(i,j) - T(p_2(i+x, j+y), r))^2 \tag{3.3}$$

where $1 \leq x \leq s_m - t_m$,$1 \leq y \leq s_n - t_n$ and $0 \leq r \leq 180$.

The 2-D image registration problem is to find the best translation (left to right and up-down), while for the 3-D registration problem we consider rotation in addition to translation of the image. The objective function we used is sum of squared image pixel intensity differences. Figure 3.7 shows the cost surface for a 2-D section of the 3-D cost function (varying x and y translations with rotation fixed) for the example image used in our experiment.

The target size is 50×50 which is the red rectangle part of Figure 3.6 and the optimized location is (299,299,90). In the experiment, we can get the optimized center

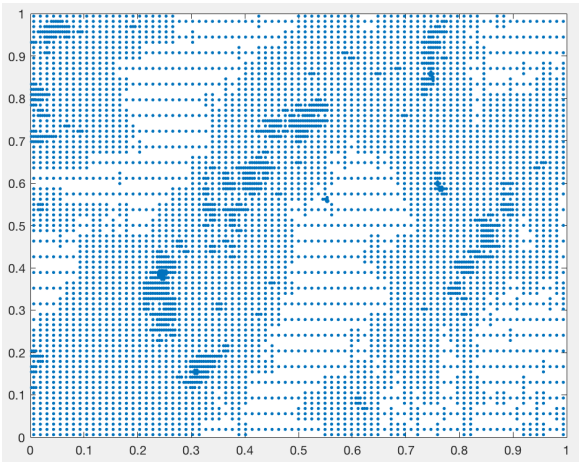

Figure 3.4 Scatter plots for image registration 2-d cost function after 2,876 iterations.

point (294.6,301.6,90) after 4,000 iterations. The piece we got by the algorithm is the right part of Figure 3.8. The figure exactly shows the target sub image and the piece that our algorithm found. The optimized function value that our algorithm got is 3.8936. The algorithm we designed cannot give us the optimized function value which is 0. So we can consider the local search using the last center point. The local search algorithm we used is gradient descent algorithm. From Figure 3.9, we see that the gradient descent algorithm really works. It gives us optimized function value 1.0492 which is much better than without gradient descent algorithm.

3.1.5 Comparisons with other algorithms

There are many alternative global optimization algorithms that might be considered for image registration, such as genetic algorithms [67], particle swarm optimization [77], and simulated annealing [88].

We compared our global optimization algorithm with the following alternatives: 1) An exhaustive search (searching over a fixed equi-spaced grid of points), followed by local descent from the best point; 2) The Matlab built-in simulated annealing algorithm [88], followed by local descent from the best point; 3) The Matlab built-

Figure 3.5 Dog image.

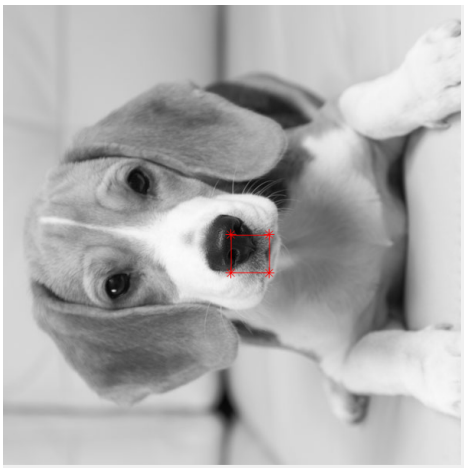

Figure 3.6 Target subimage (enclosed in red rectangle).

in genetic algorithm, followed by local descent from the best point; 4) DIRECT algorithm [19, 33], followed by local descent from the best point.

We performed 4,000 iterations, followed by local descent, for all algorithms because that resulted in a reasonable visual match of the registered image. In Figure 3.10, the x-axis indicates the smallest cost value obtained by the algorithm and the y-axis indicates the proportion of runs for which the algorithm achieved the cost value.

Simulated annealing and genetic algorithms are randomized algorithms with considerable variability in the minimal cost obtained. To get a sense of the range of solutions we ran the simulated annealing algorithm 100 times independently and plotted the empirical cumulative distribution function of the results as the black line in Figure 3.10. We also ran the genetic algorithm 100 times independently and plotted the empirical cumulative distribution function of the results as the red line in Figure 3.10. The other two algorithms are deterministic. For the algorithm we proposed, we randomized it by applying a random perturbation to the starting point. For each of 100 independent replications we started with the center translated by a uniformly distributed offset in $[-0.05, 0.05]^3$. For exhaustive search,we used a three-dimensional

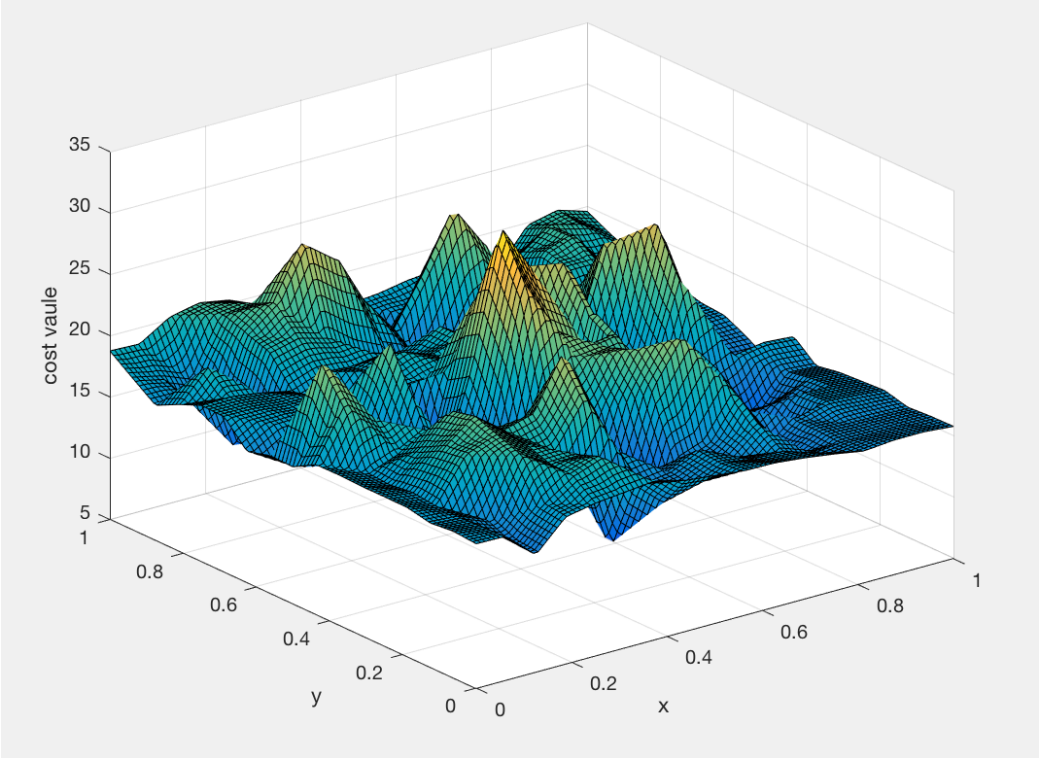

Figure 3.7 Cost function with fixed rotation.

grid of $16^3 = 4{,}096$ points, and we randomly perturbed the grid by a uniformly distributed offset in $[-0.03125, 0.03125]^3$ for each of 100 independent replications. (The exhaustive search performed better with a smaller random offset than we used for the rectangle algorithm.)

Our algorithm, named "rectangle+grad" in Figure 3.10, obtained the smallest average cost value of 1.04; the range is shown by the blue line in Figure 3.10. The exhaustive search algorithm, which obtained a smallest cost value around 2.84, performed as indicated by the green line. Simulated annealing, with an average cost over 3, performed as shown by the black line in Figure 3.10. The genetic algorithm performed worse than simulated annealing as shown in Figure 3.10. Because we repeated simulated annealing 100 times independently (with 4,000 iterations on each replication), there were several runs on which simulated annealing worked really well but on average the simulated annealing algorithm obtained worse results than the

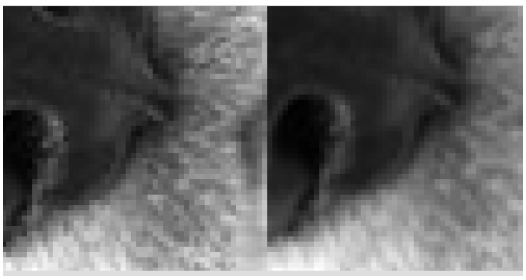

Figure 3.8 Targeted sub image and the sub-image obtained by the algorithm.

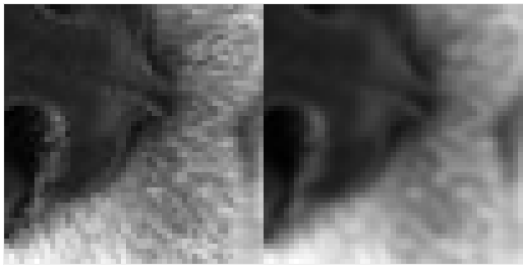

Figure 3.9 After local search with 4,000 iterations.

proposed algorithm. The DIRECT algorithm followed by local descent obtained a smallest average cost value around 2.47. It performed as shown by the magenta line in Figure 3.10, which is better than exhaustive search but still worse than the rectangle algorithm followed by local descent.

Using the same experiment setup as used for the comparison in Figure 3.10, we compared the rectangle and DIRECT algorithms based on 100 independent replications of 4,000 iterations on each; the result is shown in Figure 3.11. The red line in Figure 3.11 represent the rectangle algorithm and the blue line represents the DIRECT algorithm. The average cost value of the rectangle algorithm is 2.43 and the average value of the DIRECT algorithm is almost the same at 2.49.

3.2 Clustering

3.2.1 Background

Clustering is a technique for classifying data points into different groups by their similarities [14, 93, 80]. It is commonly used for statistical data analysis such

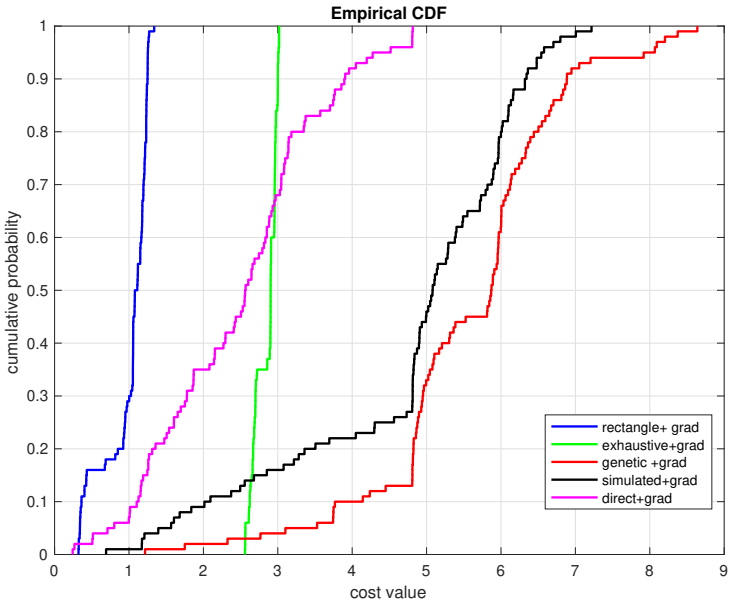

Figure 3.10 Distribution of cost values obtained by five algorithms after 4,000 iterations.

as social media data [45] but also used for machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression and computer graphics [30]. Clustering techniques are important for better visualizing large data sets.

There are many ways to determine what constitutes a cluster and many algorithms for efficiently grouping the data. The distances between the data or dense areas of the data space can give us some hints about the data itself. Clustering can be formulated as a multi-objective optimization problem because there are many different metric functions for clustering.

Cluster analysis originated in 1932 by Driver and Kroeber [14] and further studied by Zubin in 1938 [93] and Robert Tryon in 1939 [80]. It was famously used by Cattell beginning in 1943 [7] for trait theory classification in personality psychology.

There are many different clustering algorithms which solving problems based on different data quality [14, 80, 7, 30, 86]. We can divide clustering algorithms

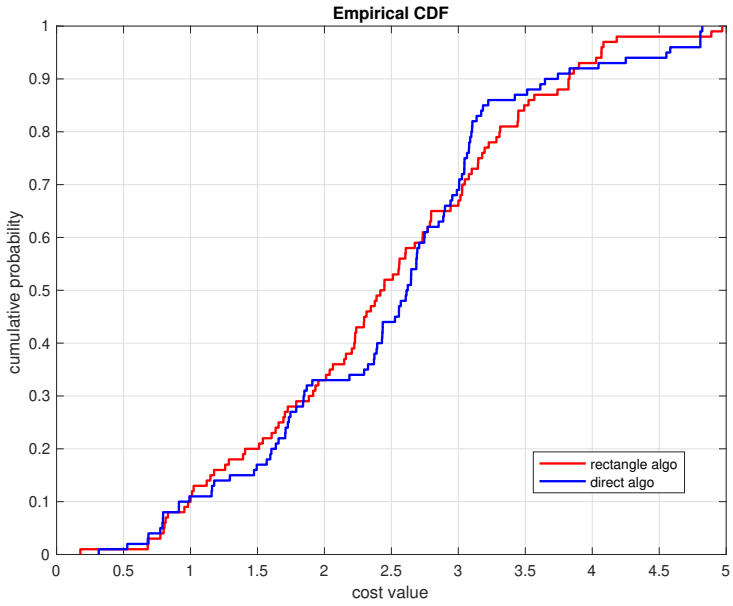

Figure 3.11 Distribution of cost values obtained by rectangle and DIRECT algorithms after 4,000 iterations.

into different groups based on the clustering model used. The approaches include hierarchical clustering, centroid-based clustering, distribution-based clustering, and density-based clustering. Hierarchical clustering starts with each data point assigned to its own cluster. Then two nearest clusters are merged. The algorithm continues in this way, reducing the number of clusters by one with each iteration. The centroid-based approach is to choose a number of centroids and then assign each data point to the nearest centroid. The question of where to locate the centroids is an optimization problem.

Most clustering formulations involve minimizing a cost based on a distance function. A popular approach is that of the k-means algorithm and its variants. k-means is an iterative clustering algorithm which finds local minima of the sum of squared Euclidean distances of the points to the nearest centroid. A good starting point is important for the k-means algorithm for avoiding bad local minima. The k-means++ algorithm uses a probabilistic method to find a good starting point for k-

means [1]. It improves the chance that k-means obtains the global minimum (though it can still terminate at a suboptimal local minimum).

Typical clustering approaches, including k-means, converge to a local minimum of the cost function that may be larger than the global minimum. The main motivation for our work is to design algorithms that converge to the global minimum cost. We describe such an optimization algorithm. The algorithm works for a general class of distance measures (including, but not restricted to, the distance measures for which k-means converges to a local minimum).

Other researchers have proposed methods with similar motivation to ours [87, 69, 50]. The reference [87] summarises several different methods which combine genetic algorithms with k-means to reduce the chance of getting stuck at a local minimum. The k-medoids algorithm was proposed to work with more general distance measures than k-means. In [69, 50], the authors combine simulated annealing with k-means to improve the chance of achieving a global minimum.

3.2.2 k-means algorithm and k-means++

One of the most popular algorithms for data clustering is k-means which is efficient and easy to use. It is an iterative clustering algorithm which aims to find local minima. There are different distance metrics such as Euclidean distance, squared Euclidean distance or Manhattan distance which may used in clustering algorithms. Suppose we use squared Euclidean distance to formulate the clustering problem. Assume we want to divide the n point set $X_n = \{x^1, x^2, \cdots, x^n\}$ into k groups ($1 \leq k < n$) where x^i is a d-dimensional vector $(x_1^i, x_2^i \cdots, x_d^i)$ to minimize dissimilarity metric. Therefore it will become a $d \cdot k$ dimensional problem for the rectangle algorithm. The cost function is in equation (3.4) if we choose a centroid-method model:

$$cost(c) = \sum_{i=1}^{k} \sum_{x \in X_n^i} dist(x, c^i), \tag{3.4}$$

where $dist(x, c^i)$ is the squared Euclidean distance between the center points and points nearest the center point. Let X_n^i denote ith cluster which includes all points closest to center point c^i. It means minimizing the following formula.

$$dist(x, c^i) = \sum_{j=1}^{d} (x_j - c_j^i)^2, \tag{3.5}$$

Choose k center points $\{c^1, c^2, \cdots, c^k\} \in [0, 1]^d$ for the clusters to minimize $cost(c)$ which is defined above. For the d-dimensional case, the ith center point is $c_i = (c_1^i, c_2^i, \cdots, c_j^i, \cdots, c_d^i)$. The process which k-means algorithm works as follows.

1. Randomly choose an initial k centers $C = \{c^1, c^2, \cdots, c^k\}$
2. For each $i \in \{1, 2, \cdots, k\}$, set the cluster C^i to points $\in X_n$ which is closest to the cluster center c^i.
3. For each $i \in \{1, 2, \cdots, k\}$, set the center c^i to be the center of the all points in C^i, that is $C^i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
4. Repeat step 2 and 3 until there is no change in C.

The algorithm converges fast but sometimes it converges to a local minimum which is not the global minimum. For getting a more reasonable output, people start to think about finding a good start point for k-means which was the motivation for k-means++. In k-means++ algorithm [1], there is a set up process for choosing the initial k centers.

1. Choose a center point c_1 uniformly at random from X_n.
2. For each data point $x \in X_n$, calculate the shortest distance D from the point to the closest center already chosen.
3. Choose a new center point c_i with probability which is proportional to D.
4. Repeat steps 2 and 3 until there are k cluster center points.

5. Do a standard k-means process.

3.2.3 Numerical experiments for clustering

In the clustering problem, we use the rectangle algorithm which is proposed in Section 2.2 followed by the local optimization. We compare the method with the k-means++ algorithm. The cost function of clustering depends on the distance measure adopted. We use squared Euclidean distance to formulate the objective cost function of the clustering problem.

We generated a random data set with 100 2-D points distributed as shown in Figure 3.12. We divided the 100 points into 5 clusters using k-means++ and also the rectangle algorithm followed by the local optimization step. The results from the Matlab implementation of k-means++ are shown in Figure 3.13.

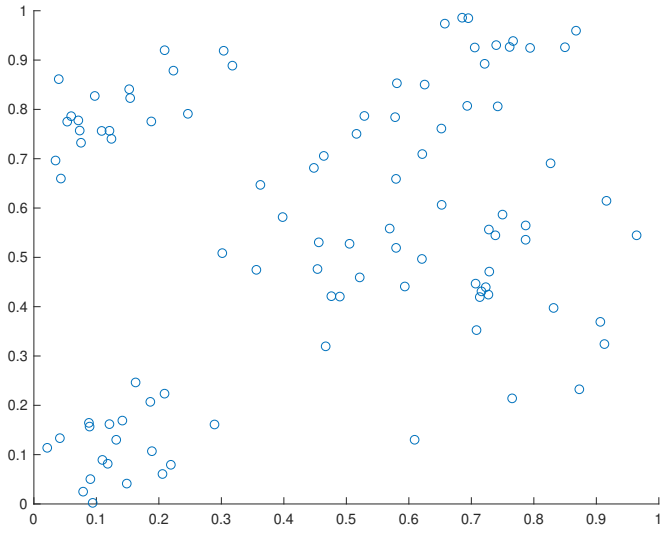

Figure 3.12 Distribution of 100 points data set.

The different colors show the original groups of the points. The center points found by the k-means++ algorithm are shown in Figure 3.13. The k-means++ algorithm obtained a suboptimal solution. We tried the k-means algorithm with more independent replications which resulted in different minima values with different

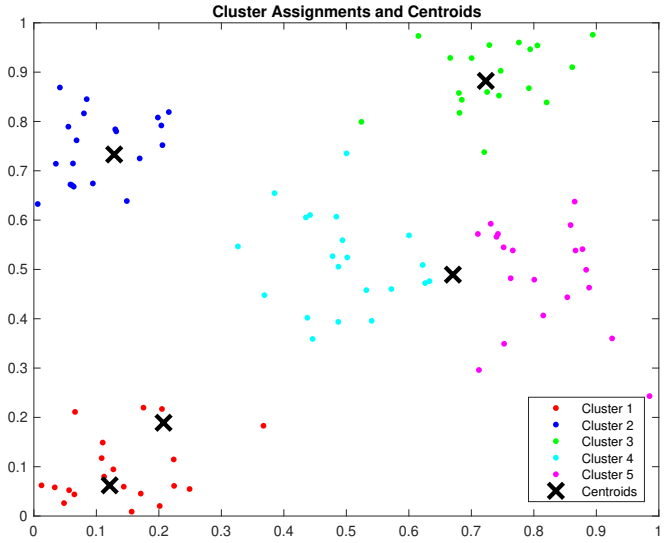

Figure 3.13 Centroids returned by k-means++.

replications, illustrating the fact that k-means is a local optimization algorithm. As with many other local optimization algorithms, a critical task is to choose a good starting point. The result obtained by the rectangle algorithm followed by one-step local search is shown in Figure 3.14. The minimum value is smaller than the minimum value that k-means obtained.

Comparisons with other algorithms In this section we compare the proposed algorithm with k-means++ and two global optimization algorithms using randomly generated test problems.

The probability model used to generate the data points in $[0,1]^m$ was as follows. We created P random $m \times m$ matrices, A^p, $1 \leq p \leq P$, with elements independent standard normal random variates. (Such a matrix is nonsingular with probability one.) We chose P means independently, uniformly over $[0,1]^m$, μ^p, $1 \leq p \leq P$. We then generated a random data point as follows:

1. Select a population p at random uniformly from $1, 2, \ldots, P$.

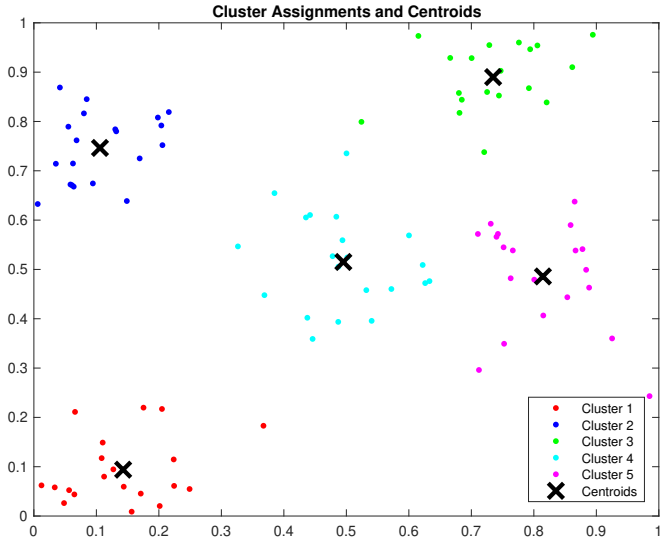

Figure 3.14 Centroids returned by rectangle algorithm followed by one-step local search.

2. Generate a trial point

$$X = \mu^p + 0.1 \cdot A^p Z,$$

where Z is a vector of independent standard normal random variates.

3. If X lies outside of $[0,1]^m$, then go back to step 2.
4. Stop when n points have been generated.

A sample 2-D point set is depicted in Figure 3.15.

We then repeated the following experiment 100 times.

1. Generate n random points in $[0,1]^m$.
2. Perform 10,000 replications of k-means, starting from independent uniformly distributed starting centroids, keeping track of the distinct solutions (local minima) obtained.
3. Report the average number of distinct local minima found.

For these experiments we chose the number of clusters $k = 3$, data dimension $m = 2$, and number of data points $n = 50$.

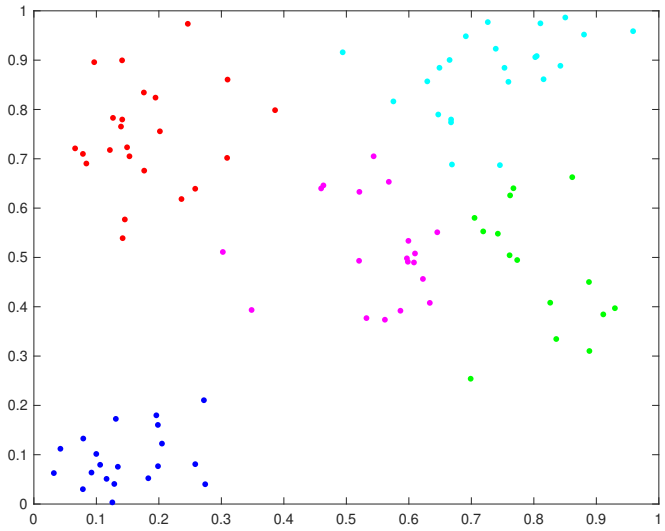

Figure 3.15 Five truncated normal populations.

Running the experiments with $m = 2$, k-means averaged 10.94 local minima (ranging from a minimum of 3 to a maximum of 21), while k-means++ averaged 10.76 local minima (ranging from a minimum of 3 to a maximum of 22).

In both cases, the number of local minima reported is a lower bound on the actual number of local minima.

A typical set of local minima for k-means was

$$6.6, 9.9, 10.3, 11.8, 12.1, 12.4, 12.5, 12.8, 13.3, 13.8, 16.6$$

and for k-means++ with the same data

$$6.6, 9.9, 10.3, 12.1, 12.4, 12.5, 13.3, 13.8.$$

The average difference with k-means was 0.30 (maximum 6.11), while the average difference with k-means++ was 0.14 (maximum 2.27). In all cases the rectangle algorithm obtained a value at least as small as the other algorithms.

Comparison of rectangle algorithm with k-means++ In this subsection we examine the percentage of time that the k-means++ algorithm obtained a worse result than the rectangle followed by one-step local search.

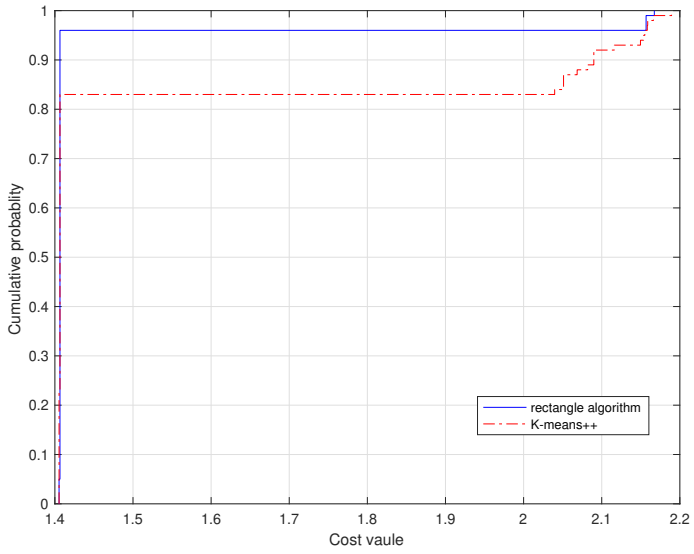

Figure 3.16 Comparison between the rectangle algorithm and k-means++ with same data set.

We did the experiments two different ways. First, we used one one data set with 100 points but ran this data set 100 times using two different methods. To obtain different results with the rectangle algorithm we randomized it by applying a small random perturbation with each run. With the same data set, the rectangle algorithm followed by one-step local search always obtained the same result or better compared with k-means++. In Figure 3.16, the blue line represents the rectangle algorithm and the red line represents k-means++. The rectangle algorithm produced a better result than k-means++ 21% of the time.

We also experimented with 100 different data sets which were generated as described above. In Figure 3.17, the percentage of time the rectangle algorithm performed better than k-means++ was 47%.

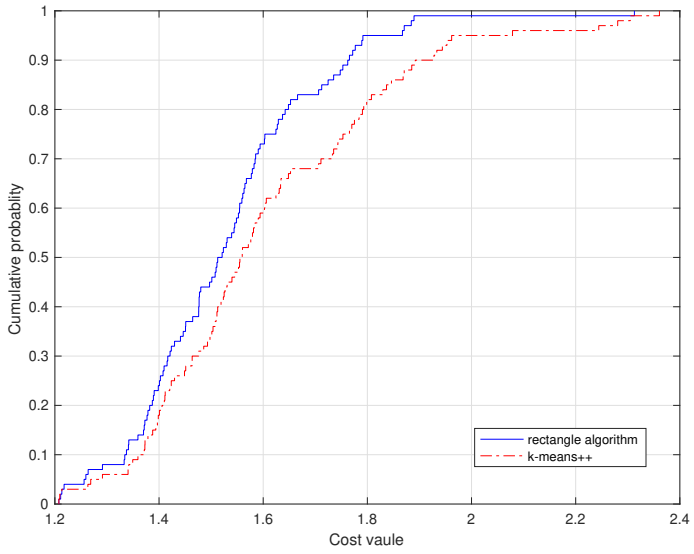

Figure 3.17 Comparison between the rectangle algorithm and k-means++ with 100 different data sets.

Comparison of k-means++ with simulated annealing and genetic algorithms

We also compared the algorithm with some other global optimization algorithms such as simulated annealing [88] and genetic algorithms [17]. In Figure 3.18, the colored lines represent the cost difference between k-means++ and different global optimization algorithms. The blue line is the cost difference between k-means with rectangle algorithm, showing that the rectangle algorithm obtained a smaller cost value than the other global optimization algorithms.

In Figures 3.18, 3.19, and 3.20 we compare the optimization algorithms for number of iterations ranging from 1,000 to 100,000.

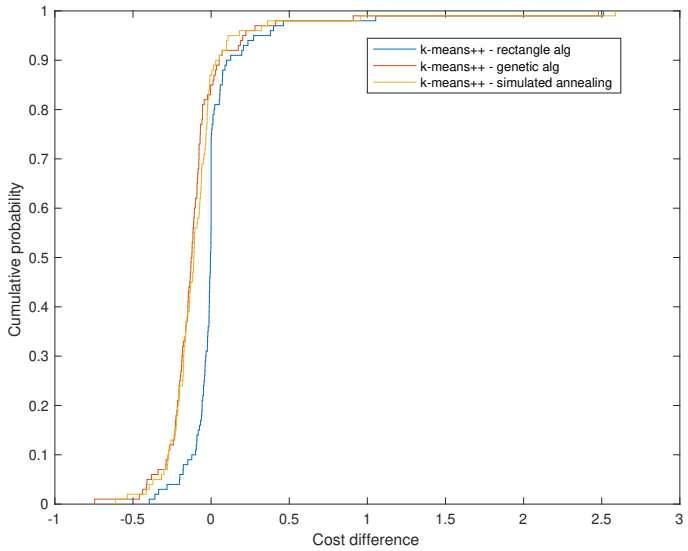

Figure 3.18 Comparison between the rectangle algorithm and other global optimization algorithms after 1,000 function evaluations.

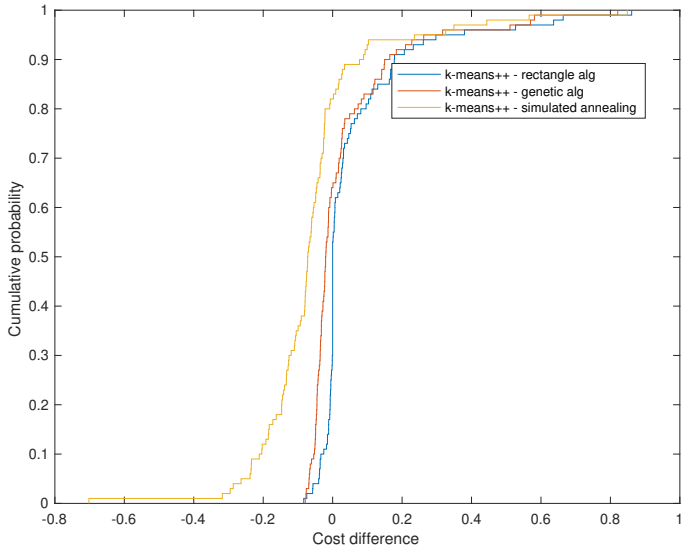

Figure 3.19 Comparison between the rectangle algorithm and other global optimization algorithms after 10,000 function evaluations.

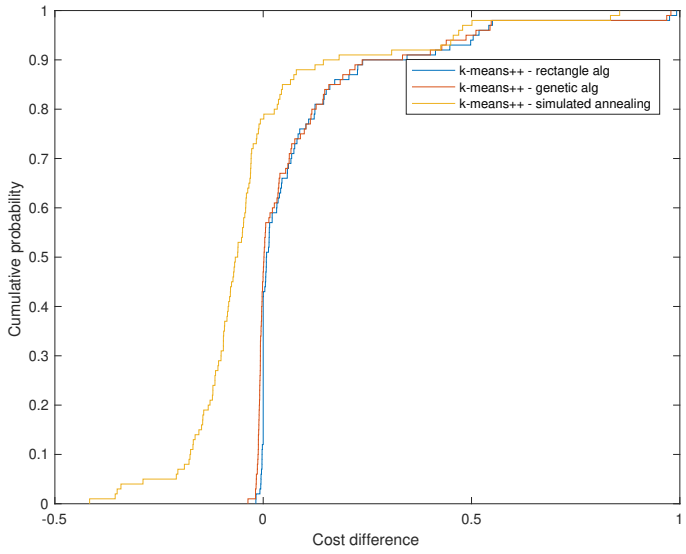

Figure 3.20 Comparison between the rectangle algorithm and other global optimization algorithms after 100,000 function evaluations.

CHAPTER 4

A NEW GLOBAL OPTIMIZATION ALGORITHM WITH DERIVATIVES

4.1 The Algorithm

In this section we describe a modification of the algorithm that uses derivatives in addition to function values, specialized to the 1-D case. Otherwise, the algorithm is as previously described. After n function evaluations we have a collection of intervals $\{R_i^n, 1 \le i \le n\}$. Denote the center and half-length of the ith interval by c_i^n and w_i^n, respectively.

Denote the smallest function value after n evaluations by $M_n = \min_{1\le i\le n}\{f(c_i^n)\}$ and the error Δ_n by $M_n - f(t^*)$. Let $\tau_n = 2\min_{1\le i\le n} w_i^n$ denote the smallest interval length.

The algorithm first evaluates the function at $c_1^1 = 1/2$. Define

$$g_n(x) = (x\log(n))^2$$

for $0 < x \le 1/2$.

For each interval R_i^n, define the first-order Taylor approximation

$$L_{i,n}(s) = f(c_i^n) + (s - c_i^n)f'(c_i^n), \quad s \in R_i^n,$$

and let $L_n(\cdot)$ denote the piecewise linear function that coincides with $L_{i,n}(\cdot)$ on the interior of R_i^n. For $n \ge 1$ and $1 \le i \le n$ set

$$\rho_i^n \equiv \frac{4w_i^n}{\sqrt{f(c_i^n) - M_n + g_n(c_i^n) - w_i^n f'(c_i^n)} + \sqrt{f(c_i^n) - M_n + g_n(c_i^n) + w_i^n f'(c_i^n)}}.$$

The equivalent integral form

$$\rho_i^n = \int_{s=c_i^n - w_i^n}^{c_i^n + w_i^n} \frac{ds}{\sqrt{L_{i,n}(s) - M_n + g_n(\tau_n)}}$$

will be useful.

Suppose we have made n evaluations. Compute ρ_i^n, $1 \le i \le n$, and let i be the first index such that $\rho_i^n \ge \rho_j^n$ for all $1 \le j \le n$. We next split R_i^n into three intervals, as described in Chapter 2, and evaluate the function at the center of two of the new intervals.

4.1.1 Convergence rate

This algorithm has the property that $\Delta_n \to 0$ as $n \to \infty$ for any continuous function f. In order to determine the rate at which the error converges to 0 we need to place some assumptions on f beyond continuity. Our main result is:

Theorem 4. *Let f be twice continuously differentiable with unique global minimizer $t^* \in (0,1)$ with $f''(t^*) > 0$. Then the error is asymptotically bounded as*

$$\liminf_{n\to\infty} \frac{\log(n)}{n} \log(1/\Delta_n) \ge \frac{f''(t^*)}{6\sqrt{3}}. \tag{4.1}$$

The rest of this section is devoted to the proof of the theorem. We can express

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} \rho_i^n &= \frac{1}{n}\int_{s=0}^{1} \frac{ds}{\sqrt{L_n(s) - M_n + g_n(\tau_n)}} \\ &= \frac{1}{n}\int_{s=0}^{t^*} \frac{ds}{\sqrt{L_n(s) - M_n + g_n(\tau_n)}} + \frac{1}{n}\int_{s=t^*}^{1} \frac{ds}{\sqrt{L_n(s) - M_n + g_n(\tau_n)}}.\end{aligned}$$

Since $f \in C^2$ and $f''(t^*) > 0$, there exists a positive number β such that

$$0 \le t^* - \beta < t^* + \beta \le 1$$

and f is convex on $[t^* - \beta, t^* + \beta]$, and furthermore

$$\frac{1}{3}as^2 \le f(t^* + s) - f(t^*) \le \frac{2}{3}as^2$$

for $|s| \leq \beta$. By choosing β small enough, we can also ensure that

$$f(t) \geq \min\{f(t^* - \beta), f(t^* + \beta)\}$$

for $t \notin [t^* - \beta, t^* + \beta]$. Since the minimizer t^* is unique, $f(t^* + s) - f(t^*)$ is bounded from below by a positive number for $|s| > \beta$.

By Taylor's theorem,

$$f(s) = f(c_i^n) + (s - c_i^n)f'(c_i^n) + 2(w_i^n)^2 f''(\xi), \quad s \in R_i^n,$$

for some $\xi \in R_i^n$. Set $B \equiv \max_{0 \leq s \leq 1} |f''(s)|$. Then

$$\max_{s \in R_i^n} |L_{i,n}(s) - f(s)| \leq 2B(w_i^n)^2.$$

For large enough n, the rectangle (say R_s^n) containing the minimizer will have length at most $3\tau_n$, and so

$$\frac{M_n - M}{g_n(\tau_n)} \leq \frac{\max_{t \in R_s^n} |L_{i,n}(t) - f(t)|}{g_n(\tau_n)} \leq \frac{2B(w_s^n)^2}{\tau_n^2 (\log n)^2} \leq \frac{9B\tau_n^2}{4\tau_n^2(\log n)^2} \to 0.$$

Using these facts,

$$\begin{aligned}
&\frac{1}{n}\int_{s=t^*}^{1} \frac{ds}{\sqrt{L_n(s) - M_n + g_n(\tau_n)}} \\
&\quad = \frac{1}{n}\int_{s=t^*}^{t^*+\beta} \frac{ds}{\sqrt{L_n(s) - M + (1 + o(1))g_n(\tau_n)}} + O(1/n) \\
&\quad \leq \frac{1}{n}\int_{s=t^*}^{t^*+\beta} \frac{ds}{\sqrt{f(s) - M + (1 + o(1))g_n(\tau_n)}} + O(1/n) \quad \text{by local convexity of } f \\
&\quad \leq \frac{1}{n}\int_{s=0}^{\beta} \frac{ds}{\sqrt{\frac{1}{3}as^2 + (1 + o(1))g_n(\tau_n)}} + O(1/n) \\
&\quad \leq \frac{1}{n}\frac{\sqrt{3}}{\sqrt{a}} \log\left(\frac{\sqrt{a}}{\tau_n \log n}\right) + O(1/n).
\end{aligned}$$

Bounding the integral from 0 to t^* in a similar way gives the upper bound

$$\frac{1}{n}\sum_{i=1}^{n} \rho_i^n \leq \frac{1}{n}\frac{2\sqrt{3}}{\sqrt{a}} \log\left(\frac{\sqrt{a}}{\tau_n \log n}\right) + O(1/n). \tag{4.2}$$

Recall that s denotes the index of the subinterval containing the minimizer. Since this interval eventually has width at most $3\tau_n$,

$$\rho_s^n \leq \frac{3\tau_n}{2\sqrt{g_n(\tau_n)}} = \frac{3/2}{\log n}.$$

In the neighborhood of t^* where f is convex, the ρ-values for each child will be at least a third the value of the parent's. For subintervals outside the neighborhood of t^*, the children of split intervals will have ρ values about a third of the parent's. Thus for all sufficiently large n,

$$\rho_s^n \geq \frac{1}{2\log n}$$

and

$$\frac{1}{3} \leq \liminf_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \leq \limsup_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \leq 3. \tag{4.3}$$

Therefore,

$$\begin{aligned}
3 &\geq \limsup_{n\to\infty} \frac{\rho_s^n}{\frac{1}{n}\sum_{i=1}^n \rho_i^n} \\
&\geq \limsup_{n\to\infty} \frac{\frac{1}{2\log n}}{\frac{1}{n}\frac{2\sqrt{3}}{\sqrt{a}} \log\left(\frac{\sqrt{a}}{\tau_n \log n}\right) + O(1/n)} \quad \text{(using 4.2)} \\
&= \limsup_{n\to\infty} \frac{\frac{1}{2\log n}}{\frac{1}{n}\frac{2\sqrt{3}}{\sqrt{a}} \left\{\log\left(\sqrt{a}\right) - \log\left(\tau_n\right) - \log\log n\right\} + O(1/n)},
\end{aligned}$$

or

$$\liminf_{n\to\infty} \frac{\log n}{n} \left\{\log\left(\sqrt{a}\right) - \log\left(\tau_n\right) - \log\log(1/\tau_n)\right\} \geq \frac{\sqrt{a}}{36\sqrt{3}}.$$

This implies that

$$\liminf_{n\to\infty} \frac{\log n}{n} \log(1/\tau_n) \geq \frac{a^{1/2}}{12\sqrt{3}}.$$

The subinterval containing the minimizer has width at most $3\tau_n$, and so for large n the error is bounded above by

$$\Delta_n \leq \frac{2}{3}a(3\tau_n)^2 = 6a\tau_n^2,$$

and so

$$\liminf_{n\to\infty} \frac{\log(n)}{n} \log(1/\Delta_n) \geq \liminf_{n\to\infty} \frac{\log(n)}{n} \log(1/6a\tau_n^2) \liminf_{n\to\infty} \frac{\log(n)}{n} 2\log(1/\tau_n^2) \geq \frac{\sqrt{a}}{6\sqrt{3}}. \tag{4.4}$$

This gives (4.1), and completes the proof of the theorem.

4.1.2 Discussion on convergence rate

The convergence rate of an algorithm for approximating the global minimum is significant for knowing how good a global optimization algorithm is. Most algorithms approximate the global minimum of a function using adaptively chosen function evaluations. Some algorithms are efficient for functions with properties such as uni-modality or convexity.

It is hard to know how to bound the error in the worst-case if there is no strong assumption on the objective function. Suppose that the second derivative of the objective function is bounded in absolute value by a finite number p. Then for any algorithm A that evaluates objective function f and its derivatives, the worst-case error is not better than exhaustive search. Define $\Delta_n(f)$ as the error which is the smallest of the first n function values minus the global minimum. We know that in the worst-case, there exists an $f \in F_p$ for which $\Delta_n(f) \geq c^2 p n^{-2}$ for a constant c.

This means for that any algorithm, the worst-case error satisfies

$$\liminf_{n\to\infty} \inf_{f\in F_p} \frac{1}{\log(n)} \log(1/\Delta_n(f)) \geq 2. \tag{4.5}$$

If we consider the subclass of functions which have a unique local minimizer, some algorithms such as golden section search method converge really fast. The golden section search method uses only function evaluations. Let $\phi \equiv \frac{1+\sqrt{5}}{2}$ denote the golden ratio. The golden search method produces a subinterval containing the minimizer of length $(1/\phi)^{n-1}$ after n function evaluations.

By Taylor's formula, we have $\Delta_n(f) \leq (1/2)h_n^2 f''(t^*) + o(h_n^2)$ where $h_n^2 = (1/\phi)^{n-1}$. We have the following result:

$$\liminf_{n\to\infty} \inf_{f\in F_p} \frac{1}{n} \log(1/\Delta_n(f)) \geq \log\left(\frac{3+\sqrt{5}}{2}\right). \tag{4.6}$$

The slow convergence of (4.5) and the fast convergence of (4.6) shows the extremes of optimization problem complexity. Our algorithm is on order of a factor $log(N)$ slower than the golden section algorithm. We can see that if we apply the global optimization algorithm to a function which has a unique optimum, there is a logarithmic slowdown compared with local optimization using the golden section algorithm. In other words, if the local optimization algorithm requires N function evaluations to approximate a minimum with prescribed accuracy, then the global optimization algorithm requires on order of $N\log(N)$ evaluations to approximate the minimum to the same accuracy.

4.2 Experiments

We perform experiments with the rectangle algorithm on quadratic and Rastrigin functions.

Quadratic function We use the general quadratic function which $f(x_i) = \sum_{i=1}^{n}(x_i - (0.5)^{\frac{1}{2}})^2$ to test the rectangle algorithm. The algorithm evaluated at the points are shown in Figure 4.1.

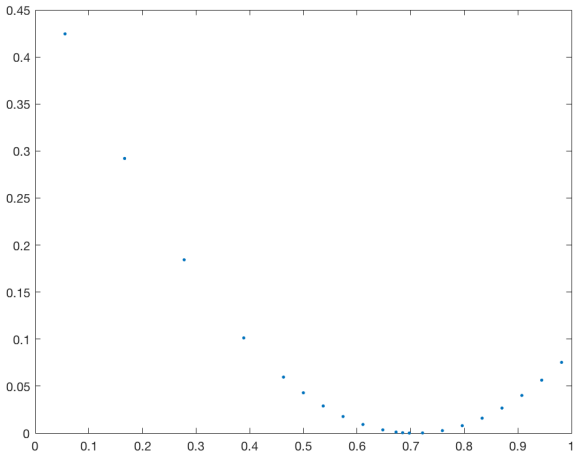

Figure 4.1 Quadratic 1-D objective function.

Rastrigin function We use the above Rastrigin function which was defined in Chapter 2 which can work well in the algorithm we proposed. In the one dimensional Rastrigin problem, the algorithm evaluated at the points shown in Figure 4.2.

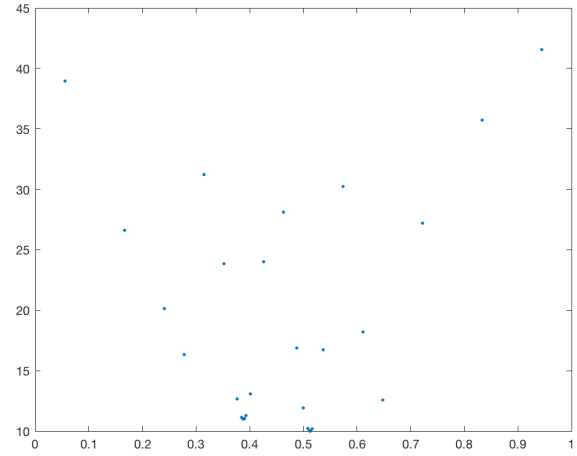

Figure 4.2 Rastrigin 1-D objective function.

CHAPTER 5

DISCRETE SETTING

5.1 Introduction

In this section, we consider the problem of approximating the minimum cost of a finite set of alternative systems. We can not directly observe the cost of the systems, but we can estimate the cost using simulation. The simulation run lengths are adaptively chosen for each system. We describe an optimization algorithm and establish a bound on the error convergence rate. Compared with a single system, the error grows by an additional factor of the square root of the logarithm of the number of systems and the simulation budget. Consider the following optimization problem. We are interested in choosing a parameter from a finite set that optimizes the long-run average performance of a stochastic system. The system is too complicated to allow for analytic treatment, but we can simulate the performance at each parameter value.

Suppose that the parameter set is $\Theta = \{\theta_1, \theta_2, \ldots, \theta_\nu\}$, and the corresponding performance values are $\{\mu_1, \mu_2, \ldots, \mu_\nu\}$. For simplicity assume that the $\{\mu_i\}$ are distinct, and without loss of generality assume that $\mu_1 < \mu_2 < \cdots < \mu_\nu$. For each θ_i we can run a simulation, observing i.i.d. random variables $\{Y_{i,j} : j = 1, 2, \cdots\}$ with mean μ_i and variance σ_i^2. We assume that the $\{Y_{i,j}, 1 \leq i \leq \nu, j \geq 1\}$ are mutually independent and defined on a common probability space $(\Omega, \mathcal{F}, P)$. We further assume that for some $\epsilon > 0$,

$$EY_{i,j}^{2+\epsilon} < \infty.$$

We can adaptively increase the simulation lengths n_i so that the simulation effort is concentrated on the parameter values that appear most promising over time. Let n_i denote the number of simulation iterations at the ith system; that is, we have

computed estimates based on $\{Y_{i,1}, \ldots, Y_{i,n_i}\}$. Let

$$n = \sum_{i=1}^{\nu} n_i.$$

Let $\mu_{n,i}$ denote the sample mean and $\sigma^2_{n,i}$ the sample variance for the simulation of the ith system:

$$\mu_{n,i} = \frac{1}{n_i} \sum_{j=1}^{n_i} Y_{i,j} \tag{5.1}$$

and

$$\sigma^2_{n,i} = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} \left(Y_{i,j} - \mu_{n,i}\right)^2. \tag{5.2}$$

After a total of n simulation steps, we have estimates $\mu_{n,i}$ and $\sigma_{n,i}$ such that

$$Z_{n,i} \equiv \frac{\sqrt{n_i}}{\sigma_{n,i}} \left(\mu_{n,i} - \mu_i\right) \xrightarrow{d} N(0,1) \tag{5.3}$$

as $n_i \to \infty$, for $i = 1, 2 \ldots, \nu$. We use the notation $X_n \xrightarrow{d} F$ to indicate that the sequence of random variables $\{X_n\}$ converges in distribution to the distribution F, and $X_n \xrightarrow{P} X$ to denote that the random variables $\{X_n\}$ converge in probability to the random variable X. We denote the standard normal distribution by $N(0,1)$, and its cumulative distribution function by Φ.

Let $\mu_{n,*} = \min_{1 \le i \le \nu} \mu_{n,i}$ and let i^*_n be the corresponding index, so that $\mu_{n,*} = \mu_{n,i^*_n}$; this will serve as our estimate of $\mu_1 = \min_i \mu_i$. We will construct an algorithm for running the simulations to efficiently estimate μ_1.

Our goal is to construct a small interval $[\alpha_n, \beta_n]$ such that

$$P\left(\alpha_n \le \mu_1 \le \beta_n\right) \to 1$$

as $n \to \infty$.

If we knew in advance that the first system was the best, then we could allocate all observations to that system and the central limit theorem implies that

$$P\left(\mu_{n,1} - \frac{\sigma_{n,i}}{\sqrt{n}} z_\alpha \le \mu_1 \le \mu_{n,1} + \frac{\sigma_{n,i}}{\sqrt{n}} z_\alpha\right) \to 1 - \alpha$$

as $n \to \infty$, where

$$1 - \Phi(z_\alpha) = \frac{\alpha}{2}.$$

Therefore, if γ_n is any increasing sequence going to $+\infty$, then the probability that μ_1 is contained in an interval of half-width

$$\frac{\sigma_{n,i}}{\sqrt{n}} \gamma_n \tag{5.4}$$

tends to 1.

Without such advance knowledge, we must allocate observations to systems $2, 3, \ldots, \nu$. We will show that the number of evaluations at sub-optimal points grows as the logarithm of the total number of evaluations.

5.2 Background

To decide the best system using minimum cost in finite alternative systems by stochastic simulation is the problem we focused on. The reference [37] developed procedures for selecting the best or near-best of a finite number of simulated systems and compared with indifference-zone procedures. The reference [47] discussed similar problems as predicting simulation budget and guarantee probably approximately correct selection.

Approximating the minimum cost value is related to the problem of selecting the system with the smallest cost value. The latter problem is reviewed in [24, 41]. Our main reason for concentrating on the problem of approximating the minimum value instead of the minimizing parameter is that selecting the best system makes

more sense for small, or at least finite, sets of alternatives. In cases of a continuum of alternatives, possibly with many isolated global minimizers, selecting the best system is not as well defined as approximating the minimum value. Nevertheless, the two problems are clearly related.

We take as given the performance measures $\{\mu_i\}$. An alternative is to start with a prior probability distribution on the $\{\mu_i\}$. This approach of Bayesian optimization is surveyed in [9, 22].

Our approach might be called a single-stage approach, in that the algorithm does not perform preliminary simulations that are used to plan subsequent simulations. Such single-stage procedures are presented in [55], where the emphasis is on constructing asymptotically valid confidence intervals for the difference of cost values. We do not construct confidence intervals in the sense of that paper.

The algorithm described in this dissertation is similar to optimization algorithms described in works on continuous Bayesian optimization; for example see [6]. In that work, the goal was to minimize a continuous function defined on the unit interval. The unknown function was assumed to be a sample path of a Wiener process. The algorithm adaptively chooses points to evaluate the random function f, with the error $\Delta_n(f)$ after n evaluations taken to be the difference between the smallest observed function value and the global minimum. It was shown that for all $r \in [1, \infty)$ and for all $p \in [1, \infty)$ there is a version of the algorithm (depending on those quantities) such that

$$(E|\Delta_n(f)|^p)^{1/p} \leq c \cdot n^{-r}$$

for a constant c and for all n. That is, any polynomial error rate can be obtained. This compares with the optimal nonadaptive error rate of $n^{-1/2}$.

The main motivation is to obtain insight into the discrete optimization problem as the number of alternatives $\nu \to \infty$. In order to formulate a reasonable version

of this question we impose some structure on the $\{\mu_i\}$. In Section 5.5 we consider a smooth cost function $f : [0,1] \to \mathbb{R}$, that can only be evaluated by simulation. We adaptively run simulations at $f(i/\nu)$, $1 \leq i \leq \nu$, according the the proposed algorithm. As $\nu \to \infty$, we show that the rate at which the evaluations concentrate on the minimizer depends on f through the second derivative at the minimizer.

5.3 The Algorithm

Recall that i_n^* is the index of the system with the smallest estimate $\mu_{n,i}$ after n simulation steps.

Define

$$g_{n,\nu} \equiv \sqrt{2\log(n\nu)}\frac{\sigma_{n,i_n^*}}{\sqrt{n_{i_n^*}}}$$

and

$$\rho_i^n \equiv \frac{\sigma_{n,i}^2}{n_i}\frac{1}{\left(\mu_{n,i} - \mu_{n,i_n^*} + g_{n,\nu}\right)^2}.$$

The idea of the algorithm is, at step n, to simulate the system i with the largest ρ_i^n. Roughly, this can be thought of as maximizing the probability that the next evaluation is below $\mu_{n,i_n^*} - g_{n,\nu}$. The "gap" $g_{n,\nu}$ is chosen large enough that these probabilities (and therefore the ρ_i^n) go to zero.

To simplify the exposition, we will assume that each system is initially simulated for two steps so that the sample mean and variances are defined. The algorithm for a simulation budget $N > 2\nu$ follows.

1. Simulate 2 steps for each of the ν systems, set $n_i = 2$, $1 \leq i \leq \nu$, and compute $\mu_{n,i}$ and $\sigma_{n,i}$, $1 \leq i \leq \nu$. Set i_n^* to the (first) index with minimal sample mean, and set $n = 2\nu$.
2. Compute ρ_i^n for each $i \leq \nu$, and let k be the (first) index with $\rho_k^n \geq \rho_i^n \ \forall i$.

3. Simulate the kth system for one step, and update the sample mean and variance of the kth system.
4. Set $n_k = n_k + 1$, $n = n + 1$, and if $n < N$ return to step 2.

We will only consider times at which we are about to evaluate the currently best system; that is, when $\rho^n_{i^*_n} \geq \rho^n_k$ for all k. Note that at such a time

$$\rho^n_k \leq \rho^n_{i^*_n} \leq \frac{1}{2\log(n\nu)}; \tag{5.5}$$

that is, for each i,

$$\frac{\sigma^2_{n,i}}{n_i} \frac{1}{\left(\mu_{n,i} - \mu_{n,i^*_n} + g_{n,\nu}\right)^2} \leq \frac{1}{2\log(n\nu)}.$$

The event that the cost of the ith system is above our lower bound $\mu_{n,i^*_n} - g_{n,\nu}$ is

$$\begin{aligned}
\{\mu_i > \mu_{n,i^*_n} - g_{n,\nu}\} &= \left\{\mu_{n,i} - \frac{\sigma_{n,i}}{\sqrt{n_i}} Z_{n,i} > \mu_{n,i^*_n} - g_{n,\nu}\right\} \quad \text{by (5.3)} \\
&= \left\{Z_{n,i} < \frac{\sqrt{n_i}}{\sigma_{n,i}} \left(\mu_{n,i} - \mu_{n,i^*_n} + g_{n,\nu}\right)\right\} \\
&= \left\{Z_{n,i} < \frac{1}{\sqrt{\rho^n_i}}\right\} \\
&\supset \left\{Z_{n,i} < \sqrt{2\log(n\nu)}\right\} \quad \text{by (5.5)} \\
&\supset \left\{Z_{n,i} < \sqrt{2\log(n_i\nu)}\right\}.
\end{aligned}$$

Therefore, by (5.3),

$$P\left(\mu_i > \mu_{n,i^*_n} - g_{n,\nu}\right) \to 1$$

as $n \to \infty$. It follows that

$$\begin{aligned}
P(\Delta_n \leq g_{n,\nu}) &= P(\mu_{n,i^*_n} - \mu_1 \leq g_{n,\nu}) \\
&= P\left(\bigcap_{i=1}^{\nu} \{\mu_i > \mu_{n,i^*_n} - g_{n,\nu}\}\right) \\
&\to 1
\end{aligned}$$

as $n \to \infty$.

The amount of simulation of the ith system satisfies

$$n_i \approx \frac{2\log(n\nu)\sigma_{n,i}^2}{(\mu_{n,i} - \mu_{n,i_n^*} + g_{n,\nu})^2},$$

and so for $i > 1$,

$$\frac{n_i}{\log(n\nu)} \xrightarrow{P} \frac{2\sigma_i^2}{(\mu_i - \mu_1)^2}$$

as $n \to \infty$. Also,

$$\frac{n_{i_n^*}}{n} \xrightarrow{P} 1.$$

This implies (5.3) ([2], Theorem 17.1).

5.4 Main Result

Set

$$\mathcal{H} = \sum_{i=2}^{\nu} \frac{\sigma_i^2}{(\mu_i - \mu_1)^2}.$$

This is a measure of how hard the optimization is. If the small μ_i's are clustered around the minimum, then the algorithm needs to spread out the effort over multiple systems that seem promising; this corresponds to a large $\mathcal{H}$. If μ_2 is much larger than the minimum μ_1, then the search can concentrate on μ_1 and $\mathcal{H}$ is small.

Then we have

$$n_{i_n^*} \approx n - 2\log(n\nu)\mathcal{H}$$

and

$$\frac{n_{i_n^*}}{n} \xrightarrow{P} 1.$$

The following limit theorem shows how small our interval containing the minimizer with probability approaching one can be. This theorem, as well as Theorem 6 in the next section, are proved in [89].

Theorem 5. *Consider a system with ν alternatives. Let*

$$\Delta_n \equiv \mu_{n,i_n^*} - \mu_1$$

denote the approximation error after n simulation steps. Then

$$\lim_{n\to\infty} P\left(-\frac{\sigma_1}{\sqrt{n}}\frac{\sqrt{2\log(n\nu)}}{\sqrt{1-2\mathcal{H}\log(n\nu)/n}} \le \Delta_n \le \frac{\sigma_1}{\sqrt{n}}\frac{\sqrt{2\log(n\nu)}}{\sqrt{1-2\mathcal{H}\log(n\nu)/n}}\right) = 1.$$

This shows the slowdown we suffer compared to (5.4). We see that whereas the sequence γ_n introduced at (5.4) can grow arbitrarily slowly, for our algorithm it must grow at rate $\sqrt{\log(n\nu)}$.

5.5 Concentration Rate

The $\{\mu_i\}$, and hence $\mathcal{H}$, could be quite arbitrary. In this section we consider the case where the goal is to approximate the minimum of a continuous cost function by simulating values at a fixed grid of points. Denote the cost function by $f : [0,1] \to \mathbb{R}$. We estimate the values $f(i/\nu)$, $i = 1, \ldots, \nu$. With our previous notation, $\{\mu_i, 1 \le i \le \nu\} = \{f(i/\nu), 1 \le i \le \nu\}$, though the ordering is different in general.

In this section, assume that $\sigma_i \equiv \sigma$ for each $1 \le i \le \nu$. Let us suppose that $f \in C^2([0,1])$ with unique global minimizer $t^* \in (0,1)$ with $f''(t^*) > 0$.

Let Π_n denote the proportion of evaluations that are not at the current estimator of the minimizer:

$$\Pi_n = 1 - \frac{n_{i_n^*}}{n}.$$

Set

$$B(f,\nu,n) \equiv \frac{\pi}{4}\nu\frac{\sqrt{\sigma}}{\sqrt{\beta(f,\nu)}}\left(\frac{2\log(n\nu)}{n}\right)^{1/4},$$

where $\beta(f,\nu) \to f''(t^*)$ as $\nu \to \infty$.

Theorem 6. *As the number of evaluations n tends to infinity,*

$$P\left(\Pi_n \leq B(f,\nu,n)\right) \to 1.$$

The bound $B(f,\nu,n)$ increases roughly linearly in the discretization order ν, and as the square root of the noise standard deviation σ. For large ν, $\beta(f,\nu) \approx f''(t^*)$ and so $B(f,\nu,n)$ increases as $1/\sqrt{f''(t^*)}$. For f that increases rapidly moving away from the global minimizer, i.e., for large $f''(t^*)$, the algorithm can concentrate the evaluations more near the minimizer, thus reducing Π_n.

5.6 Numerical Experiments

We did some experiments for the algorithm applied to the discretization of a smooth cost function, as described in the last section. The cost function is

$$f(x) = \frac{1}{2} + 2x - \cos(12x - 9), \quad 0 \leq x \leq 1, \tag{5.6}$$

which is shown in Figure 5.1. Figure 5.2 shows the discrete version of the optimization problem, where the ith system cost is $f(i/\nu)$ for $\nu = 30$ and $1 \leq i \leq \nu$.

For a fixed number ν, our goal is to approximate the minimum of $f(i/\nu)$, $1 \leq i \leq \nu$. This will be based on evaluations

$$Y_{i,j} = f(i/\nu) + \xi_{i,j},$$

where the $\xi_{i,j}$ are independent standard normal random variables.

Figure 5.2 shows the results of running the algorithm on the function shown in Figure 5.1 with $\nu = 30$ and $n = 40{,}000$. The error bars around each estimated function value indicate the standard deviation of the estimator.

In Figure 5.3 and 5.4, we plot the normalized error for the algorithm applied to discretizations of $\nu = 50$ in the left plot and $\nu = 100$ in the right plot. From Theorem

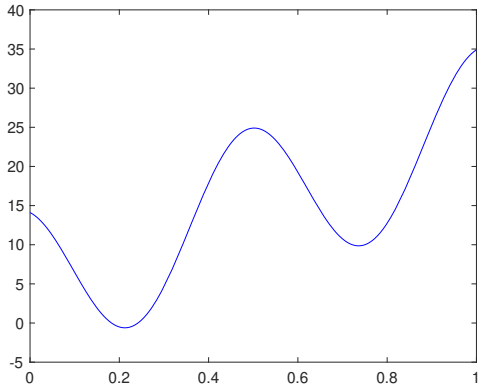

Figure 5.1 Continuous cost function (5.6).

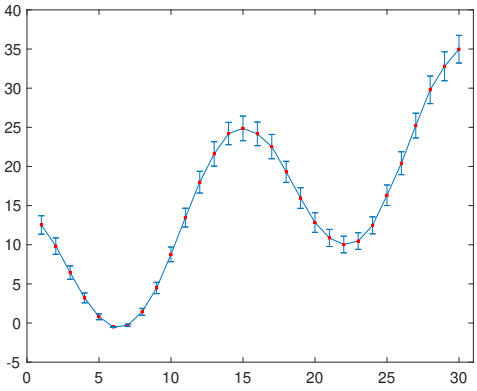

Figure 5.2 Estimates of discretized cost function (5.6).

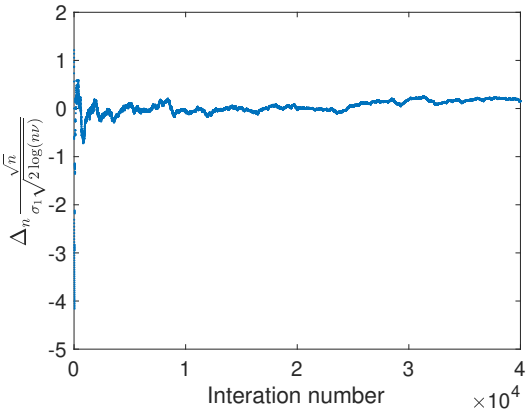

Figure 5.3 Normalized proportion of evaluations at sub optimal points, $\nu = 50$.

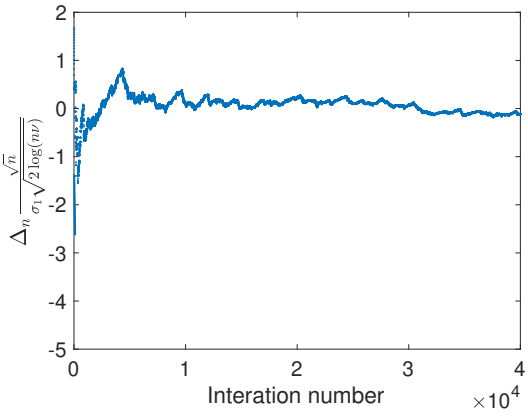

Figure 5.4 Normalized proportion of evaluations at sub optimal points, $\nu = 100$.

5, the normalized error

$$\frac{\sqrt{n}}{\sigma_1\sqrt{2\log(n\nu)}}$$

will lie in the interval $[-1, 1]$ with probability approaching 1 as $n \to \infty$. The normalized error is plotted as a function of the iteration number n for $\nu = 50$ (left plot) and $\nu = 100$ (right plot).

In Figure 5.5 and 5.6, we plot the proportion of evaluations made at points other than the current estimated minimizer, Π_n, divided by our upper bound $B(f, \nu, n)$ from

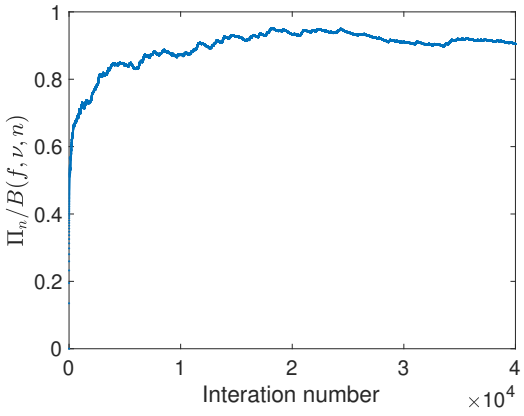

Figure 5.5 Normalized error, $\nu = 50$.

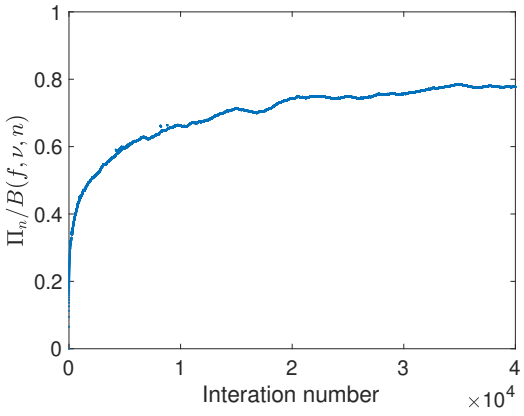

Figure 5.6 Normalized error, $\nu = 100$.

Theorem 6. We evaluate the rate $\Pi_n/B(f,\nu,n)$ with $\nu = 50$ and $\nu = 100$ with $\sigma = 1$. The total iteration number is $40,000$.

Theorem 6 implies that the ration should be below 1 with probability approaching 1 as $n \to \infty$. The results are plotted for $\nu = 50$ (left plot) and $\nu = 100$ (right plot).

5.7 Conclusions

We have constructed a single-stage procedure for adaptively controlling the simulation of multiple systems with the aim of efficiently approximating the minimum cost measure. We constructed an interval that contains the minimum cost measure with probability approaching 1 as the simulation budget grows. Compared with the ideal situation of a single system, the size of the enclosing interval grows by an additional factor of the square root of the logarithm of the number of systems and the simulation budget.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary

The dissertation presents two different global optimization algorithms, one using only function values, the other using also derivatives. Discussions about convergence rates of algorithms also are provided. They have good performance for many mathematical test functions. Applying the first algorithm to automatic image registration requires no parameters to tune the algorithm. Image registration gives rise to objective functions that may have many local minimizers if there is no obvious object in the images. In addition, applying the global optimization algorithm to clustering for finding good start points for the k-means algorithm often resulted in better solutions. We explored similar optimization problems with discrete variables and with noise corrupted function evaluations. We present a global optimization algorithm to choose a parameter from a finite set to optimize the long-run average performance of a stochastic system.

Still many theoretical and practical problems remain in the continuous and discrete settings. For example how to provide mathematical proofs and comparisons of the DIRECT algorithm and the algorithm we proposed and how to apply those algorithms in more realistic situations. Also, it is interesting to explore more about the relationship between the continuous setting and discrete setting.

BIBLIOGRAPHY

[1] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. *Proceeding SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[2] P. Billingsley. *Convergence of probability measures*. Wiley, New York, 1968.

[3] C. G. E. Boender and H. E. Romeijn. Stochastic methods. *Handbook of Global Optimization*, 1995.

[4] F. H. Branin and S. K. Hoo. A method for finding multiple extrema of a function of n variables. *Numerical Methods of Nonlinear Optimization*, pages 231–237, 1972.

[5] J. M. Calvin, G. Gimbutienė, W. O. Phillips, and A. Žilinskas. On convergence rate of a rectangular partition based global optimization algorithm. *Journal of Global Optimization*, 71:165–191, 2018.

[6] J. M. Calvin, M. Hefter, and A. Herzwurm. Adaptive approximation of the minimum of brownian motion. *Journal of Complexity*, 39:17–37, 2017.

[7] R. B. Cattell. The description of personality: Basic traits resolved into clusters. *Journal of Abnormal and Social Psychology*, 38:476–506, 1943.

[8] Y. Chen, R. R. Brooks, S. S. Iyengar, N. S. V. Rao, and J. Barhen. Efficient global optimization for image registration. *IEEE Transactions on Knowledge and Data Engineering*, 14:79–92, 2002.

[9] S. E. Chick. Bayesian methods: Bayesian methods for simulation. In K. Kang J. A. Joines, R. R. Barton and eds. P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 109–118, Orlando, Florida, 2000. Society for Computer Simulation International.

[10] J. H. Holland D. E. Goldberg. Genetic algorithms and machine learning. *Machine Learning*, 2:95–99, 1988.

[11] B. D. de Vos, F. F.Berendsen, M. A. Viergever, H. Sokooti, M. Staring, and I.Išgum. A deep learning framework for unsupervised affine and deformable image registration. *Medical Image Analysis*, 52:128–143, 2019.

[12] I. Diener. Trajectory methods in global optimization. *Nonconvex Optimization and Its Applications*, 2:649–668, 1995.

[13] L. C. W. Dixon and G. P. Szego. The global optimization problem: an introduction. *Towards global optimization*, 2:1–15, 1978.

[14] H. E. Driver and A. L. Kroeber. Quantitative expression of cultural relationships. *University of California Publications in American Archaeology and Ethnology*, 31:211–256, 1932.

[15] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[16] R. D. Eastman, N. S. Netanyahu, and Jacqueline le Moigne. Survey of image registration methods. *Image Registration for Remote Sensing*, 79:157–181, 1993.

[17] H. Eduardo and E. Nelson. A genetic algorithm for cluster analysis. *Intell. Data Anal.*, 7:15–25, 02 2003.

[18] F. E. A. EI-Gamal, M. Elmogy, and A. Atwan. Current trends in medical image registration and fusion. *Egyptian Informatics Journal*, 17:99–124, 2016.

[19] D. E. Finkel. Global optimization with the DIRECT algorithm. *PhD thesis*, North Carolina State University, Raleigh, North Carolina, 2005.

[20] D. E. Finnkel and C. T. Kelley. Convergence analysis of the direct algorithm. *Technical Report CRSC-TR04-28*, 2004.

[21] C. A. Floudas and P. M. Pardalos. *Recent advances in global optimization*. Princeton university press, Princeton, 2014.

[22] P. I. Frazier. A tutorial on bayesian optimization. *eprint arXiv*, 1807.02811, 2018.

[23] A. Gimbutas and A. Žilinskas. An algorithm of simplicial Lipschitz optimization with the bi-criteria selection of simplices for the bi-section. *Journal of Global Optimization*, 71:115–127, 2018.

[24] D. Goldsman, S. Kim, W. S. Marshall, and B. L. Nelson. Ranking and selection for steady-state simulation: Procedures and perspectives. *Informs Journal on Computing*, 14(1):2–19, 2002.

[25] A. Goshtasby. Image registration by local approximation methods. *Image and Vision Computing*, 6:255–261, 1988.

[26] E. R. Hansen. Global optimization using interval analysis. *Marcel Dekker, Inc.*, 1992.

[27] E.R. Hansen. *Global Optimization using Interval Analysis*. McGraw-Hill, Inc, New York, 1992.

[28] R. Horst and P. M. Pardalos. *Handbook of Global Optimization*. 1995.

[29] D. Ikami, T. Yamasaki, and K. Aizawa. Local and global optimization techniques in graph-based clustering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3456–3464, 2018.

[30] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

[31] M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5:142–156, 2001.

[32] J. Joglekar and S. S. Gedam. Area based image matching methods – a survey. *International Journal of Emerging Technology and Advanced Engineering*, 2, 2012.

[33] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79:157–181, 1993.

[34] A. H. G. Rinnooy Kan, C. G. E. Boender, and G. Th. Timmer. A stochastic approach to global optimization. *Computational Mathematical Programming*, 15:281–308, 1984.

[35] R.B. Kearfott and V. Kreinovich. Applications of interval computations. *Applied Optimization*, 1996.

[36] J. Kiefer. Sequential minimax search for a maximum. *Proc. Amer. Math. Soc.*, 4:502–506, 1953.

[37] S. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation*, 11(3):251–273, 2001.

[38] D. E. Kvasov and Ya. D. Sergeyev. Lipschitz gradients for global optimization in a one-point-based partitioning scheme. *Journal of Computational and Applied Mathematics*, 236:4042–4054, 2012.

[39] T. Kwok and D. Yeung. Objective functions for training new hidden units in constructive neural networks. *IEEE Transaction on Neural Networks*, 8, 1997.

[40] P. J. M. Laarhoven and E. H. L. Aarts. *Simulated annealing: theory and applications*. 1987.

[41] S. Lee and B. L. Nelson. General-purpose ranking and selection for computer simulation. *IIE Transactions*, 48(6):555–564, 2016.

[42] D. Lera and Ya. D.Sergeyev. Deterministic global optimization using space-filling curves and multiple estimates of Lipschitz and Hölder constants. *Communications in Nonlinear Science and Numerical Simulation*, 23:328–342, 2015.

[43] D. Lera and Ya. D. Sergeyev. Gosh: derivative-free global optimization using multi-dimensional space-filling curves. *Journal of Global Optimization*, 71:193–211, 2018.

[44] G. Liuzzi, S. Lucidi, and V. Piccialli. A partition-based global optimization algorithm. *Journal of Global Optimization*, 48(1):113–128, 2010.

[45] X. 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*, 6(2):301–310, 04 2019.

[46] D. Lera M. Gaviano, D. E. Kvasov and Ya. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software*, 29:469–480, 2003.

[47] S. Ma and Shane G. Henderson. Predicting the simulation budget in ranking and selection procedures. *ACM Transactions on Modeling and Computer Simulation*, 29(3):14:1–14:25, 2019.

[48] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *IEEE Transactions On Medical Imaging*, 16:187–198, 1997.

[49] J. B. A. Maintz and M. A. Viergever. An overview of medical image registration methods. 1996.

[50] S. Merendino and M. E. Celebi. A simulated annealing clustering algorithm based on center perturbation using gaussian mutation. In *FLAIRS Conference*, 2013.

[51] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. Machine learning, neural and statistical classification. *Ellis Horwood Series in Artificial Intelligence*, 1994.

[52] H. Muhlenbein and D. S. Voosen. Predictive models for the breeder genetic algorithm: continuous parameter optimization. *Evolutionary Computation*, 1:25–49, 1993.

[53] R. Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends® in Machine Learning*, 7(1):1–129, 2014.

[54] S. Nag. Image registration techniques: a survey. *arXiv:1712.07540*, 79:157–181, 2017.

[55] M. K. Nakayama. Asymptotically valid single-stage multiple-comparison procedures. *Journal of Statistical Planning and Inference*, 139(4):1348–1356, 2009.

[56] A. Neumaier. Introduction to Global Optimization. https://www.mat.univie.ac.at/ neum/glopt/intro.html. [Online].

[57] A. Neumaier. Interval methods for systems of equation. *Encyclopedia of Mathematics and its Applications*, page 37, 1990.

[58] I. H. Osman and J. P. Kelly. Meta-heuristics:theory and applications. *Kluwer Academic Publishers*, 1996.

[59] U. E. Ruttimann P. Thkvenaz and M. Unser. Iterative multiscale registration without landmarks. *IEEE International Conference on Image Processing*, 3:228–231, 1995.

[60] R. Paulavičius and J. Žilinskas. *Simplicial global optimization*. Springer, New York, NY, 2014.

[61] J. D. Pintér. *Global Optimization in Action*. 1996.

[62] J. P. Pluim, J. B. Maintz, and M. A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Trans Medical Imaging*, 22:986–1004, 2003.

[63] L. Ramirez, N. G. Durdle, and V. J. Raso. Medical image registration in computational intelligence framework: a review. *Electrical and Computer Engineering*, 2:1021–1024, 2003.

[64] L. A. Rastrigin. Systems of extremal control. *Mir, Moscow*, 1974.

[65] H. Ratschek and J. Rokne. Interval methods. *Handbook of Global Optimization*, pages 751–828, 1995.

[66] A. Roche. Unifying maximum likelihood approaches in medical image registration. *International Journal of Imaging Systems and Technology*, 11:71–80, 2000.

[67] J. M. Rouet, J. J. Jacq, and C. Roux. Genetic algorithms for a robust 3-d mr-ct registration. *IEEE Engineering in Medicine and Biology Society*, 4:126–136, 2000.

[68] H. S. Sawhney. True multi-image alignment and its applications to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:235–243, 1999.

[69] S. Z. Selima and K. Alsultanb. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, pages 1003–1008, 1991.

[70] Ya. D. Sergeyev and D. E. Kvasov. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization*, 16:910–937, 2006.

[71] Ya. D. Sergeyev and D. E. Kvasov. A deterministic global optimization using smooth diagonal auxiliary functions. *Communications in Nonlinear Science and Numerical Simulation*, 21:99–111, 2015.

[72] Ya. D. Sergeyev and D. E. Kvasov. *Deterministic global optimization*. Springer, New York, NY, 2017.

[73] Ya. D. Sergeyev, R. G. Strongin, and D. Lera. *Introduction to global optimization exploiting space-filling curves*. Springer, New York, NY, 2013.

[74] R. K. Sharma. Multisensor image registration. *Processings of the Society for Information Piexls*, 1997.

[75] G. Song, J. Han, Y. Zhao, Z. Wang, and H. Du. A review on medical image registration as an optimization problem. *Curr Med Imaging Rev*, 13(3):274–283, 2017.

[76] J. P. P. Starink and E. Backer. Finding point correspondence using simulated annealing. *Pattern Recognition*, 28:231–240, 1995.

[77] A. Tangherloni, L. Rundo, and M. S. Nobile. Proactive particles in swarm optimization: a settings-free algorithm for real-parameter single objective optimization problems. *Evolutionary Computation*, pages 1940–1947, 2017.

[78] A. S. Tikhomirov. On the markov homogeneous optimization method. *Computational Mathematics and Mathematical Physics*, (46):361–375, 2006.

[79] J. F. Traub, G. W. Wasilkowski, and H. Wozniakowski. *Information-Based Complexity*. Academic Press, New York, 1988.

[80] R. Tryon. Cluster analysis: correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality. *Edwards Brothers*, 1939.

[81] P. Viola. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24:137–154, 1997.

[82] C. Wang, W. Pedrycz, J. Yang, M. Zhou, and Z. Li. Wavelet frame-based fuzzy c-means clustering for segmenting images on graphs. *IEEE Transactions on Cybernetics*, 2019.

[83] G. W. Wasilkowski. Some nonlinear problems are as easy as the approximation problem. *Comput. Math. Appl.*, 10:351–363, 1984.

[84] Y. Wen, L. Zhang, L. He, and M. Zhou. Incorporation of structural tensor and driving force into log-demons for large-deformation image registration. *IEEE Trans on Image Processing*, 28(12):6091–6102, 12 2019.

[85] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

[86] X. Xu, J. Li, M. Zhou, J. Xu, and J. Cao. Accelerated two-stage particle swarm optimization for clustering not-well-separated data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

[87] D. Q. Zeebaree, H. Haron, A. M. Abdulazeez, and S. Zeebaree. Combination of k-means clustering with genetic algorithm: A review. *International Journal of Applied Engineering Research*, 12:14238–14245, 2017.

[88] C. Zhang and H.-P. Wang. Mixed-discrete nonlinear optimization with simulated annealing. *Engineering Optimization*, 21:277–291, 1993.

[89] C. Zheng, J. Calvin, and C. Gotsman. A DIRECT-type global optimization algorithm for image registration. *Journal of Global Optimization*, 2020.

[90] A. Zhigljavsky and A. Žilinskas. *Stochastic Global Optimization*. 2008.

[91] A. A. Zhigljavsky. *Theory of Global Random Search*. 1991.

[92] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.

[93] J. Zubin. A technique for measuring like-mindedness. *The Journal of Abnormal and Social Psychology*, 33(4):508–516, 1938.