ENHANCING SIMULATION OPTIMIZATION METHODS USING SMOOTHING AND METAMODELING TECHNIQUES

MA SICONG

(B.Eng, Shanghai Jiao Tong University, China)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE
2013

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Ma Sicong

Ma, Sicong
13 August 2013

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my main supervisor, Dr. Kim Sujin. Without her supervision, encouragement, support and patience, this thesis would not be possible. Her invaluable advice will always guide me for the rest of my life. I would also like to thank my co-supervisor, Prof. Tang Loon Ching, who is always patient and supportive of my work.

I also owe my gratitude to Dr. Cheon Myungsuk, Dr. Xu Lu and Dr. Song Jinhwa from ExxonMobil Research and Engineering Company. Their invaluable comments and advice during our collaboration help a lot for my work.

My thanks also go to other professors and my seniors in ISE department of NUS for their help and encouragement though all the way. They are Prof. Goh Thong Ngee, A/Prof. Poh Kim Leng, A/Prof. Lee Loo Hey, Dr. Ng Tsan Sheng, Fu Yinghui, Mu Aoran, Chen Liqin, Wang Qiang, and Li Juxin.

Last but not least, I would like to thank my parents for their continuing support.

Contents

Summary

Simulation optimization is widely applied and can be found in many fields with the development of computer technology. However, there are many issues arise in dealing with simulation optimization problems, which make them generally more difficult to solve compared with the ordinary optimization problems. One of the main issues is that the simulation programs could be extremely expensive and time consuming. Also, the form of the objective function is unknown and the response value returned by simulation experiment possesses random noise. The purpose of this study is to explore enhanced simulation algorithms to reduce simulation evaluations in both stochastic and deterministic cases.

In the stochastic case, a specific type of optimization problem with a probability objective is studied. Sample average approximation (SAA) method is one of the most popular methods to deal with stochastic optimization problems. Since probability functions essentially are expected value of an indicator function, optimization algorithms exhibiting a local convergence cannot work well on them when the SAA method is applied since the corresponding SAA function is ill-structured. We propose a smoothed SAA method, which incorporates smoothing techniques into the classic SAA method, so that a wide range of nonlinear optimization algorithms can be successfully applied even when the sample size is small. The convergence results of the method are also discussed.

The method is further developed by specifically focusing on direct search methods. We also incorporate our smoothing technique into the retrospective-approximation (RA) method, which solves a sequence of SAA problems with increasing sample size and decreasing error-tolerance driven by a computational strategy. The choice of parameters in the RA method in combination with direct search algorithms is investigated. According to the numerical results, the new algorithm performs better than the standard SAA method with smoothing technique. It uses less simulation evaluations in average to achieve a given accuracy level and exhibits more stable performance. In conclusion, the two enhanced methods provide new options to the users who deal with expensive probability simulation optimization problems.

In the deterministic case, we propose a new framework to incorporate local metamodeling techniques into direct search methods to improve the efficiency of the algorithms without affecting the convergence. The principle behind is to construct a local picture based on the available information so that a more promising candidate can be identified and visited first. We discuss the conditions on the local metamodels and the objective functions to guarantee the improvement of direct search algorithms. Numerical results with a wide variety of test problems show that a large amount of evaluations can be save by using direct search methods with local metamodeling techniques even when the regularity conditions are not fully satisfied.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Simulation Optimization and Methods

Simulation optimization, also known as simulation-based optimization, or optimization for simulation, can be defined as the process of finding the best input variable values from among all possibilities without explicitly evaluating each possibility(Carson and Maria, 1997). Simulation optimization is widely applied and can be found in many fields including finance, aerospace engineering, call center designs, oil detection, and medical treatment (Fu and Hu (1995); Semini et al. (2006); Rani and Moreira (2010); Carson and Maria (1997); April et al. (2003)). As its name suggests, simulation optimization can be considered an integration of classic optimization techniques and simulation analysis. The corresponding objective function is an associated measurement of an experimental simulation, which could be either physical or computer-based. Due to the lack of mathematical expression of the objectives, algorithms for simulation optimization problems are implemented based on the output measure obtained from the simulation program, while in the standard optimization scheme, algorithms rely more on the mathematical information provided by the objective function. Figure 1.1 below illustrates the process of simulation-based optimization.

Simulation optimization problems can be classified into various categories depending on the different criteria used. Depending on whether the simulation responses are deterministic or affected by noise or uncertainties, simulation optimization can be divided into deterministic cases and stochastic cases. Simulation optimization methods have also been applied to applications with

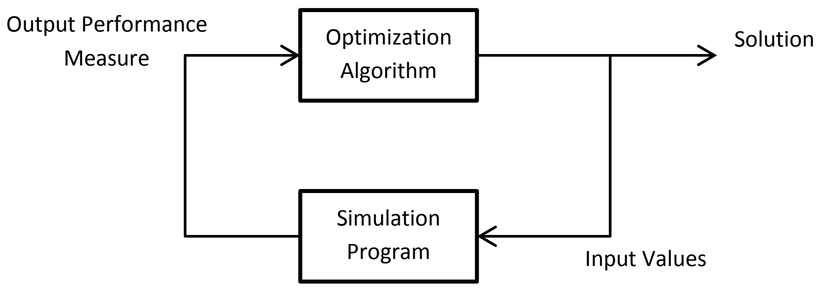


Figure 1.1: Simulation optimization process (Lacksonen, 2001)

a single objective, applications that require the optimization of multiple criteria, and even applications with non-parametric objectives (Carson and Maria, 1997). According to the nature of decision variables, simulation problems can likewise be classified as discrete state space problems or continuous state space problems.

It is worth mentioning that even in deterministic cases, simulation optimization problems are more difficult to solve compared with ordinary deterministic optimization problems. Some reasons that simulation-based optimization problems are difficult to deal with are highlighted by Banks (2005, p. 488) and Kleijnen et al. (2010). In summary, one of the most important causes is that under the simulation schemes, little information about the structure of the objective function is available. As a result, derivative information, which usually plays a critical role in the ordinary optimization algorithms, cannot be applied directly. Another crucial factor is that simulation programs might be so computationally expensive that a single run may take ten days ((Kleijnen et al., 2010)). Therefore, these two issues are typically among the top concerns during the development of algorithms for simulation optimization problems.

In this study, we investigate some enhanced methods and techniques to deal with a type of *continuous* and *stochastic* simulation optimization problems with *a single objective*.

Many algorithms and techniques have been developed for solving continuous simulation optimization problems. In addition, there are many excellent review papers on the subject of simulation-based optimization methods, such as Fu (1994); Banks (1998); Andradóttir (1998); Swisher et al. (2000); Fu (2002); Gosavi (2003).

One popular strategy to deal with the randomness in the stochastic schemes is sample average approximation (SAA) methods, also referred to as sample-path optimization. SAA methods use simulation to generate one sample path, and then obtain an approximate deterministic optimization problem (SAA problem) based on the sample path. After the deterministic approximation is obtained, any appropriate algorithm can be applied on it to get a solution, which will be considered as an estimator of the solution to the true problem.

There are many available algorithms can be applied on the deterministic problem that SAA methods generate. In a general standard, they can be classified into gradient-based methods and derivative-free methods, based on whether the algorithm requires derivative information or constructs an approximation of it.

Gradient-based methods estimate the gradient of the response function using a set of gradient estimation skills to measure the shape of the objective function and then employ some ordinary deterministic optimization algorithms to solve the problems (Carson and Maria, 1997). Gradient-based simulation optimization procedures have attracted a large amount of research attention over the past decade, due largely to the enormous amount of research attention also given to gradient estimation techniques. Some of these estimation techniques include finite difference (FD) estimation, likelihood ratio estimation, perturbation analysis (PA) and frequency domain experiments (PDE) (Fu, 1994). However, there are cases where none of these approaches works because, for a variety of reasons, derivative information is unavailable or unreliable in many cases. Consequently, derivative-free methods have always been needed.

Derivative-free methods seek to solve problems with no derivative information assumed or estimated. Two typical examples of such deterministic methods are trust region methods and direct search methods. The idea of trust region methods is to construct and optimize a local quadratic model iteratively within a ‘trust region’. The size of this region is modified during the process, depending on how well the local model agrees with the true function evaluations. More details of the trust region methods can be found in Conn et al. (2000). Direct search methods also work iteratively, but instead of building local models, the algorithms visit a set of candidates along a positive-spanning direction set and decide where to move based only on the responses obtained from the candidates. For a quick and recent review of derivative-free algorithms and comparison of software implementations, we recommend Rios and Sahinidis (2012). In addition, Vicente and Custódio (2012) introduce derivative-free optimization comprehensively in their work with a thorough statement of the art of derivative-free optimization methods, and a detailed description of the theory.

This study tries to propose a variant version of SAA methods to specifically solve a type of probability simulation optimization problems. Direct search methods, one of the derivative-free methods designed for deterministic simulation optimization problems, are proposed to work together with the modified SAA methods due to the methods' prominence, frequency of use, and appealing properties, which are highlighted next.

1.2 Direct Search Methods

Direct search methods are a class of deterministic optimization methods designed particularly for black-box optimization problems with continuous input space. The algorithms proceed iteratively, evaluate the objective functions at a finite number of points at each iteration, and decide which actions to take next exclusively based on those function values (or simulation responses) without any explicit or implicit derivative approximation or model building.

Direct search methods were first developed in the 1950s (Lewis et al., 2000), and remain popular with practitioners in the scientific and engineering communities, largely because they are straightforward to implement and do not require derivatives. However, they were not given much attention by the mathematical optimization community until the early 1990s, due to their lack of convergence proof (Swann, 1972). The situation was changed with the publication of a paper about convergence analysis on the direct search methods in 1991 (Torczon, 1991).

With the sophisticated development of gradient-based methods in recent years, direct search methods now appears less appealing. Many refined algorithms and software of gradient-based methods are available and easy to use, with diverse options to generate approximations to the derivatives. Furthermore, users can compute derivatives effortlessly with the help of automatic differentiation tools. There are also modeling languages (Brooke et al., 1996; Fourer et al., 1993) that can compute the gradient and/or the Hessian information automatically. In fact, to solve an unconstrained problem for which accurate first derivatives can be obtained, Kolda et al. (2003) recommend a gradient-based method the first choice, rather than direct search methods.

In spite of this, direct search methods are still needed for a few reasons relating to their nature. The first reason is that direct search methods remain an effective option, and are sometimes the only option for some types of scenarios. One typical and intuitive example is non-numerical functions, whose output values are not numerical in nature. Users compare objective values and decide which is better in terms of qualitative measure instead of quantitative data, which implies that gradient-based methods cannot work here as there is no gradient to use in any way. Another situation, which is happening more often in real practice, is that of gradient approximation techniques sometimes failing to give a reliable gradient estimation, in which cases gradient-based methods might fail to work well or even collapse.

Another important reason for the continuing relevance of direct search methods is that a large number of them can provide guarantees of convergence. Although direct search methods were originally designed for unconstrained optimization problems of a real-valued function, many extensions for constrained optimization problems have recently been developed. In fact, direct search methods can nowadays reach global convergence under mild conditions for a wide range of problems. More details will be reviewed in Chapter 2.

Therefore, there are conclusive reasons to believe that direct search methods will remain in favor with the simulation optimization community, including both mathematicians and practitioners.

1.3 Motivations and Objectives

The main aim of this study is to work on direct search methods to improve their performance efficiency and to extend the scale of their application in the field of simulation optimization. The study involves both stochastic and deterministic cases.

For the stochastic cases, it is mentioned in Section 1.1 that one typical way to deal with stochastic simulation optimization problems is the SAA method, which approximate the true problem by constructing a deterministic function based on a sequence of samples, and then employing an appropriate deterministic algorithm to solve the approximation problem. However, in the process of approximation, some structures of the original problem, such as continuity or differentiability, may be changed, making deterministic algorithms cannot work well on the SAA problems. One type of problems that is dealing with this issue and also widely seen in practice is a family of optimization problems with a probability objective function.

Probability functions can be also considered as the expectation measure of a random indicator function. Under the schemes of simulation optimization, the closed form of the true objective is not available, yet the distribution of the random factors is supposed to be known. According to the idea behind the SAA methods, the original probability objective can be approximated by averaging a sample of indicator values. However, due to the properties of indicator functions, the SAA problem is discontinuous as long as the sample size used is finite. Furthermore, since the possible values of the SAA function are limited by the sample size used in the approximation, it can be deduced that when the sample size is small, there is a high chance that large areas of flat regions exist. If a local convergent deterministic algorithm is applied to such a SAA problem, it can easily get trapped in the flat region and returns a wrong solution.

To help deterministic algorithms that exhibit a local convergence to deal with this issue, we propose two improved SAA methods for probability optimization problems, which will enable the local convergent algorithms to work much more smoothly even when the sample size is small. The first proposed method combines smoothing techniques with the idea behind the standard SAA methods, and the method is further improved in the second work by combining with the framework of retrospective-approximation algorithms. Under mild conditions, the solution set of the proposed methods converges to the counterpart of the true problem.

The two enhanced methods will provide new options to users who are dealing with expensive simulation-based probability optimization problems. Due to the smoothing tricks introduced here, the performances of the proposed methods are noticeably better in our numerical experiments. Furthermore, the dominant position of the proposed methods tends to be more obvious as the sample size decreases, implying that a large number of function evaluations can be saved.

In addition to the stochastic cases, we also attempt to improve direct search methods under the deterministic schemes. Driven by the need to optimize extremely expensive simulation-based problems, we propose an enhanced direct search method in the deterministic cases which makes extra use of visited points to obtain a picture of the local region. Combined with local metamodeling techniques, the enhanced algorithm makes a more flexible and intelligent decision about where to move first. Due to its capability to select a more promising candidate wisely, the proposed algorithm may find optimal solutions more quickly, and hence helps to reduce the simulation budget. The adjusted algorithm retains all the merits of the well established direct search methods without affecting the convergence results they exhibit, and therefore the quality of the obtained solution is not sacrificed.

We are also interested in the performance analysis from a theoretical point of view, and hence the study also mathematically examines the assistance provided by the local metamodels. The performance analysis provides insights into whether metamodels can really improve the efficiency of direct search methods, and how much exactly they can do. Due to different needs of real applications, two performance measures are proposed to quantify the performance of new methods from an asymptotic view and a finite time view respectively. The proposed algorithm is tested on a set of variants test problems, and numerical experiments arrive at the supporting conclusion that evaluations can be saved by using the adjusted algorithm.

1.4 Dissertation Outline

Chapter 2 provides a comprehensive review of two types of simulation optimization methods that are closely related to this thesis, namely direct search methods and sample average approximation (SAA) methods. Research gaps that exist between the up-to-date literature and the practical requirements are elaborated on, and suggest the motivations of this study.

In Chapter 3 a smoothed SAA method is proposed to smoothly approximate probability optimization problems under the simulation schemes and it is shown that asymptotically the solution set of the smoothed problem converges to the counterpart of the true problem. By introducing smoothing techniques to a SAA probability function, we should be able to prevent deterministic optimization algorithms that exhibit a local convergence from getting trapped in a flat region. Based on the numerical results, the smoothed SAA method significantly decreases the chance of the algorithm failing to work, and the proposed method also perform fairly well when the sample size is small.

The smoothed SAA method is extended in Chapter 4 with the introduction of a retrospective technique. The developed method, named retrospective smoothing SAA (RS-SAA) method, optimizes a sequence of smoothed SAA problems with an increasing sequence of sample sizes and decreasing sequences of smoothing parameters and mesh tolerances. By using the retrospective framework, the RS-SAA method can start from a very small sample size and automatically increase the size of samples once a solution of a certain accuracy is obtained. Given the convergence results in Chapter 3, similar results can be obtained for the RS-SAA method. The selection of parameters are discussed and numerical tested as well.

Chapter 5 proposes a framework for introducing local metamodels to a class of direct search algorithms in deterministic optimization problems. The proposed algorithm aims to improve the efficiency of the original standard direct search methods without affecting the convergence results. The framework of the improved algorithm is presented. We also discuss the conditions of the local metamodels and the objective functions under which the benefit of using metamodels is guaranteed. Numerical results with a wide variety of test problems show that using local metamodels can help to save simulation evaluations even when the regularity conditions are not fully satisfied.

Chapter 6 summarizes the work of this thesis and discusses several directions for further research.

Chapter 2

Literature Review

2.1 Direct Search Methods

For simplicity, we restrict our attention in this section to the following optimization problem:

$$\min_{x \in \mathscr{X}} f(x),$$

where $f : \mathbb{R}^d \to \mathbb{R}$, and $\mathscr{X} \subset \mathbb{R}^d$ is a given feasible set. We assume that f is continuously differentiable, but the closed form of it and the information about the gradient of f is either unavailable or unreliable.

2.1.1 Direct Search Methods in Early Stage

The first work using the idea of direct search methods dated back from Box (1957), who outlined a heuristic procedure called *evolutionary operation* to solve the problem of improving industrial processes and the shortage of technical personnel. Evolutionary operation relied only on simple designs and comparison of observed function values. This work also significantly defined a clear distinction between the direct search methods and response surface methodology, which was firstly proposed by Box and Wilson (1951). The phrase 'direct search' did not appear for the first time until in 1961 in the work of Hooke and Jeeves (1961). The authors defined direct search methods as "sequential examination of trial solutions involving comparison of each trial solution with the 'best' obtained up to that time together with a strategy for a determining what the next trial solution will be", which highlighted the core attribute of direct search methods even is valid today.

During the following decade, direct search methods were actively developed and remained popular with users in the scientific and engineering field because they do work well in practice though they are based on heuristics. Lewis et al. (2000) distinguished the variants in this period into three types, which are: *pattern search methods*, *simplex methods*, and *methods with adaptive sets of search directions*. Though this classification only focused on the classical direct search methods during the period of 1960-1971, most of the wide variants of algorithms developed subsequently can still be considered modifications of the three basic structures.

The work of Davidon (1991), which was originally published in 1959, provided an early form of pattern search algorithms. An abstract definition of the algorithms for solving nonlinear unconstrained optimization problems was firstly introduced by Torczon (1991). The basic idea of simplex methods, which belong the second category of direct search methods, is from Spendley et al. (1962), who constructed a nondegenerate simplex in $\mathbb{R}^d$ (a set of $d+1$ points in $\mathbb{R}^d$) and used the simplex to lead the move. Nelder and Mead (1965) improved the simplex search by adding expansion, contraction, and shrink move on the simplex to accelerate the search process, which has gained enduring success. As for the last category of direct search methods, methods with adaptive sets of search directions, as the name suggests, construct direction sets iteratively using the curvature information of the objective that is obtained before to accelerate the search. Several representative works belong to this family include Rosenbrock (1960); Powell (1964); Zangwill (1967).

Though these early developments have established a solid stage for their following achievements, direct search methods had not been recognized by mathematicians until by 1971, when a proof of first order convergence for a simplified version of pattern search methods came to light (Polak, 1971). It proved that a method named '*method of local variations*' converges to a stationary point under the assumption that $f(x)$ is at least once continuously differentiable. In the same year, using a stronger assumption, Cea et al. (1971) showed the convergence result of Hooke and Jeeves (1961)'s pattern search algorithm. Polak (1971) and Cea et al. (1971) are two important works in the justification of direct search methods, not only because they dispelled the popular view in that time that direct search methods are just heuristic approaches with no convergence derived, but also because they successfully identified the essential ingredients required for the general convergence theories of direct search methods (Lewis et al., 2000). Many recent works extended the theory framework of Polak (1971) and Cea et al. (1971). One of the representatives is performed by Torczon (1997), who relaxed some conditions required for the convergence, and hence made the convergence conclusion be more general to cover more cases.

On the other hand, compared with the convergence works on pattern search methods, Nelder-Mead simplex methods do not show a robust performance in terms of convergence despite of the fact that they can work pretty well most of the time. All we have is that Nelder-Mead can be guaranteed to converge to a stationary point under mild conditions in $\mathbb{R}^1$ (Lagarias et al., 1998). As a matter of fact, McKinnon (1998) made a final conclusion that proving first order convergence for the Nelder-Mead methods in higher dimensions is impossible. Due to this reason, we will focus our subject of this study on the other two categories of the direct search methods: pattern search methods and methods with adaptive set of search directions. Some newly developed algorithms derived from these two ancestors are reviewed next.

2.1.2 Recent Variants: GPS and MADS

Two types of direct search methods that are most popular nowadays are generalized pattern search (GPS) methods and mesh adaptive direct search (MADS) methods. As the names suggest, they are correspondingly modified from the two basic themes (patterns search methods and methods with adaptive set of search directions) have been reviewed in the previous part. Before going in depth into each of them, some common features they share are highlighted.

Both GPS and MADS have two stages at each iteration, named a SEARCH stage and a POLL stage. The SEARCH stage is flexible and optional. It allows evaluations of objectives at any finite number points to find a better objective value. There is no specific rule of points selection as long as their number remains finite. The cost of its flexibility and optionality is that the SEARCH stage cannot be used in the convergence analysis. The convergence is guaranteed by the POLL stage, which is implemented after the SEARCH only when the SEARCH fails to find a better point. The POLL stage is more like a local search. It visits a set of candidates locating along a positive-spanning direct set, and decides what to do next only based on the function values. A positive-spanning set in $\mathbb{R}^d$ means that any nonnegative linear combinations of its elements must span $\mathbb{R}^d$.

As a matter of fact, the difference between the GPS and the MADS lies exactly in the POLL stage (Audet and Dennis Jr, 2006). More specifically, they have different mechanisms to generate the direction set at each iteration. The directions used by MADS are selected in a way so that asymptotically they are not confined to a finite set, whereas for GPS, the number of direction sets can be formed is constant over all iterations.

In the development of direct search methods, much attention has been placed on exploring the types of problems on which direct search methods can work with a guarantee of convergence. As a matter of fact, a wide range of problems can be solved by sophisticated direct search methods with a global convergence currently. By saying global convergence in the nonlinear programming literature, we mean first-order convergence from an arbitrary starting point.

The framework of GPS is firstly introduced by Torczon (1997) for unconstrained optimization problems. Lewis and Torczon (1999) developed the GPS algorithms to bound constrained optimization. Later, the framework of GPS was improved to solve linearly constrained optimization problems (Lewis and Torczon, 2000), and more generally for problems with nonlinear constraints (Audet and Dennis Jr, 2004). Liu and Zhang (2006) designed a GPS framework for linearly equality constrained optimization problems. Bogani et al. (2009a) proposed a GPS method to solve a class of highly structured nonsmooth minimization problems. In the same year, Bogani et al. (2009b) proposed a generating set search method for solving nonsmooth optimization problems where the objective function is locally Lipschitz continuous and piecewise continuously differentiable on a finite family of polyhedra. Sriver et al. (2009) extended GPS for mixed variable optimization with stochastic objective functions. More recently, Custódio et al. (2011) proposed a direct multisearch (DMS) methodology for multiobjective optimization and they proved that at least on limit point of the sequence of iterates generated by the novel method lies in a stationary form of the Pareto front. Vicente and Custódio (2012) contributed analysis of direct searches for discontinuous functions, showing that contribute Rockafellar derivatives are also nonnegative along the limit directions of those subsequences of unsuccessful iterates when the function values converge to the function value at the limit point.

As for the MADS, Audet and Dennis Jr (2006) first introduced it for achieving global convergence in the nonsmooth case. MADS is able to converge to a poinst where the Clarke generalized directional derivative is nonnegative for a set of directions dense in $\mathbb{R}^d$, which is different from the result of GPS that is just for a finite set of directions. Later, Audet et al. (2008) proposed a way to combine MADS with the Variable Neighborhood Searchmetaheuristic (VNS) for nonsmooth constrained optimization. The convergence of MADS is retained while the far reaching exploration features of VNS to move away from local solutions is obtained as well. Audet et al. (2010) studied a mesh adaptive direct search algorithm for nonsmooth functions subject to general constraints.

There are many studies about practicing direct search methods in real applications as well, which cover the field of power system, chemical engineering, biology and many others (Al-Sumait et al., 2007; Güneş and Tokan, 2010; Hosseini et al., 2011; Lee et al., 2011).

All the works mentioned above are for deterministic cases. A few works have been done to extend direct search methods to stochastic cases as well. Anderson and Ferris (2001) considered the case where the responses obtained is affected by a white noise and it gave the conditions under which direct search could still work and guarantee a first-order convergence. Kim and Zhang (2010) adapted a type of direct search methods for standard stochastic optimization problems, and gave the conditions for convergence using a variable-number sampling scheme.

2.1.3 GSS Methods

As mentioned in Section 2.1.2, for both GPS and MADS, the SEARCH stage is flexible and optional, and it does not help to reach a global convergence. Focusing only on the POLL stage, the framework of GPS and MADS can be both considered derivatives from a classic direct search method named generating-set search (GSS) method introduced by Kolda et al. (2003), whose framework is presented in Figure 2.1. Without further notice, we assume an empty SEARCH stage by default for our work coming afterward and the notations used in Figure 2.1 will be reserved through out this thesis.

```
Initialization:
  For given $f : \mathbb{R}^d \to \mathbb{R}$, set $k = 1$, and initialize:
    • $x^{(k)}$ as the starting point
    • $\Delta^{\text{tol}}$ as the mesh size, or step-length tolerance to terminate the algorithm
    • $\Delta^{(0)} > \Delta^{\text{tol}}$ as the initial mesh size
    • $\rho : [0, +\infty) \to \mathbb{R}$ as a continuous forcing function such that $\rho(t)/t \to 0$ as $t \downarrow 0$, or $\rho \equiv 0$.
Ensure:
 1: for each iteration $k = 1, 2, \cdots$ do
 2:   generate a direction set $D^{(k)}$ under some technical conditions;
 3:   if $\exists d^{(k)} \in D^{(k)}$ such that $f(x^{(k)} + \Delta^{(k)} d^{(k)}) < f(x^{(k)}) - \rho(\Delta^{(k)})$ then
 4:     set $x^{(k+1)} = x^{(k)} + \Delta^{(k)} d^{(k)}$;
 5:     set $\Delta^{(k+1)} = \gamma^{(k)} \Delta^{(k)}$, where $\gamma \geq 1$ denotes an expansion factor;
 6:   else
 7:     set $x^{(k+1)} = x^{(k)}$;
 8:     set $\Delta^{(k+1)} = \theta^{(k)} \Delta^{(k)}$, where $\theta < \theta_{\max} < 1$ denotes a contraction factor;

 9:     if $\Delta^{(k+1)} < \Delta^{\text{tol}}$ then
10:       terminate the algorithm ;
11:     end if
12:   end if
13:   $k \leftarrow k + 1$;
14: end for
```

Figure 2.1: A GSS method

2.1.4 Improvements on Efficiency

It can be seen from Section 2.1.2 that mathematical communities have put lots of effects into the study and develop the convergence performance of direct search methods. By now, as a matter of fact, for a large number of direct search methods, it is possible to provide rigorous guarantees of convergence (Kolda et al., 2003, section 1.3.1).

However, one issue concerned by the users of direct search methods, other than the people in the mathematical community, is that this asymptotic convergence might be slow (Kolda et al., 2003, section 1.3.1). In fact, the truth in the real simulation problems is that many simulations are so expensive that a more efficient algorithm which can obtain a solution with less simulation consumed is much preferred. However, to the best of our knowledge, there has not been many efforts in trying to develop efficient implementations of direct search methods.

One of the earliest and also most developed trend is mentioned by Lewis et al. (2000), that is to use adaptive sets of search directions instead of fixed ones. Another trend to improve efficiency is to let the algorithm run parallel (Hough et al., 2002; Regis and Shoemaker, 2010). However, this method focuses on the saving in the running time, rather than the essential simulation evaluations that are assumed. Abramson (2005) looked at the cases where some incomplete form of gradient information is available. He showed that the information can be used to save some function evaluations. Andradóttir and Prudius (2009) studied GPS and proposed a framework to maintain an appropriate balance between global search, local search, and estimation. The numerical experiment results of these two works were good, but little mathematical justification was provided on their proposed methods.

Custódio and Vicente (2007) used simplex derivatives in the pattern search methods to reorder the sequence to visit candidate points and the work has been extended into nonsmooth functions by Custódio et al. (2008). Both works focus on the deterministic cases. Though the effect of the improvement is analyzed, the mathematical conditions required for the objective functions are not stated. Furthermore, the numerical results presented in the work were not clearly supportive. Custódio et al. (2010) used quadratic underdetermined metamodeling in the SEARCH stage of GPS and showed good performance with some numerical tests. Still, the mathematical justification of the proposed algorithms was not discussed.

2.2 Sample Average Approximation (SAA) Methods

2.2.1 Methodologies and Variants

Sample average approximation (SAA) methods, also referred as Monte Carlo sampling methods (Lasdon and Popova, 2005; Shapiro, 2003; Homem-de Mello and Bayraksan, 2013), Monte Carolo simulation approaches (Rubinstein and Kroese, 2011), or sample-path methods (Robinson, 1996), are one of the classic methods to deal with stochastic simulation optimization problems. The reason that SAA methods stay popular in the field of stochastic simulation optimization in that they are easily understood and implemented, and also can be surprisingly efficient for some classes of stochastic programming problems. As a matter of fact, in many practical applications, SAA may be the only reasonable way of dealing with the objective function (Shapiro, 2001).

Consider the optimization problem

$$\min_{x \in \mathscr{X}} f(x) := E\left[F(x, \boldsymbol{\xi})\right], \tag{2.1}$$

where $F : \mathbb{R}^d \times \Xi \to \mathbb{R}$, the expectation is taken with respect to a probability measure defined on a sample space $(\Xi, \mathscr{F})$, and $\mathscr{X} \subset \mathbb{R}^d$. Assume that for every $x \in \mathscr{X}$ the expectation $f(x)$ is well defined.

In the scenario of simulation optimization, the closed form of the objective function $f(x)$ is not available, neither can $f(x)$ be observed or computed directly. However, we suppose that the distribution of the random element $\boldsymbol{\xi}$ is known and it does not depend on x. therefore a sample of N realizations of $\boldsymbol{\xi}$ can be generated. In SAA, a sequence of samples $\{\xi_1, \xi_2, \cdots, \xi_N\}$ is first generated from the same distribution of $\boldsymbol{\xi}$. With the samples, one can estimate the true expectation function $f(x)$ by the corresponding sample average function:

$$\bar{f}_N(x) := \frac{1}{N} \sum_{i=1}^{N} F(x, \xi_i). \tag{2.2}$$

Once the samples are realized, the function (2.2) becomes deterministic, and any proper deterministic optimization algorithms could be applied to solve the SAA problem

$$\min_{x \in \mathscr{X}} \bar{f}_N(x).$$

It can be seen that SAA is essentially not an algorithm itself; it refers to a method to approximate the original objective problem by using a sample average function.

Though the origin of the approach is difficult to point, some variants of the SAA method were suggested by various researchers over the year attribute to the simplicity of the idea behind. Geyer and Thompson (1992) employed Monte Carlo techniques based on Gibbs sampling to compute Maximum Likelihood estimates. The similar idea was also suggested in the work of Rubinstein and Shapiro (1993). SAA methods are employed in the stochastic programs in a wide range of applications such as finance (Rockafellar and Uryasev, 2002; Alexander et al., 2006), engineering design (Rockafellar and Royset, 2010), inventory management (Xu and Zhang, 2009), power system planning (Linderoth et al., 2006) and many others.

One of the variants of SAA methods that relates to this thesis is a type of sequential sampling approach. Homem-de Mello and Bayraksan (2013) described this method as 'an iterative approach whereby the optimization alternates with the sampling procedure'. The idea behind employing the sequential sampling approach is to save sampling effort in the early stage of the searching, when the current solution is far from the optimal solution, and increase the sample size as the solution approaches closer to the optimal one. Driven by this rationale, Chen and Schmeiser (2001) proposed the Retrospective Approximation (RA) method for a general type of stochastic root finding problems. RA solves a sequence of root finding problems with an increasing sequence of sample size and a decreasing sequence of solution tolerance. Based on the same need of saving evaluations, Deng and Ferris (2009) developed a variable-number method to choose appropriate number of samples by using Bayesian analysis techniques at each iteration. Pasupathy and Schmeiser (2009) developed a family of RA algorithms named Bounding RA to solve a certain type of multidimensional stochastic root finding problems, and study the rate of convergence and choice of parameters of RA methods comprehensively in the work Pasupathy (2010).

In this thesis, we will describe a sequential sampling approach using the similar framework of RA but in the context of simulation optimization.

2.2.2 Theoretical Properties

One of the important topics in the theoretical study of SAA is to investigate the conditions under which the optimal solutions of the SAA problem (2.2) converge to the set of optimal solutions of the true problem (2.1) as the sample size N grows. Normally this result is conditioned on the uniform convergence of the SAA function $\bar{f}_N(x)$. Let S denote the set of optimal solutions of the true problem (2.1) and $\bar{S}_N$ denote the set of optimal solutions of the SAA problem (2.2), then we state a converge result based on uniform convergence from Shapiro

(2003, Proposition 6) in the following.

Proposition 1. *Suppose that there exists a compact set $C \subset \mathbb{R}^d$ such that:*

1. *the set S of optimal solutions of the true problem is nonempty and is contained in C,*
2. *the function $f(x)$ is finite valued and continuous on C,*
3. *$\bar{f}_N(x)$ converges to $f(x)$ w.p.1, as $N \to \infty$, uniformly in $x \in C$,*
4. *w.p.1 for N large enough the set $\bar{S}_N \subset C$.*

Then $\mathbb{D}(\bar{S}_N, S) \to 0$ w.p.1 as $N \to \infty$, where $\mathbb{D}(A,B) := \sup_{x \in A} dist(x,B)$, and $dist(x,B) := \inf_{x' \in B} \|x - x'\|$.

The third condition in Proposition 1 is a uniform version of the strong law of large numbers (Kim, 2006), and the proof of the proposition can be checked in Shapiro (2003, Proposition 6). The result of Proposition 1 will be used later in this thesis.

The following result by Shapiro (2003, Proposition 7) is cited without proof to show a set of sufficient conditions under which the uniform assumption in Proposition 1 can be ensured.

Proposition 2. *Suppose that for almost every $\xi \in \Xi$,*

1. *the function $F(x,\xi)$ is continuous on $\mathscr{X}$, and*
2. *$F(x,\xi)$ is dominated by an integrable function.*

Then $f(x)$ is finite valued and continuous on $\mathscr{X}$ and $\bar{f}_N(x)$ converges to f uniformly on $\mathscr{X}$ as $N \to \infty$ w.p.1.

In addition to the convergence on the solution set, Glasserman (1990) provided necessary and sufficient conditions to ensure $\nabla \bar{f}_N(x)$ converges to $\nabla f(x)$ as N goes to infinity. Glasserman (1990) also pointed that when $f(x)$ is an expectation measure, the most popular method to make $\nabla \bar{f}_N(x) \to \nabla f(x)$ hold is through Lebesgue's dominated convergence theorems combining with the generalized mean value theorem (Pasupathy, 2010).

There are many excellent review works on the theoretical properties of the SAA methods. Shapiro (2003); Shapiro et al. (2009) gave a comprehensive discussion on various properties of the SAA methods, including rates of convergence and complexity of the methods. Kim et al. (2011) reviewed the technical conditions in detail under which SAA methods are appropriate to use in terms of the quality of the solution. The most recent review work, to the best of our knowledge, is from Homem-de Mello and Bayraksan (2013), who surveyed the issues in use of SAA methods such as the optimality conditions, choosing appropriate sample sizes and many others.

Chapter 3

Smoothed SAA Methods for Probability Optimization Problems

3.1 Introduction

In this chapter, we focus on a certain type of probabilistic programming problems in the field of simulation optimization. Probabilistic programming normally refers to two strongly connected models: minimizing (or maximizing) a probability under constraints, and programming under probabilistic constraints (Prékopa, 2003). The subject of this study falls in the first category.

The optimization problems with an probability objective function were extensively investigated by Prékopa (1988) and Robbins and Monro (1951). This form of objective function is widely used in many fields such as finance, engineering, aviation and health care (Rockafellar and Royset, 2010). Due to the lack of closed form of the probability function, a sample average approximation (SAA) problem can be constructed and solved to obtain an estimate of the solution of the true problem. Sample average approximation (SAA) is one of the most popular methods to deal with stochastic optimization problems; however, since probability functions are essentially expected values of an indicator function, optimization algorithms exhibiting a local convergence cannot work well on them when the SAA methods are directly applied because the corresponding SAA functions are ill-structured.

We propose a smoothed SAA method, which incorporates smoothing techniques into the classic SAA methods, so that a wide range of nonlinear optimization algorithms can be successfully applied even when the sample size is small. Numerical results show that computational time is significantly saved when solvers are applied to the smoothed SAA problem compared with the case applied to the original standard SAA problem. In addition, we state that under certain conditions, the set of optimal solutions to the smoothed SAA problem converges to the counterpart of the true problem as sample size goes to infinity.

Consider a probability optimization program represented as follows:

$$\min_{x\in\mathscr{X}} f(x) := \Pr\Big\{c(x,\boldsymbol{\xi}) \geq 0\Big\}. \tag{3.1}$$

Here, $x \in \mathbb{R}^d$ is a decision vector, and $\mathscr{X}$ is a subset of $\mathbb{R}^d$. Unless stated otherwise we assume that the set $\mathscr{X}$ is $\mathbb{R}^d$ or a box constraint set, that is, $\mathscr{X} = [LB, UB]$, where LB, $UB \in \mathbb{R}^d$. $\xi \in \mathbb{R}^m$ is a vector of uncertain parameter, whose support is denoted as Ξ. In order to distinguish between random data and their numerical values, we will use bold script $\boldsymbol{\xi}$ for the random vector, and ξ for its particular realization afterwards. $c(x,\boldsymbol{\xi}) : \mathbb{R}^d \times \Xi \to \mathbb{R}$ is a real-valued function and we assume that $f(x)$ is continuously differentiable on $x \in \mathscr{X}$.

The investigation of the continuity of probability functions with respect to the decision variables is initially studied by Raik (1975). Later, Kibzun and Kan (1996) developed well-established theory on continuity and differentiability of probability functions. Some of the key results are presented first. The following proposition provides sufficient conditions on the continuity of a probability function.

Proposition 3. *Let the following conditions hold:*

1. *the loss function $c(x,\xi)$ is continuous at every point $x \in \mathscr{X}$ for almost all ξ;*
2. *$mes\{\xi : c(x,\xi) = 0\} = 0$ for every $x \in \mathscr{X}$, where $mes(A)$ denotes the Lebesgue measure of set A;*
3. *the random vector $\boldsymbol{\xi}$ has a probability density function $p(\xi)$.*

Then the probability function $f(x)$ is continuous with respect to x.

For a proof, see Lemma 2.12 in Kibzun and Kan (1996).

Before a set of sufficient conditions for the differentiability of a probability function is stated, some notations are defined first. Let $T(x) := \{\xi \in \mathbb{R}^m : c(x,\xi) \geq 0\}$, $\Delta T_{\mathscr{X}} := \mathrm{cl}\Big(\cup_{x\in\mathscr{X}} \partial T(x)\Big)$, where $\mathrm{cl}(A)$ is the closure of a set A,

$\Delta T_{\mathscr{X}\Xi} := \Delta T_{\mathscr{X}} \cap \Xi$, $\Delta G_{\mathscr{X}\Xi} := \mathrm{cl}(\mathscr{X}) \times \Delta_{\mathscr{X}\Xi}$, $B(x) := \{\xi \in m : c(x,\xi) = 0\}$, and $B_{\Xi}(x) := B(x) \cap \Xi$. The gradient of the probability function $f(x)$ can be obtained by the following proposition.

Proposition 4. *Let the following conditions hold:*

1. *the set $\Delta T_{\mathscr{X}\Xi}$ is bounded;*
2. *the vector functions $\nabla_x c(x,\xi)$ and $\nabla_\xi c(x,\xi)$ are continuous on the set $\Delta G_{\mathscr{X}\Xi}$;*
3. *the probability density function $p(\xi)$ is continuous on the set $\Delta T_{\mathscr{X}\Xi}$;*
4. *$mes_{m-1}\Big(\{\xi \in \partial\Xi : p(\xi) > 0\} \cap B(x)\Big) = 0$ holds for all $x \in \mathscr{X}$, where mes_{m-1} is the $(m-1)$-surface Lebesgue measure;*
5. *$\|\nabla_\xi c(x,\xi)\| > 0$ on the set $B_{\Xi}(x)$ for all $x \in \mathscr{X}$.*

Then the probability function $f(x)$ is differentiable for all $x \in \mathscr{X}$ and

$$\nabla f(x) = -\int_{B_{\Xi}(x)} \frac{\nabla_x c(x,\xi)}{\|\nabla_\xi c(x,\xi)\|} p(\xi)\, dT.$$

For a proof, see Lemma 2.29 in Kibzun and Kan (1996). For some other results concerning differentiability of probability functions, readers are referred to Kibzun and Kan (1996, Chapter 2).

Note that the objective function (3.1) can be reformulated as an expectation minimization problem, which is

$$\min_{x\in\mathscr{X}} f(x) = E\big[\Phi(x,\boldsymbol{\xi})\big] := E\big[\mathbf{1}_{[0,\infty)}c(x,\boldsymbol{\xi})\big], \tag{3.2}$$

where $\mathbf{1}_A(z)$ denotes the indicator function of set A that equals to 1 if $z \in A$ and 0 otherwise.

Throughout this chapter, we assume that the closed form of (3.2) is not available and hence have to be approximated by using sample average approximation (SAA) methods. By applying the SAA, a set of independent realizations $\xi_1, \xi_2, \cdots, \xi_N$ of $\boldsymbol{\xi}$ from a Monte Carlo simulation is generated and the true objective function (3.2) is approximated by

$$\min_{x\in\mathscr{X}} \bar{f}_N(x) = \frac{1}{N}\sum_{i=1}^{N}\big[\mathbf{1}_{[0,\infty)}c(x,\xi_i)\big] = \frac{1}{N}\sum_{i=1}^{N}\Phi(x,\xi_i). \tag{3.3}$$

Note that once the samples are generated, i.e., the numerical values of $\xi_1, \cdots, \xi_N$

are realized, $\bar{f}_N(x)$ becomes a deterministic function and its value can be determined at any given point $x \in \mathscr{X}$. We would refer to (3.1) and (3.2) as the true problem and (3.3) as the SAA problem hereafter.

Technically, any appropriate deterministic algorithm can be applied to (3.3) to obtain an optimal solution of the SAA problem, say x_N^*, which can be considered as an estimator of the solution to the true problem. Generally, the SAA problem (3.3) is approaching to the true problem when sample size N is large. Furthermore, under certain conditions, the set Π_N of optimal solutions to the SAA problem converges to the solution set Π to the true problem.

However, applying SAA methods to the probability optimization problem (3.3) is significantly different from the normal literature. Due to the binary-value nature of the indicator function $\mathbf{1}_{[0,\infty)}c(x,\boldsymbol{\xi})$, (3.3) is of a step-like form and consequently discontinuous whenever the sample size is finite. Furthermore, we have $\nabla \bar{f}_N(x) = 0$ for all x except discontinuity points, making any algorithms exhibiting a local convergence cannot work well as all the differentiable x has a gradient of zero. What makes things worse is that when the sample size is small, there might be a large area of flat region existing so that (nonlinear) optimization methods can easily get trapped in, with the belief that they have found the local optimal solution. In summary, we believe that the nature of (3.3) makes it difficult to apply (nonlinear) optimization methods directly on them. This issue is illustrated more clearly by the following example.

Example 1. *Consider solving a probability optimization problem with SAA methods and nonlinear optimization algorithm:*

$$\min_{x \geq 0} f(x) := \Pr\{x - \boldsymbol{\xi} \geq 0\}, \tag{3.4}$$

where $\boldsymbol{\xi} \sim Norm(0,1)$. Then, the optimal solution is $x^* = 0$.

Notice that the objective function (3.4) can be reformulated as $\min_{x \geq 0} E\left[\mathbf{1}_{[0,\infty)}(x - \boldsymbol{\xi})\right]$. By generating a sequence $\{\xi_1, \xi_2, \cdots, \xi_N\}$ of replications of $\boldsymbol{\xi}$, a SAA approximation deterministic problem is obtained, which is:

$$\min_{x \geq 0} \bar{f}_N(x) := \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{[0,\infty)}(x - \xi_i). \tag{3.5}$$

We present the shape of the SAA problem with different values of N in Figure 3.1. Due to the simplicity of the problem, we expect to construct a SAA

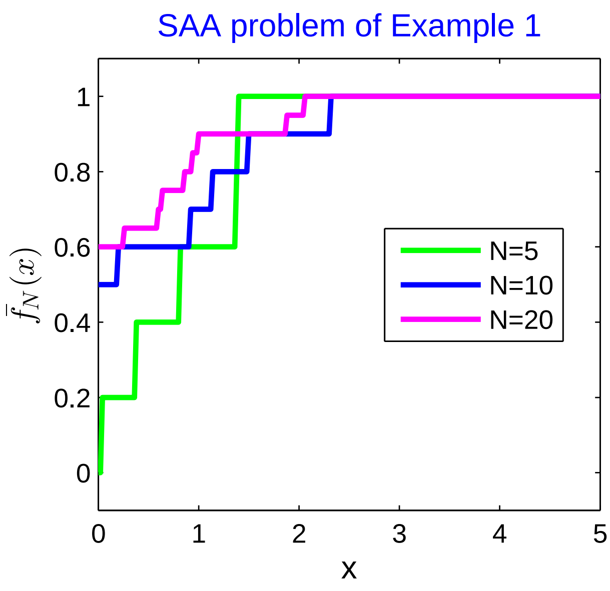


Figure 3.1: SAA problems of Example 1

problem with small sample sizes: N =5, 10 and 20. It can be seen that there are many jumps and flat regions presented in the SAA objective function, which might let users concern if (nonlinear) optimization methods can work effectively on them. In particular, we apply a direct search algorithm to solve the SAA problem. The result with $N = 20$ is present in Figure 3.2.

In this example, 10 starting points are generated from a uniform distribution in the range $[0,3]$. For each starting point, the direct search algorithm is implemented and a corresponding path of iterates x_k and accumulated consumed function evaluations are recorded. The ten paths are presented in Figure 3.2, where the x-axis denotes the total number function evaluations consumed and the y-axis denotes the distance between iterates x_k to the optimal solution $x^* = 0$.

It can be seen from Figure 3.2 that neither of the path converges to the true solution successfully. Some of the paths end up with a poor quality solution. Considering the simplicity of the example problem, it can be expected that the performance of the SAA methods will be deteriorated with more complex objective functions.

Example 1 also reveals the reason that SAA methods cannot be directly applied to probability functions. Since $\nabla \bar{f}_N(x) = 0$ for all x except discontinuity points, $\bar{f}_N$ is ill-structured for a local convergent algorithm. In this chapter, we

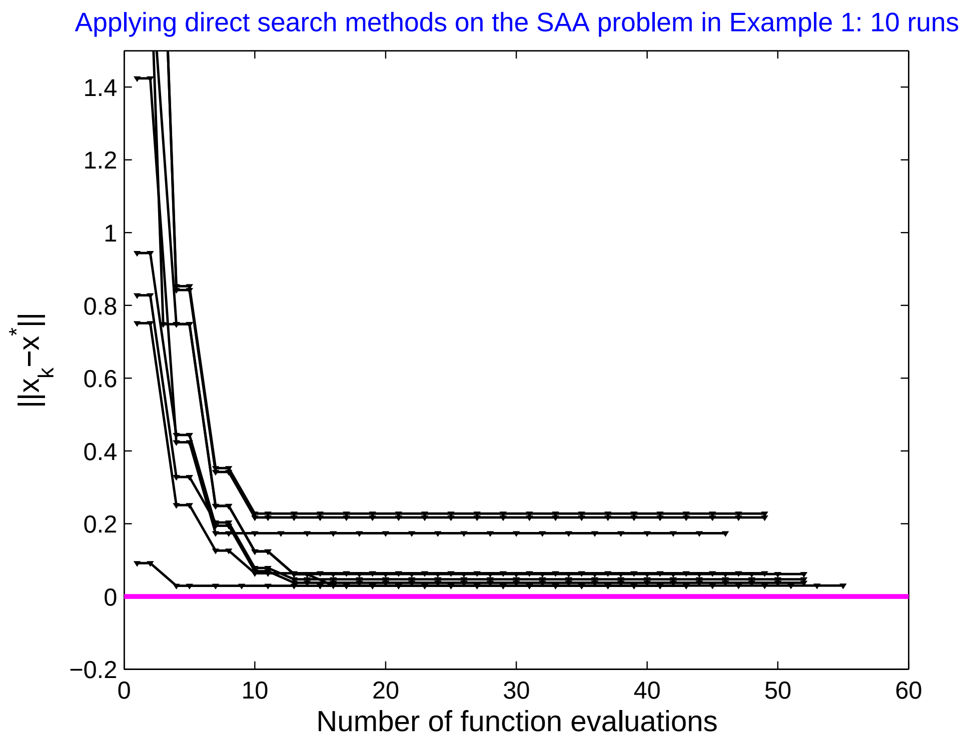


Figure 3.2: Applying a direct search algorithm on the SAA problems of Example 1

try to resolve this issue caused by the nature of indicator functions by introducing smoothing techniques to the standard SAA methods.

Smoothing techniques have been introduced to the field of optimization to make discrete problems more manageable. One trend that smoothing techniques have been used is for minimizing risk measures such as VaR and CVaR. Gaivoronski and Pflug (2005) presented a method, which is based on the approximation of historical VaR by smoothed VaR, to calculate the mean-VaR efficient frontier. Alexander et al. (2006) proposed a computational method based on smoothing techniques to solve a simulation based CVaR optimization problem efficiently. Inspired by the work of Alexander et al. (2006), Xu and Zhang (2009) studied a smoothing scheme for a class of Lipschitz continuous stochastic problems and investigated the convergence of stationary points of the smoothed SAA problems as sample size increases. The work proved that with probability one, the accumulation points of the stationary points of the approximation problem are weak stationary points of their counterpart of the true problem. Meng et al. (2011) proposed similar smoothing techniques to solve single

CVaR and mixed CVaR minimization problems, and proved that for any fixed smoothing parameter the method produces a sequence whose cluster point is a weak stationary point of the true optimization problems with probability one. To the best of our knowledge, few work has been done to combine smoothing techniques with probability optimization problem.

The main contributions of this work as far as we are concerned are as follows. First we incorporate smoothing techniques into the standard SAA methods to deal with probability simulation optimization problems. The smoothed SAA problem generated by our algorithms is well-structured so that a wide range of optimization algorithms can be successfully applied to them. In addition, direct search algorithms are employed to solve both the smoothed SAA problems proposed and classic SAA problems, and the results of the two cases are compared. It can be seen from the numerical results that our method exhibits a much better performance especially when the sample size is small. Last but not least, we show that under mild conditions, with probability one the optimal solution set of the smoothed SAA problems converges to the solution set of the true problems. Overall, the smoothed SAA methods provide a new option to the users who are dealing with expensive probability simulation optimization problems and they are fairly general so that a wide class of numerical algorithms can be incorporated with them as a solver.

The rest of this chapter is organized as follows. In Section 3.2, we discuss a smoothing technique and propose a smoothed SAA method for the probability optimization problem. In Section 3.3, we investigate the convergence performance on the solution set of the smoothed SAA problems. In Section 3.4, numerical experiments are carried out to illustrate the performance of smoothed SAA scheme. The work is concluded in Section 3.5.

3.2 Smoothed SAA Methods for Probability Optimization

We propose a smooth function $\hat{\Phi}(x,\xi,\varepsilon)$, where $\varepsilon > 0$ is an approximation parameter to approximate the standard indicator function $\Phi(x,\xi)$. In order to obtain well defined smoothed approximation problems, the following assumption is imposed.

Assumption 1. *The smooth approximation of the true indicator function can be expressed as*

$$\hat{\Phi}(x,\xi,\varepsilon) = \Psi(c(x,\xi),\varepsilon),$$

where $\Psi : \mathbb{R}\times\mathbb{R}_+ \to \mathbb{R}$ is a continuously differentiable real-valued function such that for any $z \in \mathbb{R}$,

a) $\lim_{\varepsilon\downarrow 0}\Psi(z,\varepsilon) = \mathbf{1}_{[0,\infty)}(z)$*, and*

b) $\mathbf{1}_{[0,\infty)}(z) \le \Psi(z,0) \le \mathbf{1}_{[-\varepsilon,\infty)}(z)$*, for any* $\varepsilon > 0$.

Furthermore, we assume the assumptions in Proposition 4.

It is pointed out that the conditions (a) and (b) of Assumption 1 are commonly used for a general definition of smoothing (Xu and Zhang, 2009; Kim et al., 2011).

By using the smooth function $\hat{\Phi}$ to replace the original indicator function, the objective function (3.2) is turned into the following approximation problem:

$$\min_{x\in\mathscr{X}} \; g(x,\varepsilon) := E\left[\hat{\Phi}(x,\xi,\varepsilon)\right]. \tag{3.6}$$

Appropriate numerical algorithms are employed to solve the SAA problem of (3.6) as follows:

$$\min_{x\in\mathscr{X}} \; \bar{g}_N(x) := \frac{1}{N}\sum_{i=1}^{N}\hat{\Phi}(x,\xi_i,\varepsilon(N)), \tag{3.7}$$

where the smooth parameter $\varepsilon(\cdot)$ is a function of N satisfying $\varepsilon(N) \to 0$ as $N \to \infty$.

One of the examples of Ψ functions satisfying Assumption 1 is the NEW_STEP function proposed by Vikram (1995) in his electronic technical report. As shown in Figure 3.3, the NEW_STEP function provides a mean of transitioning from a constant value h_1 to another value h_2 over a specified interval between two specified transition points x_1 and x_2.

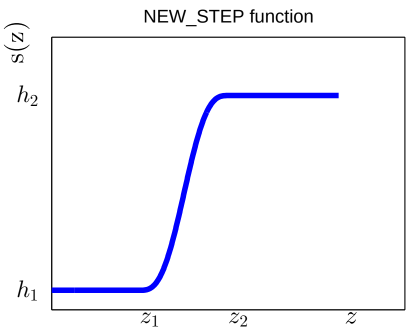


Figure 3.3: NEW_STEP function

Mathematically, the NEW_STEP function is defined as follows:

$$s(z) = \begin{cases} h_1 & , \; z < z_1; \\ s(z,x_1,h_1,x_2,h_2) & , \; z_1 \le z \le z_2; \\ h_2 & , \; z_2 > z; \end{cases}$$

where the transition function in the interval of z_1 and z_2 is in the form of

$$s(z, z_1, h_1, x_2, h_2) = h_1 + \frac{\Delta h}{\Delta x}(z - z_1) - \frac{\Delta h}{2\pi} \sin\left(\frac{2\pi}{\Delta x}(z - z_1)\right),$$

where $\Delta h = h_2 - h_1$, and $\Delta z = z_2 - z_1$.

By plugging the values $z_1 = -\varepsilon$, $z_2 = 0$, $h_1 = 0$, and $h_2 = 1$ in the NEW_STEP function, we have the smoothing function $\Psi(z, \varepsilon)$ of the following form:

$$\Psi(z,\varepsilon) = \begin{cases} 0 & , \quad z < -\varepsilon \\ \frac{1}{\varepsilon}(z+\varepsilon) - \frac{1}{2\pi}\sin\left(\frac{2\pi}{\varepsilon}\cdot z\right) & , \quad -\varepsilon \le z \le 0, \\ 1 & , \quad z > 0. \end{cases}$$

The plot of $\Psi(z, \varepsilon)$ with $\varepsilon = 1$ and $\varepsilon = 0.5$ is presented in Figure 3.4. This

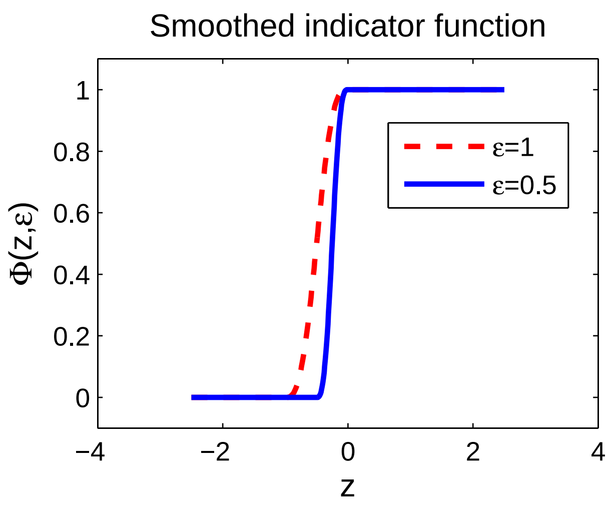


Figure 3.4: Smoothed indicator function $\hat{\Phi}(x, \xi, \varepsilon)$

form of smoothing function will be used in the numerical experiments coming afterwards.

A preview of the corresponding smoothed SAA function in Example 1 is presented in Figure 3.5. Though an approximation error is introduced by the positive ε, it successfully resolves the ill-posed structure of the original SAA problems and helps optimization algorithms applicable. It can be seen from Figure 3.5 that even when sample size is small ($N = 5$), with the help of introducing ε, the smoothed function $\bar{g}_N$ gives a more smooth and accurate picture

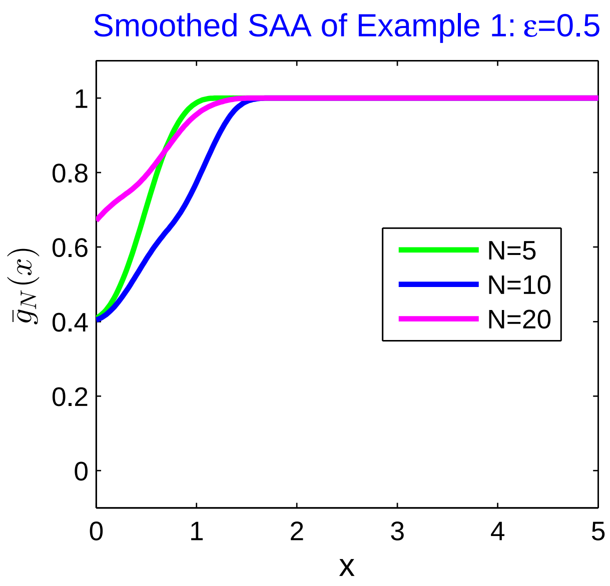


Figure 3.5: Smoothed SAA functions with different N and ε.

of the true function than function $\bar{f}_N$ does in Figure 3.1.

More detailed experiment results are presented in Section 3.4. Before moving to that, the convergence result of the smoothed SAA method are stated first. In particular, we are interested in the asymptotic performance of the solution of problem (3.7) as the sample size N goes to infinity.

3.3 Convergence of Solution Set

Let Π and Π_N^S denote the set of optimal solutions of (3.2) and (3.7) respectively. For sets $A, B \subset \mathbb{R}^d$ we denote the distance from $x \in \mathbb{R}^d$ to A by $d(x,A) := \inf_{x' \in A} \|x - x'\|$, and the deviation of the set A from the set B by $\mathbb{D}(A,B) := \sup_{x \in A} d(x,B)$. By the definition, $d(x,A) = +\infty$ if A is empty.

We are interested in the asymptotic performance of Π_N^S as the sample size N goes to infinity. We first show that under a set of conditions, $\bar{g}_k(x)$ converges to $f(x)$ uniformly in x with probability one as $N \to \infty$, and then we show that the convergence of solution set can be reached with probability one.

Assumption 2. *$\varepsilon(N) \to 0$ as $N \to \infty$, that is, the error introduced by smoothing diminishes as sample size increases.*

Assumption 3. *Π is nonempty and there exists a compact set $C \in \mathbb{R}^d$ such that Π is contained in C and $f(x)$ is continuous on C.*

Assumption 4. *With probability 1 for N large enough and ε small enough, the set Π_N^S is nonempty and $\Pi_N^S \subset C$ where C follows the conditions in Assumption 3.*

The analysis of convergence consists of two parts. We first show $\bar{g}_N(x)$ converges to $f(x)$ w.p.1 uniformly in Lemma 1. Based on that, the convergence of solutions set is developed in Theorem 1.

Lemma 1. *Suppose that Assumption 1 holds. In addition, assume that $\Psi(\cdot, \varepsilon(N))$ converges to $\mathbf{1}_{[0,\infty)(\cdot)}$ uniformly on the image set $\{c(x,\xi) : x \in \mathscr{X}, \xi \in \Xi\}$. Then for any given x, $\bar{g}_N(x)$ converges to $f(x)$ w.p.1 as N goes to ∞.*

Furthermore, assume that Assumptions 2 and 3. Then, $\bar{g}_N(x)$ converges to $f(x)$ w.p.1 uniformly on C.

Proof. Note that

$$\begin{aligned} |\bar{g}_N(x) - f(x)| &\leq \left|\bar{g}_N(x) - \bar{f}_N(x)\right| + \left|\bar{f}_N(x) - f(x)\right| \\ &\leq \frac{1}{N}\sum_{i=1}^{N}\left|\Psi\Big(c(x,\xi_i), \varepsilon(N)\Big) - \mathbf{1}_{[0,\infty)}\Big(c(x,\xi_i)\Big)\right| + \left|\bar{f}_N(x) - f(x)\right|. \end{aligned} \tag{3.8}$$

The first term in (3.8) converges to 0 w.p.1 by the uniform convergence assumption and the second term converges to 0 w.p. 1 by the LLN . In fact, the first term converges uniformly w.p.1. With additional assumption, the second term converges to zero uniformly w.p.1 by Shapiro (2003, Proposition 6).

□

Now we are showing the convergence result of optimal solution set as $N \to \infty$. The core result is based on the proposition of Shapiro (2003, Proposition 6), which is stated below without a proof.

Proposition 5 (Shapiro (2003)). *Suppose that there exists a compact set $C \subset \mathbb{R}^d$ such that: (i) the set Π of optimal solutions of the true problem is nonempty and is contained in C, (ii) the function $f(x)$ is finite valued and continuous on C, (iii) $\bar{g}_N(x)$ converges to $f(x)$ w.p.1 as $N \to \infty$, uniformly in $x \in C$, (iv) w.p.1 for N large enough the set Π_N^S is nonempty and $\Pi_N^S \subset C$. Then $\mathbb{D}(\Pi_N^S, \Pi) \to 0$ w.p.1 as $N \to \infty$.*

Theorem 1. *Let Assumption 1 to 3 hold. Then* $\mathbb{D}(\Pi_N^S, \Pi) \to 0$ *w.p.1 as* $N \to \infty$.

Proof. The conclusion is direct from Theorem 5. Conditions (i), (ii) and (iv) of Shapiro (2003, Proposition 6) hold by our Assumption 3 and 4. Condition (iii) is satisfied from our Lemma 1. Then the convergence is proved. $\square$

3.4 Numerical Experiments

3.4.1 Purpose and Test Problems

The purpose of the numerical experiments is to investigate the following two problems:

1. Does solving a smoothed SAA problem instead of a standard SAA problem really help numerical algorithms mitigate the situation of getting trapped?
2. Is there a better selection of N and $\varepsilon(N)$ we can find such that a better performance can be obtained?

The test problems are selected from the numerical examples of Kibzun and Kan (1996). The problem 'prob1' is defined as below:

$$\min_{x \in \mathbb{R}^2} f(x) := \Pr\left\{\xi(1)\left(x(1)^2 + x(2)^2\right) / \left(1 + x(1)^2 + x(2)^2\right) - \xi(2) > 0\right\},$$

where independent random variables $\xi(1)$ and $\xi(2)$ are normally distributed, which is $\xi(1) \sim \text{Norm}(1,1)$ and $\xi(2) \sim \text{Norm}(2,1)$. The problem has optimal value of 0.023 at the singular optimal point $[0,0]$.

Another test problem 'prob2' is given as

$$\text{prob2:} \quad \min_{x \in \mathbb{R}^2} := \Pr\left\{(\xi(1) + \xi(1))(x(1)^2 + x(2)^2 + 1) - 2 > 0\right\},$$

where independent random variables $\xi(1)$ and $\xi(2)$ are normally distributed, namely $\xi(1), \xi(2) \sim \text{Norm}(0,1)$. Prob2 reaches the optimal value of 0.05 at the solution point $[0,0]$.

We show the SAA shapes of test problems with different sizes of samples in Figure 3.6. It can be seen that the shape of the SAA functions is severely concealed by the discreteness. A large flat region is existing, especially when the sample size is small. Local convergent algorithms cannot work well on these functions as they would easily get trapped and announce the end of an algorithm in the flat area. A type of direct search method embedded in the pattern search toolbox of Matlab is employed trying to solve these standard SAA problems, and it turns out that the algorithm does not work until the sample size increases to 100.

On the other hand, the shape of the smoothed SAA functions $\bar{g}_N$ with different sizes of N is presented in Figure 3.7. Visually the smoothed SAA problems have better structures for optimization algorithms to work on than the standard but discontinuous SAA problems. In the next subsection, we take direct search methods as solver to illustrate how local convergent optimization algorithms perform on the smoothed SAA and standard SAA problems.

3.4.2 Implementation and Results

For simplicity, the two schemes of standard SAA methods and smoothed SAA methods are referred to as $\varepsilon = 0$ and $\varepsilon > 0$ case respectively. The solver employed in the experiment is the pattern search toolbox embedded in Matlab. For a fair comparison, the two schemes of $\varepsilon = 0$ and $\varepsilon > 0$ share all the algorithm parameters (including random stream) except for the value of ε. The procedures of the experiment are described as below. For a given test problem and a given sample size N,

1. randomly generate a starting point by using Latin hypercube sampling within a neighborhood of the optimal solution,
2. create four objective functions of the test problem with different values of ε, including $\varepsilon = 0$ and three different values of positive ε,
3. run Matlab's pattern search function with same stopping criteria and algorithm parameters on the four objective functions,
4. record the iteration history and compare the performance based on the history paths.

Note that no specific form of $\varepsilon(N)$ has been restricted by now. Driven by the convergence purpose, Assumption 2 is the only condition required. In the test, three constant values of the smoothing parameter, which are 0.5, 1, and 2, are combined with the given N trying to find out a smarter strategy to determine $\varepsilon(N)$.

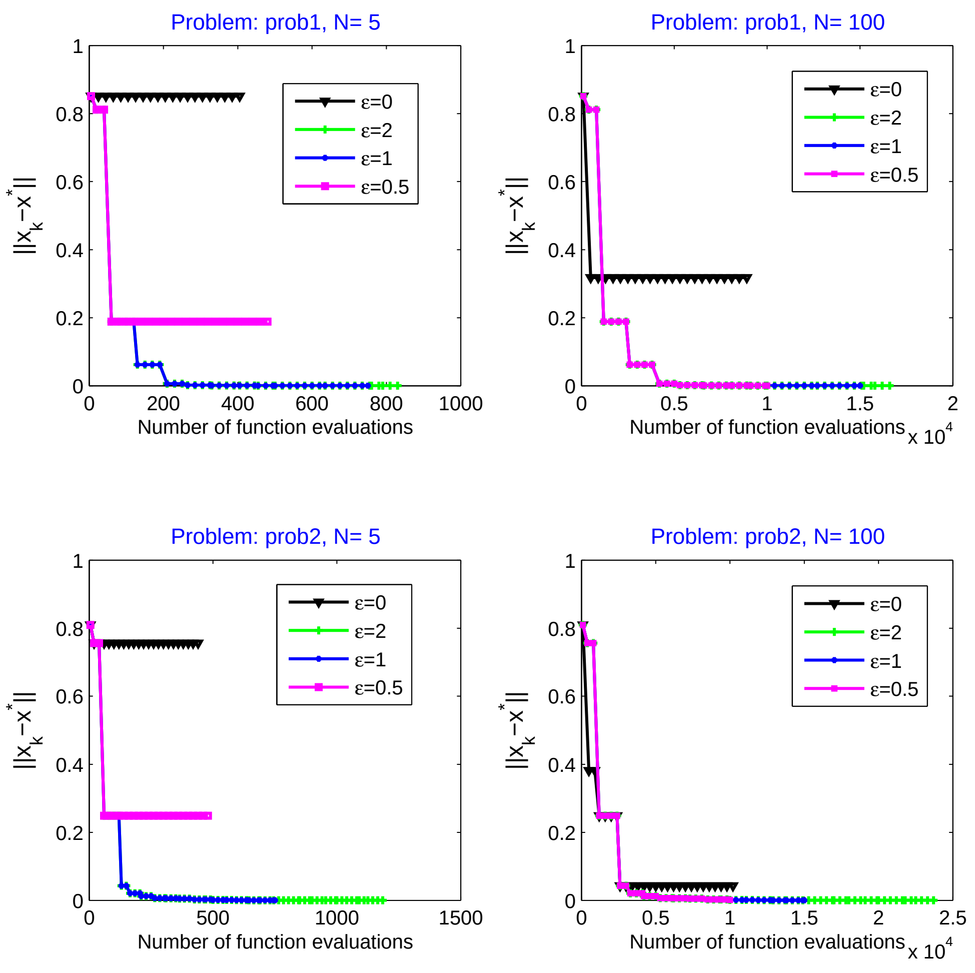


Figure 3.8: Iteration history of direct search method on true and smoothed SAA problems

The iteration paths obtained from one single run are presented in Figure 3.8. The horizontal axis denotes the accumulated function evaluations, whereas the y-axis denotes the Euclidean distance between iterate x_k to the optimal solution x^*. We choose this Euclidean distance as a measure to qualify the current solution because it can clearly indicate whether the algorithm is approaching to the correct solution or getting stuck in the current point. The stopping criteria is in terms of a limited number of function evaluations. For a clear presentation, we set different stopping criteria to the different schemes so that the curves can still be differentiated even when they overlap. When we do this, we are careful enough to make sure all the paths shown have reached the state of convergence so that no prospective information is missing.

We have chosen two sets of sample size, 5 and 100 respectively, to observe how N will affect the utility of smoothing parameter. Figure 3.8 shows that when sample size is small ($N = 5$), introducing positive smoothing parameters can significantly increase the chance of algorithm getting out trapped and moving to the correct solution. Specifically, in the $\varepsilon = 0$ case, the algorithm does not move and hence gets stuck on the both test problems from the very beginning. On the other hand, all the smoothed SAA problems successfully help the algorithm step out and move toward the correct solution. In the $N = 5$ examples, the schemes of $\varepsilon = 1$ and $\varepsilon = 2$ even obtain the optimal solutions to the both test problems.

As sample size grows to 100, the performance of the $\varepsilon = 0$ becomes better than that when $N = 5$. The reason is intuitive in that $\bar{f}_N(x)$ is approaching to the true function $f(x)$ as N grows up, implying that the SAA function may exhibit less flat area compared with smaller sample cases. Despite of the improvement of the standard SAA scheme, the three cases of $\varepsilon > 0$ still work even better. For the two test problems, when $N = 100$, all the paths of $\varepsilon > 0$ cases converge to the true solution successfully. Interestingly, the three parameters of ε exhibit amost same path so that it is difficult to tell which parameter is better.

Due to the randomness of samples, the single-run performance shown in Figure 3.8 is also random. Hence, the following numerical experiment is executed to show an average performance of the two methods based on multiple runs. The procedures are described as below:

1. randomly generate m starting points around the neighborhood of the optimal solution using Latin hypercube sampling.
2. for each starting point, run the solver multiple times with the limit evaluation budget of $\{n, 2n, 3n, \cdots, \}$, where $n = 10d$ by default. The limit of the budget sequence is tested and adjusted to make sure all paths can reach the state of convergence. Record the optimal solution solution x^*_{budg}, where the subscript implies the limited budget stopping criteria, and calculate $\|x^*_{\text{budg}} - x^*\|$ to qualify the performance of the solution.
3. average the m values of $\|x^*_{\text{budg}} - x^*\|$ and plot the average versus the budget sequence.

The average performance based on 20 runs is presented in Figure 3.9. Since same staring point is used for all the scenarios, the four curves are expected to start from the same point on the vertical axis. Two sets of N are practiced to check how the smoothed method works when the sample size is comparatively small ($N = 5$) and relatively large ($N = 100$).

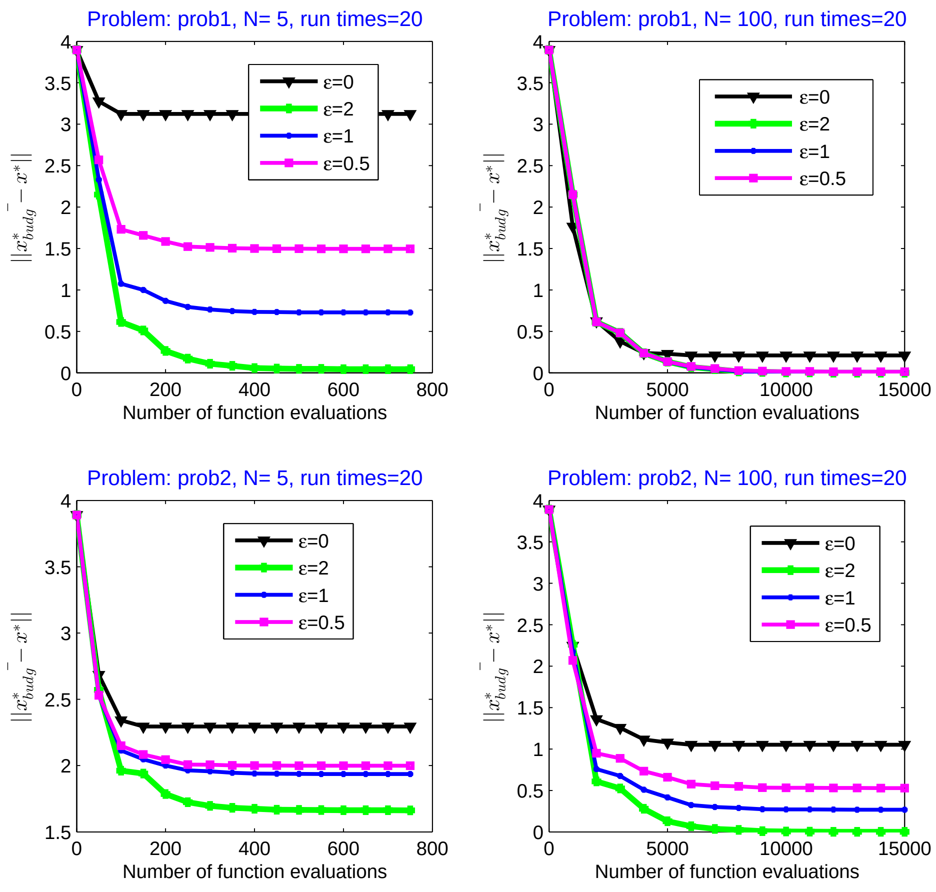


Figure 3.9: Multiple runs performance of direct search methods on standard and smoothed SAA problems

The first thing we notice in Figure 3.9 is that the results of $\varepsilon > 0$ schemes on small sample case is much better than the standard SAA method. Since the vertical axis denotes the distance between the true optimal solution and the solution obtained by the solver with limited function evaluation budget, and both test problems have only one global optimal solution and no local optimal solution, we say that the less the vertical value is, the better the algorithm performs. It can be seen that the black curve tends to stay constantly at the same level since a very early stage, indicating that the solver gets stuck in a flat region. Furthermore, it has an obvious higher value of y-axis than the other three curves. As a matter of fact, all the performance of $\varepsilon = 0$ curves confirms the issue we proposed in the beginning of this chapter, which is, local convergent algorithms cannot work well on a SAA probability problem, particularly when sample size is small.

On the other hand, it can be seen from the figure that introducing smoothing techniques gains an obvious improvement on the performance of the solver especially in the small sample case. Figure 3.9 shows that all the vertical values of colorful curves, which denote the cases of $\varepsilon > 0$, are less than the black one, where the objective problem is a true SAA discrete function. Furthermore, this dominance is consistent from the beginning to the end of the algorithm independent of the evaluation budget. Another trend can be observed is that when $N = 5$, the dominance of $\varepsilon > 0$ cases get weaken as the value of ε decreases. What surprises us most is the path of $\varepsilon = 2$ in prob1. For 20 times of simulation runs, they almost converge to the true solution every time, which is really impressive considering the sample size is only 5.

When sample size increases to 100, the performance of all schemes improves. Nevertheless, the smoothed SAA cases still perform better than the standard SAA though the dominance is not as obvious as when $N = 5$. For prob1, all paths of $\varepsilon > 0$ converge to the true optimal solution, while the path of $\varepsilon = 0$ fails to do that.

To better answer the two questions proposed in the beginning of this section, the parameters of ε and N are tuned and tested. Under different combination of ε and N, we run the algorithm 100 times using the same starting point $x_0 = [2,2]$ but different samples, and count the number that the solver gets trapped. The results are shown in Figure 3.10.

For each test problem, three sets of sample size, which are 5, 10, and 50, are tested. Under each scheme of N, the failure times of smoothed SAA and standard SAA are counted. By saying a run is failure, we mean that the algorithm fails to reach the optimal solution x^* when the function evaluation budget is unlimited. The horizontal axis denotes the smoothing parameter ε used for smoothing, which is uniformly divided with a step size of 0.02 when plotted. The vertical axis denotes the number of times that the algorithm fails to converge to the optimal solution. Three colored curves are presented to denote the three schemes of N, whereas three horizontal lines with the matching colors denote the number of failures in the standard SAA case. The legend of two plots is the same, hence we omit it in the right plot to present the curves clearly.

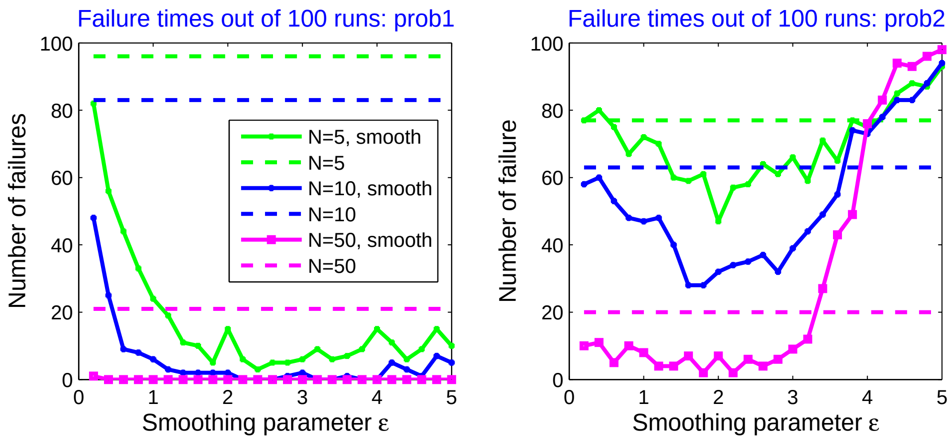


Figure 3.10: Failure times out of 100 runs

In the example of prob1, smoothing techniques improve the performance impressively compared with the standard SAA case. This dominance is especially significant when sample size is small. It can be seen that more than 80% of times standard SAA method fails to work when $N = 10$, and almost 100% fail when $N = 5$. On the other hand, smoothed SAA insists with a strong performance when $\varepsilon > 1$, which tends to suggest that the smoothing parameter should not be too small when sample size is small.

The result of prob2 on the right gives us more inspirations. For all the three values of N, the failure times of smoothed SAA are more than that of the corresponding standard SAA when ε is larger than a certain level, implying that a too big value of ε can make the problem out of shape. For different sample size, this level of breaking point also differs. In the plot, we can check this level by observing the intersection of solid colorful curve and the corresponding dash line. For $N = 5$, the intersection locates around the value of $\varepsilon = 4$; for $N = 10$, this value decrease a little to $\varepsilon = 3.7$; and for $N = 50$, the intersection locates around $\varepsilon = 3.5$. This trend of phenomenon is reasonable in that positive smoothing parameters introduce an error to the function. Though a small sample sized SAA problem might not be accurate enough either, a too large smoothing parameter can still deteriorate the shape of the function.

In addition, when sample size is small like $N = 5$ and $N = 10$, a small value of ε does not help a lot. Notice that when $N = 5$, the smoothed SAA performs even worse than the corresponding standard SAA when $\varepsilon < 0.5$. Similarly, when $N = 10$, the smoothed case does not show obvious dominance until $\varepsilon > 1$.

Going back to the two questions proposed in the very beginning of this section, we give the following answers. Solving a smoothed SAA problem instead of a standard SAA problem can help numerical algorithms get out of trapped with a higher chance especially when the sample size is small. However, this help is not guaranteed for every single run due to the randomness of the samples. The selection of smoothing parameter would also affect the performance of smoothed SAA methods. For each sample size, there should exist a range of smoothing parameters such that smoothing techniques work better with; however, an optimal value of the smoothing parameter is difficult to decide in advance. This value should be closely related to the structure of the problem and also depend on the samples especially when sample size is small.

3.4.3 Comparison with Other Algorithms

There are not many stochastic algorithms designed specially for probability simulation optimization problems. To the best of our knowledge, we compare the results of our smoothed SAA methods with the results shown in Kibzun and Kan (1996). Kibzun and Kan (1996) presented the results of the same test problems obtained from four algorithms. The four algorithms, named Uryas'ev algorithm (Uryas' ev, 1989), Lepp algorithm(Lepp, 1983), Raik algorithm(Raik, 1975), and Ubi algorithm(Ubi, 1977), are four gradient-based stochastic algorithms using different gradient approximation techniques.

The tuned parameters and computation results are presented in Table 3.1.

For all the algorithms and two test problems prob1 and prob2, the initial point x_0 is set to be $[2,2]$. The corresponding probability value is equal to $f(x_0) \approx 0.74$ for prob1 and $f(x_0) \approx 0.44$ for prob2. The algorithm is called stop when the the solution accuracy with respect to x reaches 0.01, which means the algorithm terminates when $\|x_k - x^*\| < 0.01$. The three gradient-based algorithms increase sample size iteratively depending on the iteration number k, while the smoothed SAA uses a fixed sample size of five.

We are interested in the total function evaluations consumed by the algorithm when it reaches the solution accuracy standard. For the smoothed SAA case, this number is recorded and directly output by Matlab's pattern search toolbox. For the other three counterparts, this number could be obtained by summing up all the iteration sample size from the total number of iterations.

prob1				
Algorithm of:	Uryas'ev	Lepp	Ubi	$\varepsilon = 3$
Sample size at iteration k: N_k	$5k^{1/3}+1$	$5k^{1/3}+1$	$k^{8/5}+10$	5
Total number k of iterations	70	30	25	3
Total function evaluations	1196	401	2008	60
Solution accuracy w.r.t. x	0.01	0.01	0.01	0.01
prob2				
Algorithm of:	Uryas'ev	Lepp	Ubi	$\varepsilon = 3$
Sample size at iteration k:N_k	$5k^{1/3}+1$	$5k^{1/3}+1$	$k^{8/5}+10$	5
Total number k of iteration	12	10	25	3
Total function evaluations	125	99	731	60
Solution accuracy w.r.t. x	0.01	0.01	0.01	0.01

Table 3.1: Comparison with other algorithms

From Table 3.1 we see that $\varepsilon = 3$ has the smallest function evaluations of all with obvious dominance. This is not only due the small sample size ($N_k \equiv 5$), but also owing to the fact that direct search methods quickly approach to the true solution with the help of smoothing. As a matter of fact, though the stopping accuracy is set as 0.01, direct search methods return to the optimal solution when the algorithm terminates.

Since the samples generated are random, the performance of the algorithm is affected by the samples as well especially when the sample size is small. We need to point out that for prob2, when $N = 5$ and $\varepsilon = 3$, our method is not guaranteed to reach the solution accuracy every time. Once the smoothed SAA works, the direct search solver can find the optimal solution very quickly, consuming only 60 function evaluations; however, sometimes the smoothing fails to help the algorithm get out of trapped, in which cases the algorithms stays constantly at the initial point x_0. The frequency of this failure can be roughly referred to Figure 3.10.

3.5 Conclusions

We propose a smoothed SAA method to deal with simulation-based probability optimization problems so that a wide range of (nonlinear) optimization algorithms can be successfully applied to solve the problems. It is also stated under mild conditions that the solution set of the smoothed SAA problems converges to the counterpart of the true problem as sample size goes to infinity.

In the numerical experiments, we employ a type of direct search methods as solver to illustrate that the performance of the smoothed SAA methods is better than the standard SAA methods. Depending on the problem, the dominance of our method can be significant especially when the sample size is small, meaning that a significant amount of simulation cost can be saved compared with standard SAA methods.

The relationship between the sample size and the smoothing parameter is also investigated. It is implied by the numerical results that when sample size is small, a comparatively big smoothing parameter should be used, but the issue is that selecting a proper value of either sample sizes or smoothing parameters in advance without knowing any information about the problems seems intractable. In the next chapter, we enhance the smoothed SAA methods to tackle this issue by borrowing the idea behind retrospective-approximation methods.

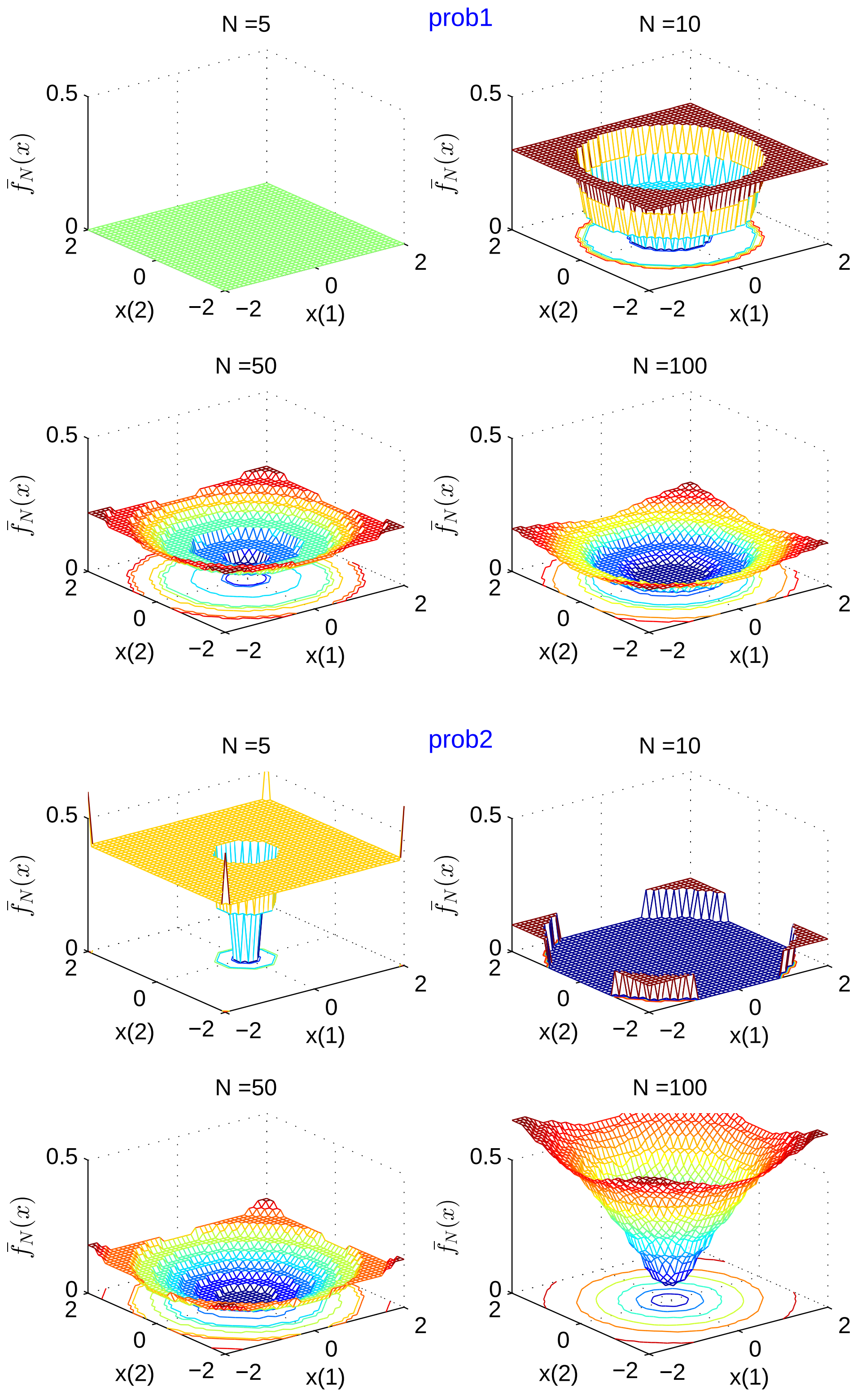


Figure 3.6: SAA functions with different N

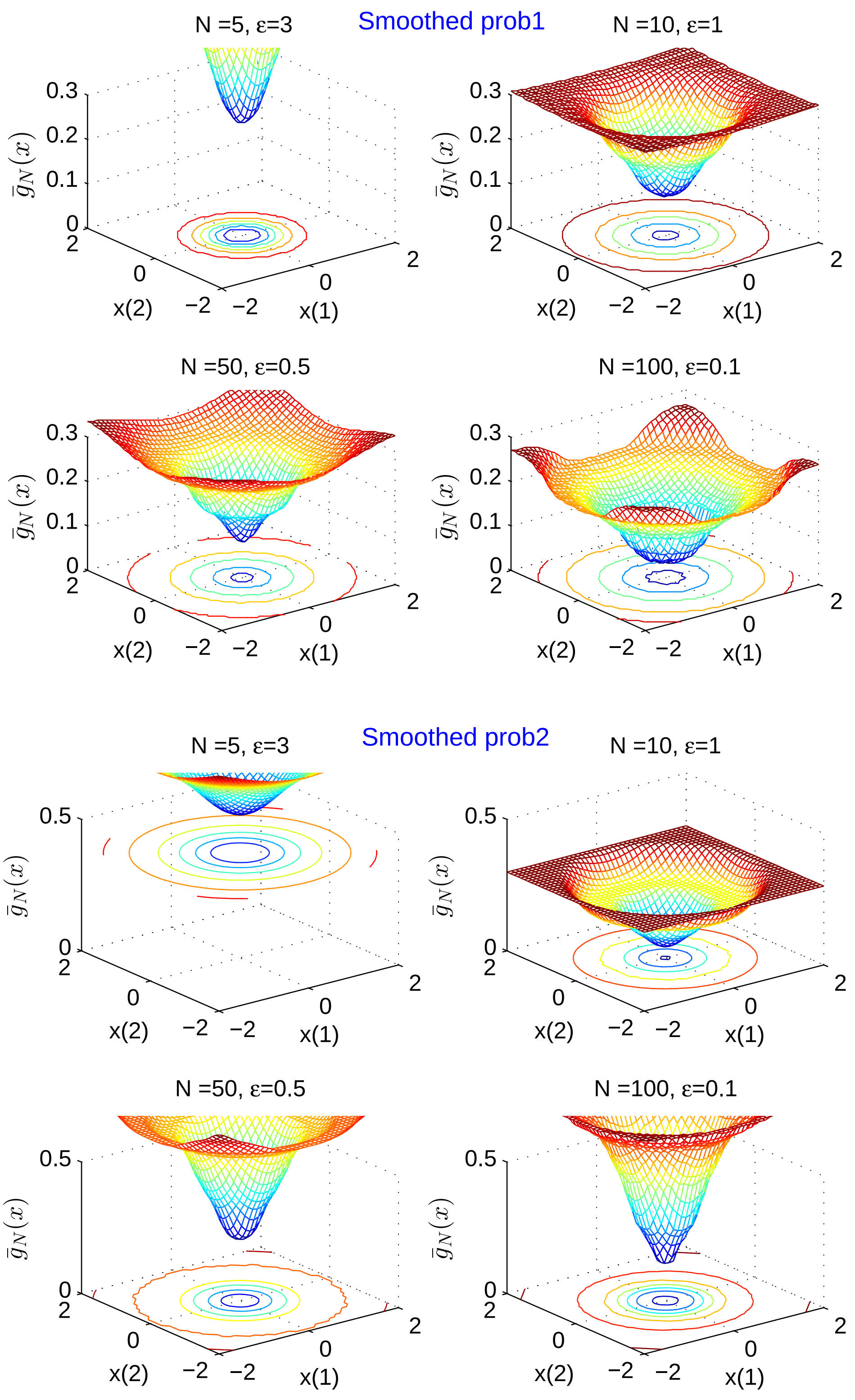


Figure 3.7: Smoothed SAA functions with different *N* and ε

Chapter 4

Retrospective-Approximation Algorithms Using Direct Search Methods

4.1 Introduction

It is presented in the previous chapter that local convergent numerical algorithms could work well on the probability optimization problems by combining the idea behind the SAA methods with smoothing techniques. However, determining an appropriate number of sample size in advance without knowing any information about the problems is difficult. To deal with that, the principle of a refined family of SAA methods named retrospective approximation is introduced to incorporate into a type of direct search methods to enhance the smoothed SAA methods proposed before.

Retrospective approximation (RA) algorithms, described as a sequential-sampling approach by Homem-de Mello and Bayraksan (2013), are originally proposed by Chen and Schmeiser (2001) and later extended by Pasupathy and Schmeiser (2009) for stochastic root finding problems. The desire of RA algorithms is due to a computational strategy. Unlike classic SAA methods that use a fixed number of sample size to generate a sample-path approximation problem, RA algorithms generate a sequence of SAA problems with increasing sample sizes. By employing an appropriate deterministic algorithm, these sequential SAA problems are solved with gradually decreasing error tolerances. The rationale behind RA algorithms is the belief that computation cost could be saved if small sample sizes are used in the early stage when the current solution is far from the optimal one, and more sampling effort is put in as the solution is approaching to the true solution in the later stage.

A similar idea is studied in the context of stochastic programming by Shapiro and Homem-de Mello (1998), who incorporated sampling into a two-stage stochastic programming problem with recourse. Homem-De-Mello (2003) studied variable-sample techniques backed up by the same principle as RA algorithms under the scheme of discrete stochastic optimization. Instead of using a pre-determined sequence of sample sizes, Deng and Ferris (2009) proposed a new variant of variable-sample method using trust region algorithms. The proposed method integrates Bayesian analysis techniques to determine an appropriate number of samples based on an instant inspection of the current step.

In this work we describe a type of RA algorithms using direct search methods. We discuss and present the numerical results of the algorithm for probability optimization problems. The algorithm is stated in two-fold. We first assume a general stochastic objective function such that direct search methods can be directly applied to its corresponding SAA problems. The convergence properties of the direct search methods are studied and it is presented that the combination of direct search methods and RA algorithms is reasonable and applicable. It is also discussed that the parameters of the proposed RA algorithms using direct search methods, named RA-DS method, can be determined under certain conditions by following the technical guidelines and specific recommendations proposed by Pasupathy (2010) for standard RA algorithms. In the second part, the RA-DS method are adjusted by introducing a new sequence of smoothing parameters to specifically deal with probability optimization problems. Due to the smoothing techniques they use, the modified methods are referred to as retrospective smoothing direct search (RS-DS) method. The numerical experiments are implemented to investigate an appropriate choice of the new parameter. In addition, the numerical results of the RS-DS method are compared with the smoothed SAA methods proposed in Chapter 3.

One of the recent works that closely relates to this study is by Pasupathy (2010), who presented a comprehensive study of convergence rate of the RA algorithms and provided a guideline for choosing the sequence of sample sizes and error tolerances in the methods. The conditions that Pasupathy (2010) proposed on the two sequences are discussed under the scheme of using direct search methods and the specific choices he recommended for the sequences will be applied to the RA-DS method and RS-DS method.

The remainder of this chapter is organized as follows. Section 4.2 proposes the outline of the RA-DS method and investigates appropriate forms of the parameters by discussing the convergence properties of the direct search methods. The RA-DS method are developed in Section 4.3 to especially deal with probability objectives. The choice of the smoothing parameter that is newly introduced and the performance of the developed methods are studied. Section 4.4 concludes the work and gives some aspects of the future work.

4.2 Retrospective-Approximation Algorithms Using Direct Search Methods

Consider the following stochastic optimization problem:

$$\min_{x\in\mathscr{X}} f(x) = E\left[\Phi(x,\boldsymbol{\xi})\right], \tag{4.1}$$

where $x \in \mathbb{R}^d$ is a decision vector, $\mathscr{X}$ is a subset of $\mathbb{R}^d$. Unless stated otherwise we assume that the feasible set $\mathscr{X} = \mathbb{R}^d$ or it is boxed constrained as $\mathscr{X} = [LB, UB]$, where $LB, UB \in \mathbb{R}^d$. $\xi \in \mathbb{R}^d$ is a vector of uncertain parameter, whose support is denoted as Ξ. We also assume that the expected function $f(x)$ is well defined for $x \in \mathscr{X}$, which means that $\Phi(x,\cdot)$ is measurable, with respect to the Borel sigma algebra of $\mathbb{R}^d$, and either $E[\Phi(x,\boldsymbol{\xi})_+]$ or $E[-\Phi(x,\boldsymbol{\xi})_+]$ is finite (Shapiro, 2003). In order to distinguish between random data and their numerical values, we would use bold script $\boldsymbol{\xi}$ for the random vector, and ξ for its particular realization afterwards.

Assumption 5. *The true objective function $f(x)$ is continuously differentiable on $x \in \mathscr{X}$ and ∇f is Lipschitz continuous with constant κ^f.*

Assumption 5 is used to clearly define the original objective function. It is a very standard assumption for direct search methods to guarantee a first-order convergence.

Assumption 6. $\Phi(x,\xi)$ *is continuously differentiable on* $x \in \mathscr{X}$ *for almost every* $\xi \in \Xi$.

Assumption 6 is to assure that a sample-path problem of (4.1) can be solved by direct search methods likewise.

4.2.1 Outlines of the RA-DS Method

In the context of simulation optimization, given a realization of ξ, $\Phi(x,\xi)$ can be evaluated via a single simulation run. Applying sample average approximation (SAA) methods, $f(x)$ is approximated by taking an average over the sample responses:

$$\min_{x\in\mathscr{X}} \bar{f}^N(x) := \frac{1}{N}\sum_{i=1}^{N}\Phi(x,\xi_i). \tag{4.2}$$

Instead of solving a single SAA problem, RA algorithms work sequentially by generating a sequence of (4.2) with increasing sample sizes and solving them by employing a numerical method with decreasing error-tolerance.

Let index variables $j = 1,2,\cdots$ denote the stage that the RA algorithms are at, and N_j denote the sample size used in the jth stage. Let $\underline{\xi}_j = \{\xi_{j,1},\cdots,\xi_{j,N_j}\}$ denote the samples generated at jth stage, where $\xi_{j,i}$, $i = 1,\cdots,N_j$ mean the ith realization of randomness $\boldsymbol{\xi}$ at jth stage. Then RA algorithms generate and solve a SAA problem at each stage of the following form:

$$\min_{x\in\mathscr{X}} \bar{f}_j(x) = \frac{1}{N_j}\sum_{i=1}^{N_j}\Phi(x,\xi_{j,i}), \quad j = 1,2,\cdots. \tag{4.3}$$

For any given stage j, the samples within the stage $\xi_{j,i}$s are generated independently by Monte Carlo sampling. However, the samples between stages can be generated either independently or dependently. In the independent sampling scheme, $\underline{\xi}_j$ contains N_j new observations independent from all previous samplings; whereas in the later case, there is $\underline{\xi}_j = \underline{\xi}_{j-1} \cup \{\xi_{j,N_{j-1}+1}, \xi_{j,N_{j-1}+2}, \cdots, \xi_{j,N_j}\}$, meaning that all the observations in $\underline{\xi}_{j-1}$ are reused and combined with $(N_j - N_{j-1})$ new observations to consistent of $\underline{\xi}_j$.

Direct search methods are applied to solve the stage problem (4.3). For a given stage j, the algorithms terminate when the mesh size shrinks to or smaller than Δ_j^{tol}. Each stage returns a solution of x_j, which is the best solution have found when the stage terminates. The stage solution x_j is set as the initial guess (starting point) for the $(j+1)$ stage. The procedures of the RA-DS method are described in Figure 4.1.

```
**Initialization:**
    Set $j = 1$, $k = 1$, and set:
        • an initial sample size $N_1$ and a rule for increasing $N_j$ for $j \geq 2$;
        • an stopping meshsize $\Delta_j^{\text{tol}}$ and a rule for decreasing $\Delta_j^{\text{tol}}$ for $j \geq 2$;
        • a starting point $x_j^{(k)}$ and other parameters required for direct search
          methods (which can be referred to Figure 2.1).
**Ensure:**
for $j = 1, 2, \cdots$ or while there is simulation budget, do
    for $k = 1, 2, \cdots$ do
        generate a direction set $D_j^{(k)}$ under some technical conditions;
        if $\exists d_j^{(k)} \in D_j^{(k)}$ such that $\bar{f}_j\big(x_+^{(k)}\Delta_j^{(k)} d_j^{(k)}\big) < \bar{f}_j(x_)^{(k)} - \rho(\Delta_j^{(k)})$ then
            set $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)} d_j^{(k)}$;
            set $\Delta_j^{(k+1)} = \gamma_j^{(k)} \Delta_j^{(k)}$;
        else
            set $x_j^{(k+1)} = x_j^{(k)}$;
            set $\Delta_j^{(k+1)} = \theta_j^{(k)} \Delta_j^{(k)}$, where $\theta_j^{(k)} \geq \theta_{\min}$ denotes a contraction factor;
            if $\Delta_j^{(k+1)} < \Delta_j^{\text{tol}}$ then
                set $x_j = x_j^{(k+1)}$;
                set $x_{j+1}^{(1)} = x_j$;
                set $\Delta_{j+1}^{(1)} = \Delta_j^{(k)}$;
                return $x_j$;
                break;
            end if
        end if
    end for
end for
```

Figure 4.1: RA-DS method

4.2.2 Determination of N_j and Δ_j^{tol}

For the RA algorithms, Chen and Schmeiser (2001) suggested that a family of $\{N_j\}$ with the form of $N_j = c_1 N_{j-1}$ where the constant $c_1 \geq 1$ is reasonable for their stochastic root finding problems. Correspondingly, they suggested the form $e_j = c_1^{-1/2} e_{j-1}$ for their decreasing error-tolerance sequence of $\{e_j\}$. The sequence $\{e_j\}$ in the RA is a measure of error-tolerance with respect to the distance of the current solution to the optimal solution, while in our case, $\{\Delta_j^{\text{tol}}\}$ plays a similar role of error-tolerance but with respect to the mesh size.

Pasupathy (2010) provided technical conditions on choosing the sequences of $\{N_j\}$ and $\{e_j\}$ in RA algorithms so that a certain rate of convergence of RA's iterates is guaranteed. A specific recommendation for the sample size and the error-tolerance sequences proposed by Pasupathy (2010) to reach the convergence rate is $N_j = c_N N_{j-1}$ and $e_j = c_e / \sqrt{N_j}$, where c_N is some constant greater than one and c_e is some constant greater than zero. There are two critical components required to reach the recommended form. The first is that the numerical algorithm used to solve sample-path problems exhibits linear or polynomial convergence. The second is that the stage solution x_j satisfies $\|x_j - x_j^*\| \leq e_j$ w.p.1, where x_j^* is the unique sample-path solution to the jth stage SAA problem.

To obtain appropriate forms of the N_j and Δ_j^{tol} in the RA-DS method, we first state the result that under certain conditions $\|x_j - x_j^*\| \leq c\Delta_j^{\text{tol}}$ holds. Based on this result, it can be justified that the specific recommendations of Pasupathy (2010) could be borrowed to the RA-DS method as the parameters of Δ_j^{tol} play a similar role as the error-tolerance e_j in the retrospective-approximation algorithm.

We impose the following assumption on the response function $\Phi(x,\xi)$ and sample-path functions $\bar{f}_j(x)$.

Assumption 7. *Suppose that*

1. $\Phi(x,\xi)$ *is twice continuously differentiable on* x *w.p.1;*
2. $\nabla^2 \bar{f}_j(x_j^*)$ *is positive definite w.p.1 where* x_j^* *is a local minimizer of* $\bar{f}_j$ *for* $j = 1, 2, \cdots$*;*
3. *the set* $L(x_0) := \{x | \bar{f}_j(x) \leq \bar{f}_j(x_0)\}$ *is compact for any given* $x_0 \in \mathscr{X}$*.*

The first condition in Assumption 7 is to assure that for any j, the sample-path problem $\bar{f}_j(x)$ is twice continuously differentiable with respect to x w.p.1.

We also place the following conditions on the direct search methods used in the RA-DS methods.

Assumption 8. *Suppose that $\rho(\cdot) \equiv 0$ and for $j = 1, 2, \cdots$, there exists a K such that for all $k \geq K$, $\gamma_j^{(k)} \equiv 1$.*

Proposition 6. *Let Assumptions 7 and 8 hold. Then for $j = 1, 2, \cdots$, if $x_j^{(1)}$ is sufficiently close to x_j^* and $\Delta_j^{(1)}$ is sufficiently small, then we have that*

$$\|x_j - x_j^*\| \leq c_j \Delta_j^{tol}$$

w.p.1 for some constant c_j independent of k.

Proof. The proof is trivial based on the local convergence results by Dolan et al. (2003, Theorem 4.4). For any stage j, under the conditions of Assumption 7 and Assumption 8, according to Dolan et al. (2003, Theorem 4.4), for all unsuccessful iterations k, there is

$$\|x_j^{(k)} - x_j^*\| \leq c_j^1 \Delta_j^{(k)}, \tag{4.4}$$

for some constant c_j^1 independent of k. Let superscript '(end)' denote the last iteration of a stage. Due to the procedures of the RA-DS methods, it can be claimed that the '(end)' iteration must be unsuccessful. Furthermore, we have

$$\theta_j^{\text{end}} \Delta_j^{\text{end}} < \Delta_j^{\text{tol}} < \Delta_j^{\text{end}}.$$

Embedding the iteration of '(end)' in the equation (4.4), we have

$$\|x_j^{(\text{end})} - x_j^*\| \leq c_j^1 \Delta_j^{(\text{end})} \leq \frac{c_j^1}{\theta_j^{(\text{end})}} \Delta_j^{\text{tol}} \leq \frac{c_j^1}{\theta_{\min}} \Delta_j^{\text{tol}} := c_j \Delta_j^{\text{tol}}.$$

□

A similar result can also be obtained for the case of $\rho(\cdot) \neq 0$. Specific conditions need to be imposed on the $\rho(\cdot)$ to obtain the result. The conditions are clearly stated in Assumption 9.

Assumption 9. *Suppose that*

1. *for $j = 1, 2, \cdots$, $\gamma_j^{(k)} = 1$ for all unsuccessful iterations k;*
2. *$\rho(\Delta) = \alpha \Delta^p$ for some fixed $\alpha > 0$ and fixed $p \geq 2$;*

3. *there exists some $\beta_{\min}$ and $\beta_{\max}$ such that $\beta_{\min} \leq \|d_j^{(k)}\| \leq \beta_{\max}$ for all $d_j^{(k)} \in D_j^{(k)}$.*

The similar results in the case of $\rho(\cdot) \neq 0$ is stated in Proposition 7 without a proof.

Proposition 7. *Suppose Assumptions 7 and 9 hold. Then for $j = 1, 2, \cdots$, if $x_j^{(1)}$ is sufficiently close to x_j^* and $\Delta_j^{(1)}$ is sufficiently small, then there is*

$$\|x_j - x_j^*\| \leq c_j \Delta_j^{tol}$$

w.p.1 for some constant c_j independent of k and j.

For a proof of the inequality $\|x^{(k)} - x^*\| \leq c^1 \Delta^{(k)}$ for any unsuccessful iteration k, see Theorem 3.15 in Kolda et al. (2003).

Propositions 6 and 7 reveal two properties of direct search methods which make the choice of the parameters in RA-DS method and RA algorithms connected.

1. Suppose the constants c_j for $j = 1, 2, \cdots$ are upper bounded by some constant $c_{\max}$. Then according to Propositions 6 and 7, $\|x_j - x_j^*\| \leq c_{\max} \Delta_j^{\text{tol}}$ holds w.p.1 for $j = 1, 2, \cdots$, indicating that Δ_j^{tol} in the RA-DS methods plays a similar role as the error-tolerance e_j in the RA methods.
2. Direct search methods exhibit r-linearly convergence on the unsuccessful iterates, meaning that $\|x^{(k)} - x^*\| \leq \alpha_k$ for all $k \in \mathscr{U}$ sufficiently large, where $\{\alpha_k\}$ is a sequence of scalars that is q-linearly convergent to zero, and $\mathscr{U}$ denotes the subsequence of unsuccessful iterations. Hence it can be approximately supposed that direct search methods exhibit linear convergence.

Based on the above two components, according to Pasupathy (2010), a natural choice for the sequences of N_j and Δ_j^{tol} is

$$N_j = c_N N_{j-1} \quad \text{for } j > 2, \tag{4.5}$$

and

$$\Delta_j^{\text{tol}} = \frac{c_\Delta}{\sqrt{N_j}} \quad \text{for } j = 1, 2, \cdots, \tag{4.6}$$

where c_N is some constant greater than 1 and c_Δ is some constant greater than zero.

4.2.3 Numerical Experiments

We apply the RA-DS method to an adapted Rosenbrock function. The deterministic Rosenbrock function is a well-known test problem introduced by Rosenbrock (1960), and it has an single optimal solution inside a long, narrow, parabolic shaped flat valley. A two-dimensional Rosenbrock function is of the form

$$\underline{f}(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2.$$

To make the problem stochastic, x_1 is multiplied by a random variable $\xi \sim \mathcal{N}(1, 0.1^2)$, and hence the noisy Rosenbrock function is given as

$$\Phi(x,\xi) = 100\big(x_2 - (\xi x_1)^2\big)^2 + (\xi x_1 - 1)^2, \tag{4.7}$$

and the objective function is give as

$$f(x) = E[\Phi(x,\xi)]. \tag{4.8}$$

The optimal solution of (4.8) is $x^* = (0.4162, 0.1750)$, and the optimal value is $f(x^*) = 0.4616$.

The RA-DS methods with the parameters in forms of (4.5) and (4.6) are compared with SAA methods using direct search methods. Specifically, we let $N_1 = 5$ and $N_j = 2N_{j-1}$ for $j \geq 2$ and $\Delta_j^{\text{tol}} = 0.01/\sqrt{N_j}$ for $j = 1, 2, \cdots$. For the SAA methods, we test three cases of $N = 10$, $N = 20$ and $N = 50$.

The initial guess $x^{(0)}$ is randomly generated by a Latin hypercube sampling within the range of $B(x^*, 1)$, and 100 runs are implemented with a limit number of function evaluations. The quality of the solution is quantified by its distance to the optimal solution, mathematically denote as $\|x^*_{\text{budgt}} - x^*\|$. The average performance of the quality measure $\|x^*_{\text{budgt}} - x^*\|$ is given in Table 4.1.

It can be seen that in this example, when budget is small, RA-DS method exhibit a better performance than the standard SAA methods in terms of $\mu\left(\|x^*_{\text{budgt}} - x^*\|\right)$ value. The reason behind this is that the RA-DS method start with a small sample size. When the budget grows, the dominance of RA-DS method becomes obvious because as the employed sample size increases, the error of the sample-path function decreases and hence the solution estimator converges to the true solution. On the other hand, as the budget grows, the quality of the so-

		Fixed-N SAA		
Budget	RA-DS	$N = 10$	$N = 20$	$N = 50$
1e+3	0.5863	0.6886	0.7287	0.7202
1e+4	0.3342	0.5893	0.6416	0.7160

Table 4.1: $\mu\left(\|x^*_{\text{budgt}} - x^*\|\right)$ based on 100 runs, where x^*_{budgt} is the solution obtained for a given limited budget and a random initial guess

lution obtained by the SAA methods is capped because the solution converges to the SAA problem's optimal solution instead of the true solution. Since determining an appropriate sample size in advance for SAA methods is generally not easy, RA algorithms tend to be a better choice due to their gradually-increase-samples strategy when the simulation budget is large.

4.3 Retrospective-Smoothing Algorithms Using Direct Search Methods for Probability Optimization Problems

In this section we propose a variant of the RA-DS method specifically for probability simulation optimization problems. The new algorithm is referred to as the retrospective-smoothing direct search (RS-DS) method, in that it combines the smoothing techniques presented in Chapter 3 and the framework of the retrospective-smoothing algorithms discussed in the previous section.

4.3.1 Outlines of the RS-DS Methods

Considered a probability optimization program represented as follows:

$$\begin{aligned} \min_{x \in \mathscr{X}} f(x) &:= \Pr\left\{c(x,\boldsymbol{\xi}) \geq 0\right\} \\ &:= E\left[\Phi(x,\boldsymbol{\xi})\right] = E\left[\mathbf{1}_{[0,\infty)} c(x,\boldsymbol{\xi})\right], \end{aligned} \tag{4.9}$$

where all the notations denote the same as in (4.1).

We propose to optimize a sequence of smoothed SAA problems with certain stopping tolerance at successive stage $j = 1, 2, \cdots$. To accomplish that, three sequences are involved. Let $\{N_j\}$ denotes the sequence of increasing sample sizes of each stage, $\{\varepsilon_j\}$ denotes the sequence of decreasing smoothing parameters, and $\{\Delta_j^{\text{tol}}\}$ denotes a mesh-tolerance sequence that determines when to terminate current stage. We assume $\{N_j\}$ goes infinity w.p.1, and $\{\varepsilon_j\}$ and $\{\Delta_j^{\text{tol}}\}$ goes to zero w.p.1.

At stage j, once $\underline{\xi}_j$ is realized, the RS-DS methods optimize the smoothed SAA problem as follows:

$$\begin{aligned}\min_{x\in\mathscr{X}} \bar{g}_j(x) \quad &:= \quad \frac{1}{N_j}\sum_{i=1}^{N_j}\hat{\Phi}\big(x,\xi_{j,i},\varepsilon_j\big) \\ &:= \quad \frac{1}{N_j}\sum_{i=1}^{N_j}\Psi\big(c(x,\xi_{j,i}),\varepsilon_j\big). \end{aligned} \tag{4.10}$$

Suppose the smoothed functions $\hat{\Phi}$ and Ψ satisfy Assumption 1 as well.

The procedures of the RS-DS method are summarized in Figure 4.2.

4.3.2 Choice of Parameters and Numerical Experiments

Choice of ε_j

Introducing the new parameter $\{\varepsilon_j\}$ to the retrospective-approximation algorithm causes an error of function approximation, but the error is vanished asymptotically in that $\varepsilon_j \to 0$ as $j \to \infty$. In this section, we exploit an appropriate form of the parameters ε_j by conducting numerical experiments under different schemes of $\{\varepsilon_j\}$. The test problems employed in this part are the same as Chapter 3, and the specific forms of the three sequences used in the experiments are stated in Table 4.2.

$j = 1,2,\cdots$	**Scheme 1.1**	**Scheme 1.2**	**Scheme 2.1**	**Scheme 2.2**
N_j	$2^{j-1}\times 5$			
Δ_j^{tol}	$4^{-0.5(j-1)}\times 5$			
ε_j	$\frac{c_\varepsilon}{\sqrt{N_j}}$		$\frac{c_\varepsilon}{N_j}$	
c_ε	4	5	8	10

Table 4.2: Parameters settings for different schemes of ε_j

Initialization:

Set $j = 1$, $k = 1$, and set:

- an initial sample size N_1 and a rule for increasing N_j for $j \geq 2$;
- an initial smoothing parameter ε_1 and a rule for decreasing ε_j for $j \geq 2$;
- an stopping meshsize Δ_j^{tol} and a rule for decreasing Δ_j^{tol} for $j \geq 2$;
- a starting point $x_j^{(k)}$ and other parameters required for direct search methods (which can be referred to Figure 2.1).

Ensure:

```
 1: for $j = 1, 2, \cdots$ or while there is simulation budget, do
 2:   for $k = 1, 2, \cdots$ do
 3:     generate a direction set $D_j^{(k)}$ under some technical conditions;
 4:     if $\exists d_j^{(k)} \in D_j^{(k)}$ such that $\bar{g}_j\big(x_+^{(k)}\Delta_j^{(k)} d_j^{(k)}\big) < \bar{g}_j(x_)^{(k)} - \rho(\Delta_j^{(k)})$ then
 5:       set $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)} d_j^{(k)}$;
 6:       set $\Delta_j^{(k+1)} = \gamma_j^{(k)} \Delta_j^{(k)}$, where $\gamma_j^{(k)} \geq 1$ denotes an expansion factor;
 7:     else
 8:       set $x_j^{(k+1)} = x_j^{(k)}$;
 9:       set $\Delta_j^{(k+1)} = \theta_j^{(k)} \Delta_j^{(k)}$, where $\theta_j^{(k)} \geq \theta_{\min}$ denotes a contraction factor;
10:       if $\Delta_j^{(k+1)} < \Delta_j^{\text{tol}}$ then
11:         set $x_j = x_j^{(k+1)}$;
12:         set $x_{j+1}^{(1)} = x_j$;
13:         set $\Delta_{j+1}^{(1)} = \Delta_j^{(k)}$;
14:         return $x_j$;
15:         break;
16:       end if
17:     end if
18:   end for
19: end for
```

Figure 4.2: The RS-DS method

Schemes 1.1 and 1.2 take the specific form recommended by Pasupathy (2010) for the error-tolerance sequences in the retrospective-approximation algorithms. Under these schemes, the smoothing parameters $\{\varepsilon_j\}$ decrease at the same rate as Δ_j^{tol}, and further we have $\varepsilon_j^2 = O\big(\frac{1}{N_j}\big)$. On the other hand, Schemes 2.1 and 2.2 let ε_j go to zero faster than the first two schemes by setting

$\varepsilon_j^2 = o(\frac{1}{N_j})$. The row of c_ε is determined as in the table to make sure the ε_1 values under the four schemes are all approximately around 2.

The procedures of comparing the RS-DS method under different schemes of $\{\varepsilon_j\}$ are as follows:

1. generate 100 initial guesses within $B(0,5)$ using Latin Hypercube sampling;
2. for each initial guess, employ the RS-DS method with the parameters in Table 4.2 to get a solution $\underline{x}^*$ such that $\|x^* - \underline{x}^*\| < e$, where multiple values of the error-tolerance e are set and tested; and $\underline{\boldsymbol{\xi}}_j$ is independent of $\underline{\boldsymbol{\xi}}_{j-1}$ in the implementation;
3. record the accumulated function evaluations consumed by the four schemes and compare the average evaluations of 100 runs.

The results of testing the RS-DS method on problem prob2 is presented in Table 4.3.

error-tolerance e	**Scheme 1.1**	**Scheme 1.2**	**Scheme 2.1**	**Scheme 2.2**
10^{-1}	1540	1192	3769	1810
10^{-2}	2853	2522	20189	15670
10^{-3}	6058	5011	34282	29606
10^{-4}	10177	8718	78064	63547

Table 4.3: Average evaluations consumed by the RS-DS method under different schemes of $\{\varepsilon_j\}$ and different values of error-tolerance

Based on the results for this example, the evaluations consumed by Schemes 2.1 and 2.2 grow much faster than the ones consumed by the other two schemes as the error-tolerance e decreases. This trend suggests that the smoothing parameter should not decrease too fast (compared with Δ_j^{tol} or $O(\frac{1}{N_j})$), which is consistent with numerical results in Chapter 3 indicating that when sample is undersized, a too small value of ε does not show much help. The rationale behind this is that if ε_j converges to zero too fast, it might become too small (relative to N_j) to play an effective role of smoothing the stage SAA function. Hence, in the numerical experiments coming afterwards, the smoothing parameter ε_j is defined of the form $\varepsilon_j = c_\varepsilon / \sqrt{N_j}$, where c_ε is some constant greater than zero.

Comparison with the Smoothed SAA Methods

In this test, the performance of the RS-DS method is compared with the smoothed SAA methods proposed in Chapter 3. To have a fair comparison, the test problems employed in the experiments are the prob1 and prob2 used in the previous chapter. Based on the numerical results in Chapter 3, the values of sample size N and the smoothing parameter ε are closely related to the performance of the proposed methods. The parameters of the smoothed SAA methods we decide to take in this test are $N = 10$ and $N = 20$, and correspondingly $\varepsilon = 2$ and $\varepsilon = 1$. The reason of selecting these values is that according to the failure frequencies shown in Figure 3.10, both values of the ε show a comparatively good performance with the matching sample size in terms of the failure frequency.

As for the RS-DS method, the parameters of each stage j are specified in Table 4.4. By using the tuned parameters shown in the last column, we have

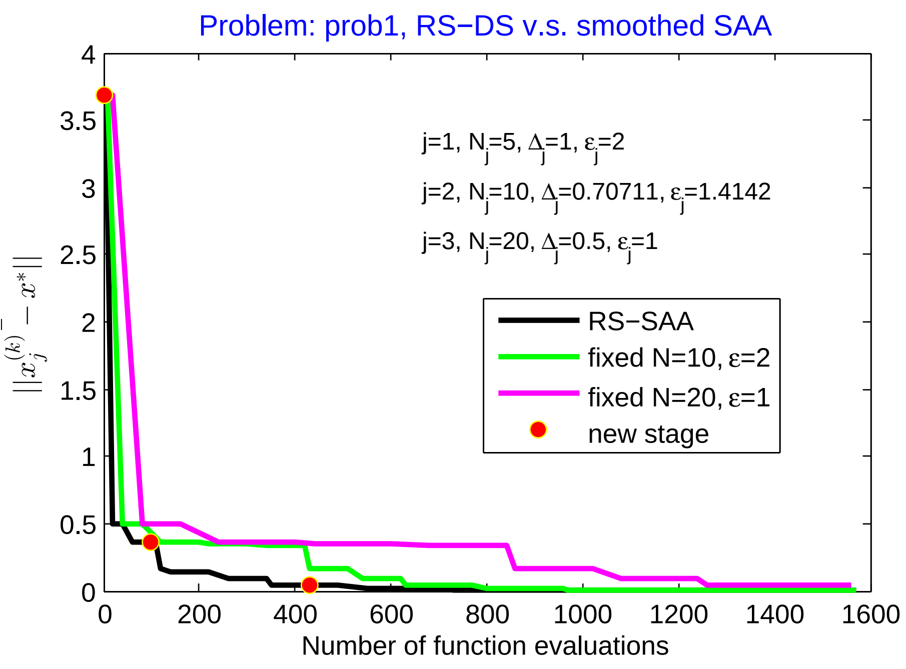
	$j = 1$	$j > 1$	tuned parameters
N_j	5	$c_N N_{j-1}$	$c_N = 2$
Δ_j^{tol}	1	$c_\Delta^{-0.5(j-1)} \Delta_1^{\text{end}}$	$c_\Delta = 0.1$
ε_j	2	c_ε / N_j	$c_\varepsilon = \sqrt{20}$

Table 4.4: Choice of parameters for the RS-DS method

$N_2 = 10$ and $N_3 = 20$, meaning that the RS-DS problem has the same size of samples as the fixed N smoothed SAA problem during the 2nd and 3rd stage. In addition, we have $\varepsilon_1 = 2$ and $\varepsilon_3 = 1$, hence both the RS-DS (in the early stage) methods and the smoothed SAA methods have similar range of smoothing parameter. Furthermore, since the default value of the mesh tolerance in Matlab is $1e-6$, we believe that $\Delta_1^{\text{end}} = 1$ should be a reasonable choice. All the other parameters involved in the direct search methods are set as the default values of the 'pattersearch' function in Matlab.

Starting from the same initial guess, the history paths of the RS-DS method and the fixed-N smoothed SAA methods from one single run are presented in Figure 4.3. The beginning of a new stage is highlighted in the figure.

It can be seen that by using a small sample in the beginning, the RS-DS method to the solution faster than the two smoothed SAA methods with fixed sample size. Though during its procedures, the RS-DS method shares some



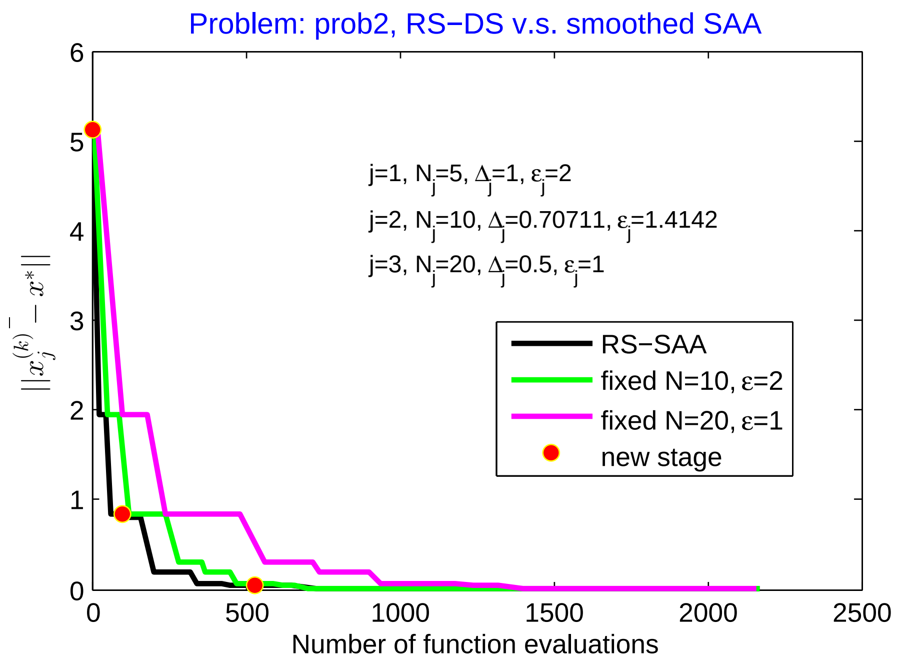


Figure 4.3: Iteration history of the RS-DS method and the smoothed SAA method

same values of N_j and ε_j with the fixed N cases, the performance of RS-DS method is better than both of the fixed-N paths. What helps the RS-DS method outperform is that a very small number of samples is used in the beginning of the algorithm, which enables the algorithm to save budget when the iterates are far away from the true solution.

The stability of the RS-DS method are also of interest. Though the smoothed SAA methods can sometimes work surprisingly well with a pretty small sample, this is dependent on the structure of the original objective function, the value of the smoothing parameter, and also the realization of the samples. As suggest in Figure 3.10, there is a chance that smoothed SAA methods fail to converge to the true solution when the sample is finite. In another words, when they work, the smoothed SAA methods can find the correct solution quickly and successfully save a significant amount of evaluations, but when they do not work, they may consume the budget but lead to a wrong solution. The issue is that without any information about the underlying problem, this possibility of 'failure' is difficult to estimate in advance. We present an example in Figure 4.4 to illustrate a case when the smoothed SAA methods get stuck.

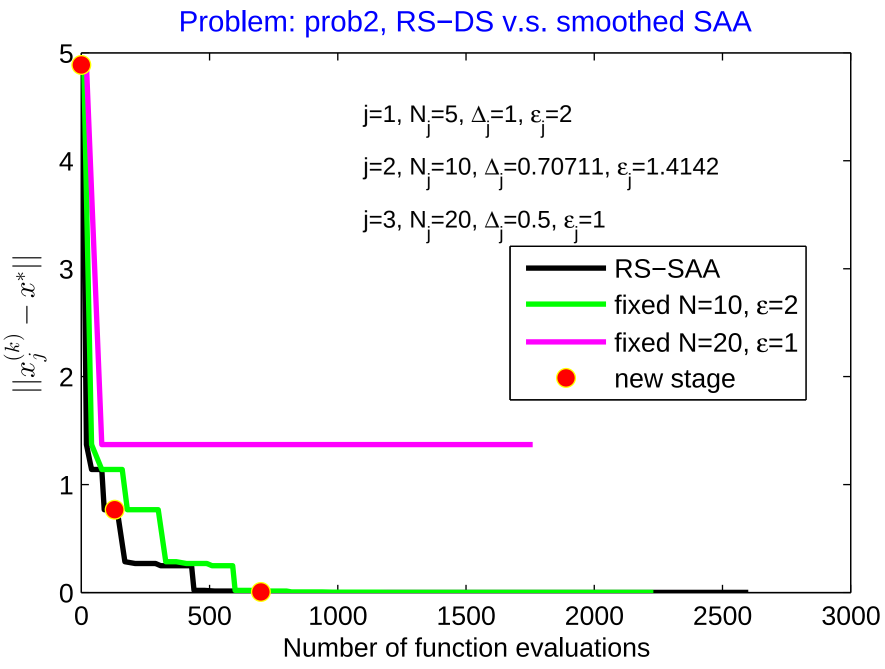


Figure 4.4: Smoothed SAA with a small sample fails to converge to the true solution

By using an increasing sequence of sample size, we the expect RS-DS method to relieve this issue by showing a better stability in terms of the capability of finding the correct solution with limited function evaluations. To investigate this, multiple simulations are ran and the average performance of the RS-DS method and the smoothed SAA methods are checked. The parameters of the two methods are set the same as in Table 4.4 and Figure 4.3. The initial guess $x^{(0)}$ for each run is randomly generated by a Latin hypercube sampling within the range of $B(x^*,5)$, and 1000 runs are implemented with a certain number of function evaluation budget. The quality of the solution is quantified by its distance to the true solution, mathematically denote as $\|x^*_{\text{budgt}} - x^*\|$. After 1000 runs, the statistical performance of the quality measure $\|x^*_{\text{budgt}} - x^*\|$ is given in Table 4.5.

		RS-DS	$N=10, \varepsilon=2$	$N=20, \varepsilon=1$
budget $= 600$	$\mu(\|x^*_{\text{budgt}} - x^*\|)$	0.4004	0.5006	0.6977
	$Var(\|x^*_{\text{budgt}} - x^*\|)$	0.7998	1.1210	0.8748
budget $= 2000$	$\mu(\|x^*_{\text{budgt}} - x^*\|)$	0.0939	0.4997	0.6621
	$Var(\|x^*_{\text{budgt}} - x^*\|)$	0.2073	1.1674	0.7744

Table 4.5: Statistical performance of the solutions obtained by the RS-DS method and the smoothed SAA methods based on 1000 runs

The two algorithms are tested under two values of simulation budget, which are 600 and 2000. The two schemes can be considered to measure a finite-time performance and an asymptotic performance correspondingly. The justification for the choice of the two values is the singular run result presented in Figure 4.3 and Figure 4.4. It can be roughly said that a budget of 2000 evaluations should be large enough for all of the three paths to converge.

From the statistical results in Table 4.5, two trends reflecting the advantages of the RS-DS method rather than the smoothed SAA methods can be observed. The first is that under both budget schemes , the RS-DS method performs better than the smoothed SAA counterparts in terms of both mean and variance of the measure $\|x^*_{\text{budgt}} - x^*\|$. This not only means that in average RS-DS method can get a better solution than the smoothed SAA methods, but also implies that former have a better stability rather than the later, which is consistent with our expectation. Another trend implied by Table 4.5 is that when the simulation budget increases from 600 to 2000, the performance of the two solutions obtained by the smoothed SAA methods does not change much. This is understandable in that for smoothed SAA methods, the more evaluations it consumed, the closer it will approach to the solution; however, this solution is the optimal solution of the SAA problem rather than the true objective problem. In another word, for the smoothed SAA cases, the error of the solution cannot be wiped off by increasing simulation budget once the a number of samples is finite and fixed. On the other hand, the performance of the RS-DS method improves substantially as the simulation budget is enlarged. This is due to the fact that the RS-DS method increases its sample size automatically in stage, making it possible for the algorithm to eliminate the function error and converge to the true solution asymptotically. By proceeding in this retrospective way, the RS-DS method outperforms the smoothed SAA methods.

4.4 Conclusions

We propose a specific type of retrospective-approximation algorithm using direct search methods to deal with stochastic simulation optimization problems. The convergence properties of the direct search methods make this combination reasonable and applicable, and further under certain conditions the parameters of the new algorithm can be determined following the guidelines proposed for a general retrospective-approximation algorithm.

This retrospective-approximation direct search methods are further developed by introducing a new smoothing parameter to solve probability problems in simulation-based optimization. We conduct numerical experiments to find out appropriate forms of the newly introduced parameter, and present numerical results showing that the new algorithm performs better than the standard SAA methods with smoothing technique.

Recommendations for future work on the proposed methods include: (i) proposing specific mathematical justification for the selection of the smoothing parameters; (ii) deriving convergence rate or asymptotic distributions of $\|\nabla f(x_j)\|$.

Chapter 5

Direct Search Methods Combined with Local Metamodeling

5.1 Introduction

The aim of this study is to improve the efficiency of direct search methods by using less evaluations than they normally take. More specifically, a local metamodeling methodology is introduced to combine with the standard direct search methods. The performance of the new algorithm is analyzed mathematically.

Our work is closely related to Custódio and Vicente (2007) and Custódio et al. (2010), both of which aim to improve the efficiency of the POLL stage of direct search methods by reducing the number of evaluations that each iteration consumes. Custódio and Vicente (2007) used simplex derivatives in the pattern search methods to reorder the visiting and the work has been extended into nonsmooth functions by Custódio et al. (2008). Custódio et al. (2010) used quadratic underdetermined metamodels in the SEARCH stage of the pattern search methods and showed good performance of the enhanced methods with numerical tests. In this work, several types of local metamodeling techniques are introduced to the standard POLL stage of direct search methods. This adaption can be applied to the POLL stage of any classical direct search methods without causing any extra computational time , and the existing convergence results of direct search methods are also preserved.

The key differences between this work and the two mentioned above lie in two folds. First, we use a set of local metamodeling techniques instead of simplex derivatives to realize the improvement, which is more flexible and general. Second, while Custódio and Vicente (2007) analyzed the effect of the improvement, the conditions required to guarantee the utility of the simplex derivatives are not stated and the improvement is not mathematically justified either. On the other hand, we analyze the our algorithm by proposing two types of measures to quantify the performance of the algorithm from an asymptotic point of view and a finite-time point of view, respectively. The measures are closely dependent on the local metamodeling techniques used in the proposed algorithm and the nature of the metamodels plays an important role in justifying the utility of the algorithm.

5.2 Some Metamodeling Techniques

We turn our attention to a set of local metamodeling techniques in this section. Throughout this chapter, we denote a design set with p design points as $DS = \{x^1, x^2, \cdots, x^p\}$. We will introduce three types of commonly used metamodeling techniques . Some mathematical properties of these techniques, which will be used in our mathematical analysis, will be presented as well.

5.2.1 Linear Interpolation and Regression

Let $m_L(x)$ denote a first-order polynomial metamodeling function, and $x = \{x(1), x(2), \cdots, x(d)\} \in \mathbb{R}^d$. Then $m_L(x)$ can be expressed in the following form:

$$m_L(x) = \beta_0 + \beta_1 x(1) + \cdots + \beta_n x(d).$$

We say that metamodel $m(x)$ interpolates the true function $f(x)$ at a given point x' if $m(x') = f(x')$. Let p denote the number of design points, and $\Phi_L(x) := \{1, x(1), x(2), \cdots, x(d)\} \in \mathbb{R}^{d+1}$ denote the linear polynomials of degree 1. In the case of linear interpolation, let $m_L(x)$ interpolate $f(x)$ at the p points in the

DS, i.e., $m_L(x^i) = f(x^i), i = 1, \cdots, p$. Mathematically

$$\begin{bmatrix} \Phi_L(x^1) \\ \Phi_L(x^2) \\ \vdots \\ \Phi_L(x^p) \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = \begin{bmatrix} f(x^1) \\ f(x^2) \\ \vdots \\ f(x^p) \end{bmatrix}. \tag{5.1}$$

The linear parameter β can be obtained by solving the above linear system as long as $p = d$ and the design set is poised, meaning that the corresponding design matrix, the matrix on the left in (5.1), is nonsingular (Conn et al., 2009, pg. 24).

5.2.2 Minimum Frobenius Norm Model

A quadratic polynomial model can be considered as the simplest yet often the most efficient nonlinear metamodel (Conn et al., 2009, pg. 37). Minimum Frobenius norm model, denoted as m_F, is essentially a type of quadratic polynomial interpolation model. Let $\mathbb{R}^d$ denote the d-dimensional real space and $\mathbb{N} = \{0, 1, 2, \cdots\}$, then define $\mathbb{N}_0^d$ to be

$$\mathbb{N}_0^d = \{z = (z_1, z_2, ..., z_d) \in \mathbb{R}^d : z_i \in \mathbb{N}, i = 1, 2, ..., d\}.$$

Let $\Phi(x) \in \mathbb{R}^{(d+1)(d+2)/2}$ be a vector of monomials which will be appear in the m_F model to be presented later. Mathematically, the lth element of $\Phi(x)$, denoted as $\Phi^l(x)$, can be expressed as

$$\Phi^\ell(x) = \frac{1}{(\alpha_\ell)!} x^{\alpha_\ell}, \quad l = 0, \cdots, \frac{(d+1)(d+2)}{2},$$

where $\alpha_\ell = \big(\alpha_\ell(1), \cdots, \alpha_\ell(d)\big) \in \mathbb{N}_0^d$, such that $\sum_{j=1}^d \alpha_\ell(j) \leq 2$, $(\alpha^\ell)! = \prod_{j=1}^d \big(\alpha_\ell(j)!\big)$, and $x^{\alpha_\ell} = \prod_{j=1}^d x(j)^{\alpha_\ell(j)}$. More intuitively, for a given $x \in \mathbb{R}^d$,

$$\Phi(x) = \{1, x(1), x(2), \cdots, x(d), x(1)^2/2, x(1)x(2), \cdots, x(d-1)x(d), x(d)^2\}.$$

When $p = (d+1)(d+2)/2$, we let m_F interpolate f at all design points and solve the following the system of linear equations:

$$\begin{bmatrix} \Phi(x^1) \\ \Phi(x^2) \\ \cdots \\ \Phi(x^p) \end{bmatrix} \beta = \begin{bmatrix} f(x^1) \\ f(x^2) \\ \cdots \\ f(x^p) \end{bmatrix}, \tag{5.2}$$

where β is a vector in $\mathbb{R}^{(d+1)(d+2)/2}$. Recall that the definition of Φ_L given in Section 5.2.1. Let $\Phi_Q(x)$ be the complementary set of $\Phi(x)$ and $\Phi_L(x)$, in which case we have $\Phi(x) = \big(\Phi_L(x), \Phi_Q(x)\big)$. Further denote the corresponding parameters of $\Phi_L(x)$ and $\Phi_Q(x)$ as β_L and β_Q, thus we have $\beta = (\beta_L, \beta_Q)$. Same as linear interpolation, a solution of β can be obtained as long as the design matrix on the left in (5.2) is nonsingular.

One difficulty in determining a quadratic interpolation is that there may not be enough well-poised design points, in which case the left hand side is nonsingular. Minimum Frobenius norm (MFN) model provides an option when the design points are large enough for a linear regression but not enough for a quadratic interpolation. When $d < p < (d+1)(d+2)/2$, the information of curvature is appreciated driven by the belief that the actual accuracy of a underdetermined quadratic model is better than that of a purely linear model.

Let $\Phi_L(x) := \{1, x(1), x(2), \cdots, x(d)\}$ and $\Phi_Q(x) := \{x(1)^2/2, x(1)x(2), \cdots x(d)^2\}$ be the linear and quadratic natural basis, and define $\beta = (\beta_L, \beta_Q)$, where β_L and β_Q are the correspondent linear and quadratic part of the coefficient vector. Then the MFN parameter β is defined as the solution of the following optimization problem:

$$\begin{array}{ll} \min & \frac{1}{2}\|\beta_Q\|^2 \\ \text{s.t.} & \begin{bmatrix} \Phi_L(x^1) \\ \Phi_L(x^2) \\ \cdots \\ \Phi_L(x^p) \end{bmatrix} \beta_L + \begin{bmatrix} \Phi_Q(x^1) \\ \Phi_Q(x^2) \\ \cdots \\ \Phi_Q(x^p) \end{bmatrix} \beta_Q = \begin{bmatrix} f(x^1) \\ f(x^2) \\ \cdots \\ f(x^p) \end{bmatrix}. \end{array} \tag{5.3}$$

MNF model gets its name due to the fact that minimizing the norm of β_Q is equivalent to minimizing the Frobenius norm of the Hessian of $m_F(x)$. Once we get the solution of β, the MNF estimator can be obtained by:

$$m_F(x) = \Phi_L(x)\beta_L + \Phi_Q(x)\beta_Q. \tag{5.4}$$

5.2.3 Kriging Models

Kriging model is a geostatistical method to infer the value of an unobserved location from available samples Simple kriging assumes that the true function value $f(x)$ is a sample-path of a random process dependent on x. Mathematically,

$$f(x) = \beta + z(x),$$

where $z(x)$ follows a stationary Gaussian process of mean 0 and variance σ_x^2. The covariance between two sample points x_1 and x_2 is dependent on the distance between the two points such that $\text{Cov}(z(x),z(y)) = R\big(\|x-y\|\big)$ for some radial basis function R. For a simple notation, we will use $c(x,y)$ to refer $\text{Cov}\big(z(x),z(y)\big)$ afterwards.

Given a set of design points DS, the kriging model m_K interpolates the observed values on them, and estimates the values of an unvisited point x as a linear combination of the available observations:

$$m_K(x) = \sum_{i=1}^{p} \lambda_i(x) f(x^i). \tag{5.5}$$

The weights $\lambda_i(x),\ i = 1, \cdots, p$ are chosen such that the kriging variance is to be minimized subject to the unbiasedness condition. In this sense, kriging can be considered as a type of least squares estimation algorithm and it reaches the best linear unbiased estimator. The kriging variance is a popular measure to quantify the reliance of kriging, which is

$$\begin{aligned} \sigma_K^2(x) \quad &:= \quad Var\Big(m_K(x) - f(x)\Big) \\ &= \quad \sum_{i=1}^{p}\sum_{j=1}^{p} \lambda_i(x)\lambda_j(x)c(x^i, x^j) + Var\big(f(x)\big) - 2\sum_{i=1}^{p} \lambda_i(x)c(x^i, x). \end{aligned}$$

Given the unbiased estimator and kriging variance, a confidence interval of each prediction can be obtained. This information of confidence interval plays an important role later when the performance of kriging local metamodel is analyzed quantitatively.

In addition to the kriging variance, Yamamoto (2000) has propose interpolation variance as another measure to quantify the local accuracy of kriging estimation, which is

$$\sigma_I(x) = \sum_{i=1}^{p} \lambda_i \big[f(x^i) - m_K(x)\big]^2.$$

Compared to the kriging variance, interpolation variance is considered to be a better measure for quantifying the reliability of kriging in that it is objective-value dependent, while kriging variance is only location dependent. Due to this reason, interpolation variance is selected to measure the reliability of the kriging local model in this work. In the proposed algorithm, kriging model is used for a local estimation instead of for a global fitting, hence only the visited points located in the neighborhood of the estimate point will be used. Therefore, a reliability measure that can capture more available information should be more helpful.

One thing needs to be pointed out is that interpolation variance is the variance conditional to the p design points ((Froidevaux, 1993)), which is

$$\sigma_I(x) = E\left[(f(x) - m_K(x))^2 \middle| \left\{f(x_{(i)}), i = 1, \cdots, p\right\}\right]. \tag{5.6}$$

Because ordinary kriging variance is equal to the mean squared error, which is $\sigma_K(x) = E\left[(f(x) - \hat{f}(x))^2\right]$, we have $\sigma_K(x) = E[\sigma_I(x)]$.

The two measures of kriging's reliability are highlighted here because they will be employed later to analyze the performance of local metmodels. To the best of our knowledge, kriging has been little used in a local sense, but it does exhibit some properties that make employing it as a local metamodel applicable. Since the set of design points is limited in size, $\Delta^{(k)}$ goes to zero as algorithm progresses. In addition, $|f(x^i) - f(x^j)|$ for $i, j \in \mathbb{Z} \leq p$ also decreases under some conditions, which provides a chance to quantify the asymptotic performance of the metamodels in our analysis.

The software package DACE (Design and Analysis of Computer Experiments) is a Matlab toolbox to construct a kriging approximation model based on data from a computer experiment, and to use this approximation model as a surrogate for the computer model. The DACE toolbox will be used in the numerical experiments part and a good survey on DACE is referred to Sacks et al. (1989).

5.3 Framework of Improved Algorithms

We now present a description of the new direct search methods which are combined with some local metamodeling techniques mentioned above. The motivation of the adaption is that in the normal GSS, though the direct set is generated using different techniques, the candidates points determined by this direct set are normally visited in a predetermined order (Conn et al., 2009). We want to make the maximum use of the available information to decide the visiting order more wisely so that we can reduce some number of evaluations in each iteration. For this purpose, we construct a local metamodel at the beginning of each iteration and keep it updated whenever new information is available. With the help of the local metamodels, we can identify the most potential candidate by evaluating some estimation measures. More specifically, the framework of our algorithm is illustrated in Figure 5.1.

```
I. Initialization. Use the same initialization in GSS algorithms. Initialize information set $\mathscr{I}_0$.
II. Algorithm. For each iteration $k = 0, 1, 2, \cdots$, set nested index $j = 1$,
  • Step I (next iterate selection) Define candidate set $C^{(k)} = \{x_i^{(k)},\ i = 1, \cdots, n_k\}$, where $x_i^{(k)}$ is defined as in Figure 2.1. Set unvisited candidate set $UC^{(k)} = C^{(k)}$.
  • Step II(local metamodels) Determine design set $DS^{(k_j)}$ by following Equation (5.7). Construct a local metamodel $m^{(k_j)}$ based on $DS^{k_j}$ using the techniques described in Section 5.2. Identify the most potential candidate, $x_{mp}^{(k)}$, using Equation (5.8).
  • Step III(update models or algorithm parameters) Evaluate $f(x_{mp}^{(k)})$:
    – (i) (Success) If $f(x_{mp}^{(k)}) < f(x^{(k)}) - \rho(\Delta^{(k)})$, then
      ∗ set $x^{(k+1)} = x_{mp}^{(k)}$, and $\Delta^{(k+1)} = \gamma\Delta^{(k)}$, where $\gamma \geq 1$.
      ∗ update $\mathscr{I}_k = \mathscr{I}_k \cup \{(x_{mp}^{(k)}, f(x_{mp}^{(k)}))\}$.
      ∗ iteration terminates, $k \leftarrow k+1$.
    – (ii) (Failure) Otherwise,
      ∗ update $\mathscr{I}_k = \mathscr{I}_k \cup \{x_{mp}^{(k)}, f(x_{mp}^{(k)})\}$.
      ∗ update $UC^{(k)} = UC^{(k)} - \{x_{mp}^{(k)}\}$:
        · if $UC^{(k)} \neq \emptyset$, $j \leftarrow j+1$, goto Step II and repeat.
        · if $UC^{(k)} = \emptyset$, then $x^{(k+1)} = x^{(k)}$, $\Delta^{(k+1)} = \theta\Delta^{(k)}$, where $0 < \theta \leq \theta_{\max} \leq 1$.
    – Stopping criterion check:
      ∗ if $\Delta^{(k+1)} < \Delta_{\text{tol}}$, then terminate.
```

Figure 5.1: GSS algorithm with local metamodeling techniques.

As Figure 5.1 says, in the beginning of iteration k, a local metamodel is constructed first. A design set $DS^{(k)}$ is identified depending on whether the previous iteration is successful or not. More specifically, we define $DS^{(k)}$ as follows:

$$DS^{(k)} = \begin{cases} \{x^{(k)}\} \cup C^{(k-1)}, & \text{, if } (k-1)\text{th iteration is a failure;} \\ (n_k+1) \text{ points in } \mathscr{I}_k & \\ \quad \text{that are closest to } x^{(k)} & \text{, otherwise} \end{cases} . \tag{5.7}$$

Note that according to the definition of $DS^{(k)}$, it can be seen that no matter in which case, the current center point $x^{(k)}$ is always included in the corresponding design set.

Although metamodel is updated within one iteration once new candidate is visited, we abuse $m^{(k)}$ to refer the $m^{(k_j)}$ defined in Figure 5.1 afterwards for a simplification. The estimation measures used to identify the most potential candidate, $x_{mp}^{(k)}$, are different for different types of local metamodeling techniques. Mathematically, it can be expressed as following:

$$x_{mp}^{(k)} := \begin{cases} \arg\min_{x \in UC^{(k)}} m^{(k)}(x) & , \; m^{(k)} \in \{m_L, m_F\}; \\ \\ \arg\max_{x \in UC^{(k)}} \Pr\{f(x) < f(x^{(k)}) - \rho(\Delta^{(k)})\} & , \; m^{(k)} \in \{m_K\}. \end{cases} \tag{5.8}$$

For the second category $m^{(k)} \in m_K$, the probability measure implies the chance that the unvisited point outperforms than $x^{(k)}$ for at least $\rho(\Delta^{(k)})$ based on the available interpolation points and together with kriging metamodels. The reason that we choose probability interval instead of mean value as the measure for kriging meatmodels is that when compare a candidate with the current center point, the only thing that matters is whether the performance on the candidate is better than that of current best at least for a value of $\rho(\Delta^{(k)})$, while the actual value of the function gap does not matter. $\Pr\{f(x) < f(x_k)\}$ can be obtained from the unbiased estimator $m_K^{(k)}(x)$, kriging variance $\sigma_k(x)$, and together with the Gaussian assumption of the model. This probability information revealed by kriging metamodels will play an important role in the performance analysis of our adapted algorithm.

The local metamodeling techniques used in the new algorithm can be seamlessly embedded in a wide range of direct search methods. Also, the existing convergence results of the direct search methods are not affected. Furthermore, because the improved algorithm does not require extra evaluations in the implementation, it improves the efficiency of the algorithm by making more use of local information and reducing the evaluations that it would cost to find a better point. This saving would be significant especially when the problem size is big, or the running simulation is extremely costly. Note that we only expect local metamodels to help in successful iterations, in which cases there exist at least one better point. Otherwise, all the candidate points have to be visited before the algorithm moves to the next iteration anyway.

5.4 Asymptotic Performance Analysis

In this section, we study the asymptotic performance of local metamodels that are embedded in the standard direct search methods. By saying ‘asymptotic performance’, we mean a situation where the simulation budget is sufficient enough to get an optimal solution. More specifically, we want to learn how does the accuracy of local metamodels change as meshsize $\Delta^{(k)}$ decreases to zero. Remember that the embedded metamodels would not effect the convergence results of the algorithm. They only aim to improve the algorithm efficiency by providing with a wiser visiting order to the candidate points.

Three types of metamodeling techniques mentioned before, m_L, m_F, m_K are to be looked into. Linear metamodels m_L and minimum Frobenius norm models m_F are considered in the same category since they are both interpolation or regression polynomial models but just with different orders.

5.4.1 Performance of Local Interpolation and Regression

We study the asymptotic performance of m_L and m_F in this section. We firstly give an assumption on the local metamodels to guarantee their estimation accuracy quantitatively.

Assumption 10. *Assume that for all the point $x \in B(x^{(k)}; r\Delta^{(k)})$, the error bound of the metamodel estimation $m^{(k)}(x)$ to the true value $f(x)$ is upper bounded as*

$$|f(x) - m^{(k)}| \leq \kappa \cdot (\Delta^{(k)})^{\alpha},$$

for $\alpha \geq 2$, where κ is a constant depending on the $f(\cdot)$ and geometrical properties of the design set $DS^{(k)}$.

As a matter of fact, a fully linear model satisfies Assumption 10 with $\alpha = 2$ and a fully quadratic model satisfies it with $\alpha = 3$ (Conn et al., 2009, Chapter 6.1). As for the design points, Custódio and Vicente (2007) have shown that $DS^{(k)}$ is guaranteed to be poised if $DS^{(k)} = \{x^{(k)}\} \cup C^{(k-1)}$, where the set of candidate points $C^{(k-1)}$ is defined as in Figure 5.1. According to our definition to the design set (5.7) and Conn et al. (2009, Theorem 2.11), it can be shown that m_L and m_F would satisfy Assumption 10 with $\alpha = 2$ whenever the previous iteration is a failure. Therefore, in this subsection, we would focus on the performance of m_L and m_F.

We will check the asymptotic performance of metamodels conditioned on the form of forcing function $\rho(\Delta^{(k)})$. Recall that ρ can be either constantly equal to zero, or some positive-valued function satisfying $\lim_{\Delta^{(k)}\to 0} \rho(\Delta^{(k)})/\Delta^{(k)} = 0$. Both cases will be analyzed, respectively.

Convergence analysis for the case with $\rho \equiv 0$

Assumption 11. *Let $x_i^{(k)} = x^{(k)} + \Delta^{(k)} d_i^{(k)}, i = 1, \cdots, n_k$, and for the objective function $f(x)$, define $OD^{(k)} := \min\left\{|f(x^{(k)}) - f(x_i^{(k)})| : \ |f(x^{(k)}) - f(x_i^{(k)})| \neq 0,\ i = 1, \cdots, n_k\right\}$. Furthermore, $\lim_{k\to\infty} \dfrac{{\Delta^{(k)}}^{\alpha}}{OD^{(k)}} = 0$.*

Note that $\left(\Delta^{(k)}\right)^{\alpha}$ quantifies the metamodel estimation error, whereas OD_k can be considered as a measure of optimality error. Assumption 11 basically requires that the metamodel should be accurate enough such that its estimation error would be dominated by optimality error asymptotically. We may use the mean value theorem for a better description. For some t between 0 and 1, we have

$$\begin{aligned} f(x^{(k)}) - f(x_i^{(k)}) &= (x^{(k)} - x_i^{(k)}) \cdot \nabla f\left((1-t)x^{(k)} + t x_i^{(k)}\right) \\ &= \Delta^{(k)} {d_i^{(k)}}^T \nabla f\left((1-t)x^{(k)} + t x_i^{(k)}\right). \end{aligned}$$

One sufficient condition for Assumption 11 is that at the region near stationary points, ∇f should converge to zero slower than $O(\alpha - 1)$. For models like m_L and m_F with $\alpha = 2$, this is to require that ∇f should converge to zero slower than linear rate near the local solutions.

It can be seen that Assumption 11 is strong and typically violated in practice. However, later in the numerical experiments, we also test a wide variety of problems that do not follow Assumption 11. The results show that the performance with metamodels are still better than those without metamodels.

By the following theorem, we will show that under Assumption 10 and Assumption 11, asymptotically, metamodel gives a correct ranking of candidate points and center point at successful iterations as the meshsize decreases to zero.

We first define an indicator variable to evaluate the consistency of the metamodel ranking and true ranking. Define subsequence $\{k_l : l \geq 1\} \subset \{k \geq 1\}$ be a sequence of successful iterations. The reason why only successful iterations are considered is that for unsuccessful iterations all the candidate points have to be visited before the algorithm moves to the next iteration anyway. Let the indicator function 1_{cm} denote the correctness of local metamodel, which is

$$1_{\text{cm}}(x^{(k_l)}) = \begin{cases} 1 \ , \ \text{if } \operatorname{sgn}\left(m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right) = \operatorname{sgn}\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right) \\ \\ 0 \ , \ \text{otherwise,} \end{cases} .$$

Notice that the indicator value of 1_{cm} implies that the sign of $\left(m^{(k_l)}(x_{mp}^{(k_l)})\right) - f(x^{(k_l)})$ is consistent with the sign of $f(x_{mp}^{(k_l)}) - f(x^{(k_l)})$, meaning that meatmodel estimation is accurate enough for a correct ranking between candidate point and center point when the candidate is really better than the center point. We give the limit value of 1_{cm} indicator function by stating the theorem below.

Theorem 2. *Suppose that Assumptions 10 and 11 hold. Then,* $\lim_{l\to\infty} 1_{cm}(x^{(k_l)}) = 1$.

Proof. By Assumption 10, $|m^{(k)}(x) - f(x)| \leq \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha}$, which implies

$$f(x) - \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha} \leq m^{(k)}(x) \leq f(x) + \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha}. \tag{5.9}$$

Let $L_{k_l} = \left(m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right)\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right)$, then

$$\begin{aligned} L_{k_l} \ \geq \ \min\Big\{ & \left(f(x_{mp}^{(k_l)}) - \kappa \cdot (\Delta^{(k)})^{\alpha} - f(x^{(k_l)})\right)\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right), \\ & \left(f(x_{mp}^{(k_l)}) + \kappa \cdot (\Delta^{(k)})^{\alpha} - f(x^{(k_l)})\right)\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right)\Big\}. \end{aligned} \tag{5.10}$$

By Assumption 11, there exists a positive L such that for all $l > L$, the sign of the right-hand side of (5.10) $= \operatorname{sgn}\left(\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)})\right)^2\right)$. Since k_l denotes the subsequence of successful iterations, we have $f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) < 0$, meaning that for all $l > L$, L_{k_l} is positive. According to the definition of L_{k_l}, a positive value of L_{k_l} implies that the signs of $m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)})$ and $f(x_{mp}^{(k_l)}) - f(x^{(k_l)})$

are consistent. According to the definition of $1_{cm}(x^{(k_l)})$, it can be obtained $\lim_{l\to\infty} 1_{cm}(x^{(k_l)}) = 1$.

□

Convergence analysis for the case with $\rho(\Delta^{(k)}) \neq 0$

The conditions required to get Theorem 2 is strong. The reason is that when $\rho \equiv 0$ metamodel approximations should be pretty accurate so that they can make a correct ranking between $f(x^{(k_l)})$ and $f(x_{mp})$. The strong condition needed can be relaxed when ρ is a positive-valued function rather than zero. When analyzed, the positive ρ could be considered as a tolerable estimation error for making a correct ranking between the current center point and a better candidate. Compared to the zero tolerance case with $\rho \equiv 0$, in the case of $\rho \geq 0$ the asymptotic performance of metamodels can be guaranteed under much relaxed conditions.

Assumption 12. *For some $\alpha > 0$, the forcing function $\rho(\Delta^{(k)})$ satisfies*

$$\lim_{\Delta^{(k)}\to 0} \frac{\left(\Delta^{(k)}\right)^{\alpha}}{\rho(\Delta^{(k)})} = 0. \tag{5.11}$$

Recall that a prerequisite for $\rho(\cdot)$ is that $\lim_{\Delta^{(k)}\to 0} \dfrac{\rho(\Delta^{(k)})}{\Delta^{(k)}} = 0$. Combining this condition with Assumption 12, one of the simplest form of the forcing function is $\rho(\Delta^{(k)}) = (\Delta^{(k)})^{\gamma}$, where $1 < \gamma < \alpha$.

We will show that if Assumption 12 is satisfied, for the sequence of successful iterations $\{k_l : k \geq 1\} \subset \{k \geq 1\}$, the model error of m_L and m_F is asymptotically dominated by the objective difference between $x_{mp}^{(k_l)}$ and $x^{(k_l)}$, meaning that metamodel can identify $x_{mp}^{(k_l)}$ successfully in one shot.

Proposition 8. *Suppose that Assumptions 10 and 12 hold. Then,*

$$\lim_{l\to\infty} \frac{|f(x_{mp}^{(k_l)}) - m(x_{mp}^{(k_l)})|}{f(x^{(k_l)}) - f(x_{mp}^{(k_l)})} = 0.$$

Proof. By Assumption 10, for any $l \geq 1$,

$$0 \leq \frac{|f(x_{mp}^{(k_l)}) - m(x_{mp}^{(k_l)})|}{f(x^{(k_l)}) - f(x_{mp}^{(k_l)})} \leq \frac{\kappa \cdot (\Delta^{(k_l)})^{\alpha}}{\rho(\Delta^{(k_l)})}. \tag{5.12}$$

By Assumption 12, the upper bound in (5.12) converges to zero as $\Delta^{(k)}$ goes to zero. Thus the proof is done. $\square$

Based on Proposition 8, a similar asymptotic result to the case of $\rho \equiv 0$ can be obtained.

Let the indicator function 1_{cm} denote the correctness of local metamodel in the cases of $\rho \neq 0$, which is

$$\bar{1}_{\text{cm}}(x^{(k_l)}) = \begin{cases} 1 \ , \ \text{if } \operatorname{sgn}\left(m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right) \\ \qquad\qquad = \operatorname{sgn}\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right); \\ 0 \ , \ \text{otherwise.} \end{cases}$$

Theorem 3. *Suppose that Assumptions 10 and 12 hold. Then,* $\lim_{l\to\infty} \bar{1}_{cm}(x^{(k_l)}) = 1$.

Proof. By Assumption 10, $|m^{(k)}(x) - f(x)| \leq \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha}$, which is

$$f(x) - \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha} \leq m^{(k)}(x) \leq f(x) + \kappa \cdot \left(\Delta^{(k)}\right)^{\alpha}.$$

Let $L_{k_l} = \left(m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right)\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right)$, then

$$\begin{aligned} L_{k_l} \ \geq \ \min\Big\{ & \left(f(x_{mp}^{(k_l)}) - \kappa \cdot (\Delta^{(k)})^{\alpha} - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right) \\ & \quad \times \left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right), \\ & \left(f(x_{mp}^{(k_l)}) + \kappa \cdot (\Delta^{(k)})^{\alpha} - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right) \\ & \quad \times \left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right)\Big\}. \end{aligned} \tag{5.13}$$

By Assumption 12, there exists a positive L such that for all $l > L$, $\operatorname{sgn}\big(\text{RHS of (5.13)}\big) = \operatorname{sgn}\Big(\left(f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})\right)^2\Big)$. Since k_l denotes the subsequence of successful iterations, we have $f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)}) < 0$, meaning that for all $l > L$, L_{k_l} is positive. According to the definition of L_{k_l}, a positive value of L_{k_l} implies that the signs of $m^{(k_l)}(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})$ and $f(x_{mp}^{(k_l)}) - f(x^{(k_l)}) + \rho(\Delta^{(k_l)})$ are consistent. According to the definition of $\bar{1}_{cm}(x^{(k_l)})$, it can be obtained $\lim_{l\to\infty} \bar{1}_{cm}(x^{(k_l)}) = 1$.

$\square$

Theorem 3 implies that for the sequence of successful iterations, asymptotically metamodels can identify a better candidate in just one shot.

5.4.2 Asymptotic Performance of m_K

We have introduced two different measures, kriging variance σ_K and interpolation variance σ_I, to quantify the reliance of kriging in Section 5.2.3. In this section, we will check the asymptotic performance of m_K by studying the limit values of these two measure.

Assumption 13. *$f(\cdot)$ is Lipschitz continuous with some constant κ_L, i.e., $|f(x)-f(y)| \le \kappa_L \|x-y\|$ for any $x, y \in \mathbb{R}^d$.*

Proposition 9. *Suppose that Assumption 13 holds. Then*

$$\lim_{k\to\infty} \sigma_I(x^{(k)}) = 0.$$

Proof. We firstly rewrite the interpolation variance $\sigma_I(x)$ as follows:

$$\begin{aligned}
\sigma_I(x) &= \sum_{i=1}^{p} \lambda_i \left[f(x_{(i)}) - m_K(x)\right]^2 \\
&= \sum_{i=1}^{p} \lambda_i \left[f(x_{(i)}) - \sum_{i=1}^{p} \lambda_i f(x_{(i)})\right]^2 \\
&= \sum_{i=1}^{p} \lambda_i \Bigg[(1-\lambda_i) f(x_{(i)}) - \sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)})\Bigg]^2.
\end{aligned}$$

Define $\Delta_k^\delta := \max_{i\neq j} \|x_{(i)} - x_{(j)}\|$. Since $\sum_i \lambda_i = 1$, two bounds of $\sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)})$ can be obtained based on Assumption 13:

$$\begin{aligned}
\sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)}) &\le \sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j \left(f(x_{(i)}) + \kappa_L \|x_{(i)} - x_{(j)}\|\right) \le (1-\lambda_i)\cdot\left(f(x_{(i)}) + \kappa_L \Delta_k^\delta\right); \\
\sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)}) &\ge \sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j \left(f(x_{(i)}) - \kappa_L \|x_{(i)} - x_{(j)}\|\right) \ge (1-\lambda_i)\cdot\left(f(x_{(i)}) - \kappa_L \Delta_k^\delta\right).
\end{aligned}$$

Then, we have

$$\begin{aligned}
(1-\lambda_i)f(x_{(i)}) - \sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)}) &\geq (1-\lambda_i)f(x_{(i)}) - (1-\lambda_i)\big(f(x_{(i)}) + \kappa_L \Delta_k^\delta\big) \\
&= -\kappa_L \Delta_k^\delta (1-\lambda_i), \\
(1-\lambda_i)f(x_{(i)}) - \sum_{\substack{j=1,\cdots,p \\ \& \ j\neq i}} \lambda_j f(x_{(j)}) &\leq (1-\lambda_i)f(x_{(i)}) - (1-\lambda_i)\big(f(x_{(i)}) - \kappa_L \Delta_k^\delta\big) \\
&= \kappa_L \Delta_k^\delta (1-\lambda_i).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\sigma_I(x) &= \sum_{i=1}^{p} \lambda_i \big[f(x_{(i)}) - m_K(x)\big]^2 \\
&\leq \sum_{i=1}^{p} \lambda_i \Big(\kappa_L \Delta_k^\delta (1-\lambda_i)\Big)^2 \\
&= (\kappa_L \Delta_k^\delta)^2 \sum_{i=1}^{p} \lambda_i (1-\lambda_i)^2 \\
&\leq p(\kappa_L \Delta_k^\delta)^2, \qquad (5.14)
\end{aligned}$$

where the last inequality is from $0 < \lambda_i < 1$. Since $\Delta_k^\delta \to 0$ as $k \to \infty$, we have $\lim_{k\to\infty} \sigma_I x^{(k)} = 0$. □

Recall that it has been shown in Section 5.2.3 that $\sigma_K(x) = E[\sigma_I(x)]$. Since $\sigma_I(x)$ is upper bounded as in (5.14), according to Dominated Convergence Theorem, $\lim_{k\to\infty} \sigma_K(x^{(k)}) = 0$.

5.5 Finite-time Performance Analysis

In this section, we study the finite-time performance rather than asymptotic one. By saying finite-time performance, we mean the quality of the solutions can be obtained with limited simulation budget. We compare our algorithm which uses local metamodeling techniques with the direct search methods which visit candidates in a random order. For the regression metamodels, we show that the probability of detecting a better candidate is higher in the first scheme. As for the kriging, the number of evaluations consumed by the two schemes are analyzed and it is shown that evaluations can be reduced by using local kriging in direct search methods.

5.5.1 Regression Metamodels

The idea of prediction interval in the statistical inference will be used in the following analysis. A prediction interval is an estimate of an interval in which future observations will fall, with a certain probability, based on what has already been observed, which is different from estimation in that the later is to estimate a parameter, like mean or variance, instead of individual observation.

In ordinary linear regression theory, where residuals are assumed to be independent and with the same variance, the best prediction is consistent with a linear regression estimation (Stein and Corsten, 1991). Furthermore, Stein and Corsten (1991) point out that the possible future observation on a point x can be considered to follow a normal distribution with mean of $m(x)$ and variance of $\sigma(x) := 1 + x(X'X)^{-1}x$, where X is the design matrix used in the metamodel construction. Based on this assumption, for any given a, if the metamodel value $m(x) > a$, we have $\Pr\{f(x) < a\} < 1/2$ and if $m(x) < a$, we have $\Pr\{f(x) < a\} > 1/2$.

Let S_r denote the event of finding a better point in the first trial by the algorithm which visits candidates randomly, and S_m denote the event of finding a better point in the first trial by our algorithm which uses regression metamodels. Let $C^{(k)}$ denote the set of candidate points as defined in Figure 5.1 and $n_1 = |C^{(k)}|$ denote its cardinality. Let $BC^{(k)} = \{x \in C^{(k)} : m(x) < f(x^{(k)})\}$ denote the set of candidate points with a better meatmodel estimation than the current center point and $n_2 = |BC^{(k)}|$ denote its cardinality. Then, we have

$$\begin{aligned}
\Pr\{\mathrm{S}_r\} &= \frac{1}{n_1} \cdot \sum_{x_{c_i} \in C^{(k)}} \Pr\left\{ f(x_{c_i}) < f(x^{(k)}) \right\} \\
&= \frac{1}{n_1} \Bigg(\sum_{x_{c_i} \in BC^{(k)}} \Pr\left\{ f(x_{c_i}) < f(x^{(k)}) \right\} \\
&\qquad + \sum_{x_{c_i} \in \left(C^{(k)} - BC^{(k)}\right)} \Pr\left\{ f(x_{c_i}) < f(x^{(k)}) \right\} \Bigg).
\end{aligned} \tag{5.15}$$

Selecting a candidate point based on the metamodel values, and assuming that the number of candidate points with a better metamodel value is positive, we have

$$\Pr\{S_m\} = \frac{1}{n_2} \cdot \sum_{x_{c_j} \in BC^{(k)}} \Pr\left\{ f(x_{c_j}) < f((x^{(k)}) \right\}.$$

Under the assumptions of independent prediction, for any $x \in BC^{(k)}$, we have $\Pr\{f(x) < f(x^{(k)}\} > 1/2$ and for any $x \in \left(C^{(k)} - BC^{(k)}\right)$ we have $\Pr\{f(x) < f(x^{(k)}\} < 1/2$, and hence it can be obtained that $\Pr\{S_m\} > \Pr\{S_r\}$.

5.5.2 Expected Number of Evaluations of Kriging

Let $x_i^{(k)} = x^{(k)} + \Delta^{(k)} d_i^{(k)}, i = 1, \cdots, n_k$ and $Y^{(k)}$ denote the vector of response of currently used design set. By the assumptions of kriging model, we have $f(x_i^{(k)}) | Y^{(k)} \sim N\big(m_K(x_i^{(k)}), Var(x_i^{(k)})\big)$.

Since we select the points with the largest probability of outperforming the current center point as the most potential candidate, we are interested in checking this would help to save the total evaluations consumed. We compare a metamodel-guided visit with a random visit. In the former scenario, candidates are visited in a decreasing order of $\Pr\left\{f(x_i^{(k)}) < f(x^{(k)}) - \rho(\Delta^{(k)})\right\}$, i.e., $x_{mp}^{(k)}$ would be visited first, whereas in the later one, we visit candidate points randomly.

Let $EV_i^{(k)}$, $i = 1, \cdots, n_k$ denotes the expected number of evaluations required to terminate this iteration when there are i candidates to visit at iteration k. We add subscript R and K to denote the scheme of random visit and kriging-guided visit respectively.

Proposition 10. *For $\forall k = 1, 2, \cdots$, $EV_{R,i}^{(k)} \geq EV_{K,i}^{(k)}$ for $i = 1, 2, \cdots, n_k$.*

Proof. For $i = 1$, $EV_{R,1}^{(k)} = EV_{K,1}^{(k)} = 1$. This is due to the fact that we need to visit the only candidate anyway.

Assume that for $i = \ell$, $E_{R,\ell}^{(k)} \geq E_{K,\ell}^{(k)}$. Then at $(\ell+1)$th evaluation,

$$\begin{aligned} EV_{R,\ell+1}^{(k)} &= 1 + EV_{R,\ell}^{(k)} \cdot \left(\frac{\sum_{i=1}^{\ell+1} \Pr\left\{f(x_i^{(k)}) > f(x^{(k)}) - \rho(\Delta^{(k)})\right\}}{\ell+1} \right), \\ EV_{K,\ell+1}^{(k)} &= 1 + EV_{K,\ell}^{(k)} \cdot \Pr\left\{f(x_{mp}^{(k)}) > f(x^{(k)}) - \rho(\Delta^{(k)})\right\}. \end{aligned}$$

According to the definition of $x_{mp}^{(k)}$ given in equation (5.8), we have $EV_{R,\ell+1}^{(k)} \geq$

$EV_{K,\ell+1}^{(k)}$. $\square$

5.6 Numerical Experiments

We test our algorithm on a set of standard problems selected from some optimization library. The dimensions of these problems range from 2 to 10 and some of them have more than one stationary point. A summary of the selected problems can be referred in Table 5.1. The detailed information of each problem is presented in Appendix A.

Table 5.1: A summary of test problems.

Problem	d	No. of x^*	No. of local solutions (other than x^*)
Schwefel	2	1	several
Easom	2	1	several
Rosenbrock-10D	10	1	several
Branin	2	3	0
Goldstein-Price	2	1	several
Price Transistor	9	1	0
Singular	4	1	0
Shekel5	4	18	several

For each test problem, we implement a pattern search (GPS) algorithm without search step and use a classical update of the mesh size parameter of $1/2$. The positive spanning sets are set to have only one positive basis, which is $D^{(k)} \equiv D$. More specifically, we set $D := [-I, I]$, where I is the identity matrix of size d. This direction set with $2d$ columns corresponds to coordinate search and provides more accurate final iterates. In addition to GPS, we have also implemented a lower triangular MADS (Audet and Dennis Jr, 2006) for a reference. MADS and GPS use the same set of algorithm parameter and initialization information expect that MADS generates a random direction set at each iteration.

Three types of local metamodeling techniques are tested, which are linear regression (m_L), minimum Frobenius norm quadratic interpolation (m_F), and

Kriging (m_K). In the standard algorithm in which no metamodels are used (m_N), candidates are visited in a random order. This is different from the normal routine that the polling vectors are visited in a fixed predetermined order. This idea behind is that we want to remove the shape effect of the objective function.

The procedures for testing the finite-time performance of different algorithms are listed below.

1. Sample N starting points $x^{(0)}$ using LHS method within the feasible region X.
2. For each $x^{(0)}$, run the algorithm, and record a path of $f(x^{(k)})$ and the correspondent accumulated evaluations at each iteration k.
3. Take average on the N paths in terms of the $f(x^{(k)})$ and the accumulated evaluations at a sequence of simulation budget nodes, therefore we get one performance path from a certain algorithm for each test problem.
4. Eight algorithms from two schemes (GPS and MADS), which are denoted as GPS_N, GPS_L, GPS_F, GPS_K, $MADS_N$, $MADS_L$, $MADS_F$, $MADS_K$ are figured and compared together.

Both cases of $\rho \equiv 0$ and $\rho \neq 0$ are tested, and in the later case, we set $\rho(\Delta^{(k)}) = \left(\Delta^{(k)}\right)^{1.8}$. Numerical results are presented in Figure **??** and Figure 5.2. under the two schemes of $\rho(\cdot)$. In each plot, the performance of GPS is presented in solid line, while the counterpart of MADS is presented in dot dash line. The purpose is to compare the N model with other three (L, F, and K) in a matching group, i.e., whether the colored curves, both in solid and dash, are better than the black curve in solid and dash correspondingly.

Some trends can be observed from the results:

1. In most cases, m_N performs worse than the other three. This is implied by that the black curve tends to have a high y-axis value compared with the other curves, meaning that for a given budget, a random visit without a metamodel obtains a worse solution than our algorithms with local metamodels.
2. Compared with the case of $\rho \equiv 0$, the advantage of using metamodels becomes less obvious in the case of $\rho(\cdot)$. This makes sense in that metamodels are only expected to help in a successful iteration, while otherwise all the candidate points have to be evaluated anyway according to the algorithm. If $\rho(\cdot) \neq 0$, it becomes more difficult for an iteration to be successful, which makes the advantage of local metamodels less significant.

3. m_L and m_K exhibit a more promising performance on the test problems with multiple local optimal solutions, while m_F seems to be a better choice for singular-solution problems.
4. GPS and MADS do not show an obvious difference in terms of average performance.

5.7 Conclusions

Direct search methods can be made more efficient if past function evaluations are appropriately reused. We propose the use of local metamodeling techniques in direct search methods to improve the algorithm efficiency. The local metamodels are introduced and incorporated into the existing standard direct search methods to help save simulation cost without affecting the convergence results of the original algorithm. At each iteration of a pattern search method, one can attempt to compute a local metamodel by identifying a sampling set of previous visited points with good geometrical properties. To evaluate how much the local metamodel can help to save evaluations, we propose two measures to estimate its utility from two aspects, which are asymptotic performance and finite-time performance respectively. We draw the conclusion that under some assumptions, as iteration k goes infinity, local metamodels can detect an existing better candidate with one evaluation. We also justify the help of local metamodel in the sense of saving evaluations.

Three variants of local metamodels are discussed, which are simplex linear regression, minimum Frobenius norm quadratic interpolation and kriging. All the three variants satisfy some accuracy requirements of the estimation, which is a necessary condition for the expected asymptotic performance. For the construction of the metamodels, we assumed that a design set with a good geometric property was available so that no extra simulations were required for the metamodel construction. However, this good-geometric-property condition may not be satisfied in real-world implementation because none of our modifications demands new function evaluations. This issue is checked in the numerical experiments as the metamodels are constructed only based on the visited points which can be not well located in terms of their geometrical properties. The numerical results on a set of test problems show that the efficiency of the algorithm still get improved with the help of local metamodels even in this case.

Conclusively, function evaluations can be saved by incorporation local metamodels into direct search methods. This evaluation reduction is supportive by both mathematical analysis and numerical results, and this improvement is significant in practice especially in the cases where simulation cost is extremely high.

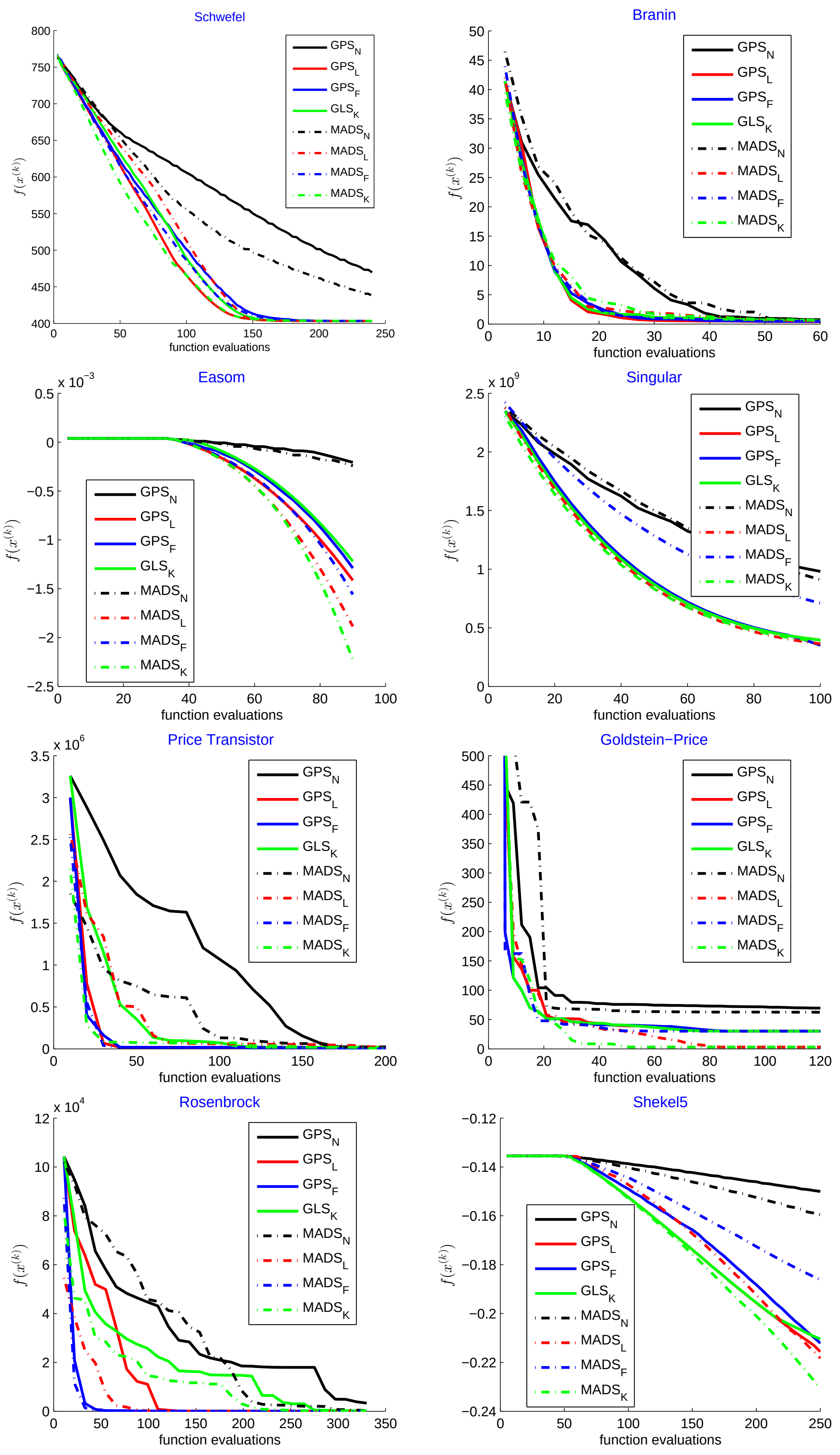


Figure 5.2: Results for case with $\rho \neq 0$

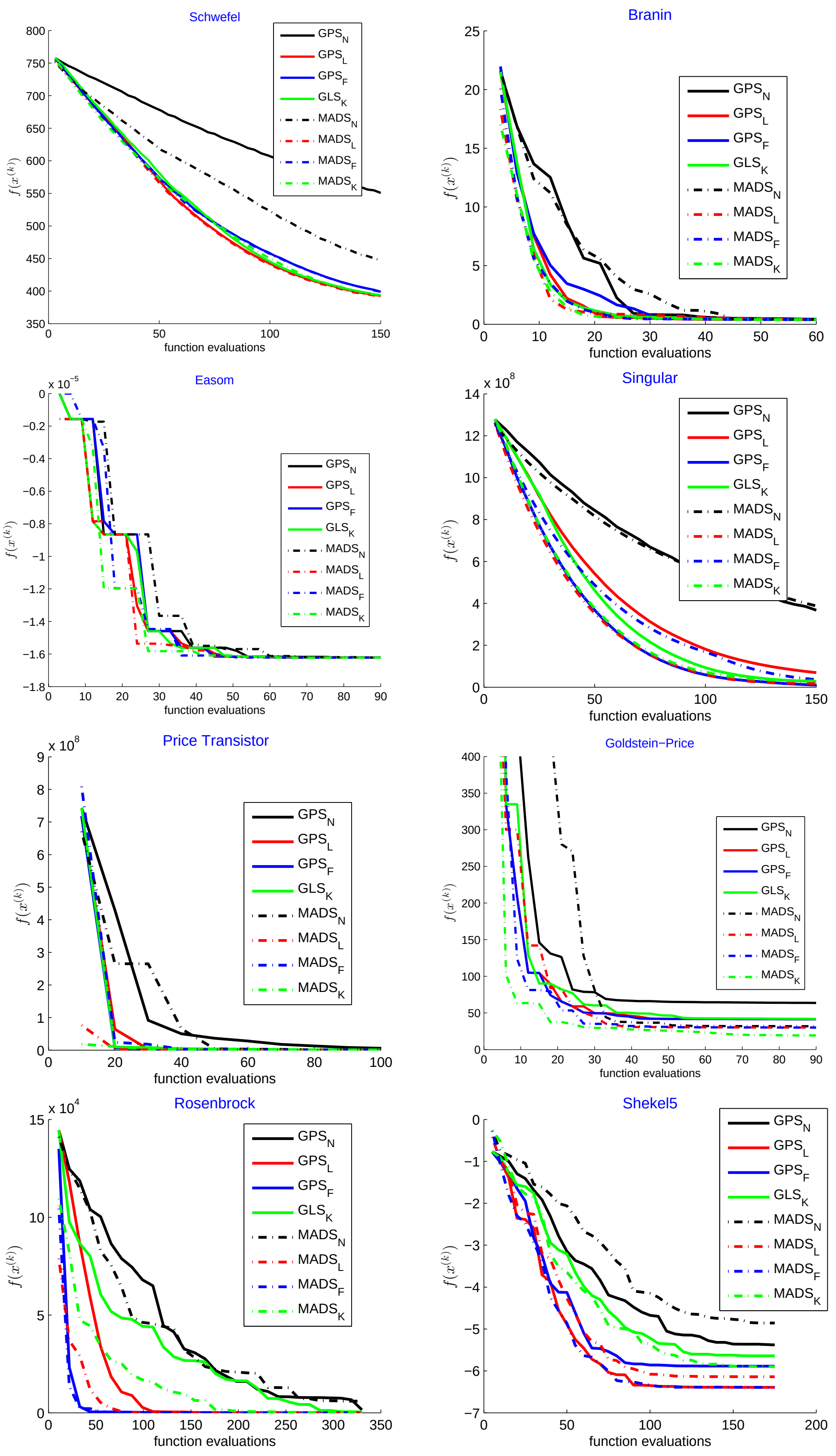


Figure 5.3: Results for case with $\rho \equiv 0$.

Chapter 6

Conclusions and Future Work

In this thesis, we explore enhanced simulation algorithms to reduce simulation evaluations in both stochastic and deterministic cases.

For the stochastic case, a specific type of optimization problems with a probability objective is studied in Chapter 3. Since probability functions are essentially expected values of indicator functions, optimization algorithms exhibiting a local convergence cannot work well on them when the SAA method is applied since the corresponding SAA function is ill-structured. We propose smoothed SAA methods to deal with this issue so that a wide range of (nonlinear) deterministic optimization algorithms could be successfully applied even when the sample size is small. The convergence of the solution set of the smoothed SAA methods is also discussed.

The smoothed SAA methods are further developed in Chapter 4 by introducing the idea behind retrospective-approximation (RA) methods. The enhanced method solves a sequence of smoothed SAA problems with increasing sample size, decreasing smoothing parameter, and decreasing error-tolerance driven by a computational strategy. The selection of the parameters are discussed. More specifically, we suggest to decrease the smoothing parameters at the same rate of error-tolerance based on the convergence properties of direct search methods and numerical test. The method numerically performs better than the fixed-sample-size SAA methods with smoothing techniques.

Both proposed enhanced SAA methods exhibit better performance than the classic SAA methods in dealing with probability objective problems. They use less simulation evaluations in average to achieve a given accuracy level and also perform more stable. Furthermore, our methods are general enough so that many variants of numerical algorithms can be employed. In conclusion, the two methods provide new options to the users who deal with expensive probability simulation optimization problems.

For the deterministic case, we propose a new framework in Chapter 5 to incorporate local metamodeling techniques into direct search methods to improve the efficiency of the algorithms without affecting their convergence results. The conditions on the local metamodels and the objective functions are discussed so that the utility of the local metamodels is guaranteed. Numerical results with a wide variety of test problems show that a large amount of evaluations can be saved by using direct search methods with the local metamodeling techniques even when the regularity conditions are not fully satisfied.

The following are some possible future research directions for this thesis.

1. The assumption imposed in Chapter 3 to obtain Lemma 1 is strong, which should be relaxed in our future work.
2. Mathematical justification for the selection of the smoothing parameters in Chapter 4 can be investigated.
3. The convergence rate or asymptotic distributions of $\|\nabla f(x_j)\|$ in Chapter 4 can also be explored to give a clearer guideline on the choice of three sequences of parameters.

Bibliography

Abramson, M.A. 2005. Convergence of mesh adaptive direct search to second-order stationary points. Tech. rep., DTIC Document.

Al-Sumait, J. S., A. K. Al-Othman, J. K. Sykulski. 2007. Application of pattern search method to power system valve-point economic load dispatch. *International Journal of Electrical Power & Energy Systems* **29** 720–730.

Alexander, S., T. F. Coleman, Y. Li. 2006. Minimizing cvar and var for a portfolio of derivatives. *Journal of Banking & Finance* **30** 583–605.

Anderson, E. J., M. C. Ferris. 2001. A direct search algorithm for optimization with noisy function evaluations. *SIAM Journal on optimization* **11** 837.

Andradóttir, S., A. A. Prudius. 2009. Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing* **21** 193–208.

Andradóttir, Sigrún. 1998. A review of simulation optimization techniques. *Simulation Conference Proceedings, 1998. Winter*, vol. 1. IEEE, 151–158.

April, J., F. Glover, J. P. Kelly, M. Laguna. 2003. Practical introduction to simulation optimization. *Simulation Conference, 2003. Proceedings of the 2003 Winter*, vol. 1. IEEE, 71–78.

Audet, C., V. Béchard, S. L. Digabel. 2008. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization* **41** 299–318.

Audet, C., J. E. Dennis Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization* **17** 188–217.

Audet, C., G. Savard, W. Zghal. 2010. A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research* **204** 545–556.

Audet, Charles, JE Dennis Jr. 2004. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization* **14** 980–1010.

Banks, J. 1998. *Handbook of simulation*. Wiley Online Library.

Banks, J. 2005. *Discrete Event System Simulation, 4/e*. Pearson Education India.

Bogani, C., M. G. Gasparo, A. Papini. 2009a. Generalized pattern search methods for a class of nonsmooth optimization problems with structure. *Journal of computational and applied mathematics* **229** 283–293.

Bogani, C., M. G. Gasparo, A. Papini. 2009b. Generating set search methods for piecewise smooth problems. *SIAM Journal on Optimization* **20** 321–335.

Box, G. E. P. 1957. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics* 81–101.

Box, George EP, KB Wilson. 1951. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)* **13** 1–45.

Brooke, A., D. Kendrick, A. Meeraus. 1996. *GAMS Release 2.25: A user's guide*. GAMS Development Corporation Washington, USA.

Carson, Y., A. Maria. 1997. Simulation optimization: methods and applications. *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society, 118–126.

Cea, J., J. Céa, J. Céa, J. Céa, F. Mathematician. 1971. *Optimisation: Théorie et algorithmes*. Dunod Paris.

Chen, H., B. W. Schmeiser. 2001. Stochastic root finding via retrospective approximation. *IIE Transactions* **33** 259–275.

Conn, A. R., N. I. M. Gould, P. L. Toint. 2000. *Trust-region methods*, vol. 1. Society for Industrial Mathematics.

Conn, A. R., K. Scheinberg, L. N. Vicente. 2009. *Introduction to derivative-free optimization*. SIAM.

Custódio, A. L., J. E. Dennis Jr, L. N. Vicente. 2008. Using simplex gradients of nonsmooth functions in direct search methods. *IMA journal of numerical analysis* **28** 770–784.

Custódio, A. L., H. Rocha, L. N. Vicente. 2010. Incorporating minimum frobenius norm models in direct search. *Computational Optimization and Applications* **46** 265–278.

Custódio, A.L., L.N. Vicente. 2007. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization* **18** 537–555.

Custódio, Ana Luísa, José F Aguilar Madeira, A Ismael F Vaz, Luís N Vicente. 2011. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization* **21** 1109–1140.

Davidon, W. C. 1991. Variable metric method for minimization. *SIAM Journal on Optimization* **1** 1–17.

Deng, G., M. C. Ferris. 2009. Variable-number sample-path optimization. *Mathematical Programming* **117** 81–109.

Dolan, E. D., R. M. Lewis, V. Torczon. 2003. On the local convergence of pattern search. *SIAM Journal on Optimization* **14** 567–583.

Fourer, R., D. M. Gay, B. W. Kernighan. 1993. *Ampl*. Boyd and Fraser.

Froidevaux, R. 1993. Constrained kriging as an estimator of local distribution functions. *Proceedings of the International Workshop on Statistics of Spatial Processes: Theory and Applications. Bari, Italy*. 106–118.

Fu, M. C. 1994. Optimization via simulation: A review. *Annals of Operations Research* **53** 199–247.

Fu, M. C. 2002. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing* **14** 192–215.

Fu, M. C., J. Q. Hu. 1995. Sensitivity analysis for monte carlo simulation of option pricing. *Probability in the Engineering and Informational Sciences* **9** 417–446.

Gaivoronski, A. A., G. Pflug. 2005. Value at risk in portfolio optimization: properties and computational approach. *Journal of Risk* **7** 1–31.

Geyer, C. J., E. A. Thompson. 1992. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)* 657–699.

Glasserman, P. 1990. *Gradient estimation via perturbation analysis*, vol. 116. Springer.

Gosavi, A. 2003. *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, vol. 25. Springer.

Güneş, F., F. Tokan. 2010. Pattern search optimization with applications on synthesis of linear antenna arrays. *Expert systems with applications* **37** 4698–4705.

Homem-De-Mello, T. 2003. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* **13** 108–133.

Homem-de Mello, T., Güzin Bayraksan. 2013. Monte carlo sampling-based methods for stochastic optimization .

Hooke, R., T. A. Jeeves. 1961. “direct search”solution of numerical and statistical problems. *Journal of the ACM (JACM)* **8** 212–229.

Hosseini, S. S. S., Ali Jafarnejad, Amir Hossein Behrooz, Amir Hossein Gandomi. 2011. Combined heat and power economic dispatch by mesh adaptive direct search algorithm. *Expert Systems with Applications* **38** 6556–6564.

Hough, P. D., T. G. Kolda, V. J. Torczon. 2002. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing* **23** 134–156.

Kibzun, A. I., Y. S. Kan. 1996. *Stochastic programming problems with probability and quantile functions*. Wiley.

Kim, S. 2006. Gradient-based simulation optimization. *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*. IEEE, 159–167.

Kim, S., R. Pasupathy, S. G. Henderson. 2011. A guide to sample-average approximation. *Handbook of Simulation Optimization* .

Kim, S., D. Zhang. 2010. Convergence properties of direct search methods for stochastic optimization. *Winter Simulation Conference (WSC), Proceedings of the 2010*. IEEE, 1003–1011.

Kleijnen, J. P. C., W. V. Beers, I. V. Nieuwenhuyse. 2010. Constrained optimization in expensive simulation: Novel approach. *European Journal of Operational Research* **202** 164–174.

Kolda, T. G., R. M. Lewis, V. Torczon. 2003. Optimization by direct search: New perspectives on some classical and modern methods. *Siam Review* 385–482.

Lacksonen, T. 2001. Empirical comparison of search algorithms for discrete event simulation. *Computers & Industrial Engineering* **40** 133–148.

Lagarias, J. C., J. A. Reeds, M. H. Wright, P. E. Wright. 1998. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization* **9** 112–147.

Lasdon, L., E. Popova. 2005. Monte carlo sampling-based methods in stochastic programming .

Lee, D., J. W. Kim, C. G. Lee, S. Y. Jung. 2011. Variable mesh adaptive direct search algorithm applied for optimal design of electric machines based on fea. *Magnetics, IEEE Transactions on* **47** 3232–3235.

Lepp, R. 1983. Stochastic approximation type algorithm for the maximization of the probability function'. *Eesti NSV Teaduste Akademia Toimetised, Füüsika, Matem* **32** 150–156.

Lewis, R. M., V. Torczon. 1999. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization* **9** 1082–1099.

Lewis, R. M., V. Torczon. 2000. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization* **10** 917–941.

Lewis, R. M., V. Torczon, M. W. Trosset. 2000. Direct search methods: then and now. *Journal of Computational and Applied Mathematics* **124** 191–207.

Linderoth, J., A. Shapiro, S. Wright. 2006. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research* **142** 215–241.

Liu, L., X. Zhang. 2006. Generalized pattern search methods for linearly equality constrained optimization problems. *Applied mathematics and computation* **181** 527–535.

McKinnon, K. I. M. 1998. Convergence of the nelder–mead ximplex method to a nonstationary point. *SIAM Journal on Optimization* **9** 148–158.

Meng, F., J. Sun, M. Goh. 2011. A smoothing sample average approximation method for stochastic optimization problems with cvar risk measure. *Computational Optimization and Applications* **50** 379–401.

Nelder, J. A., R. Mead. 1965. A simplex method for function minimization. *The computer journal* **7** 308–313.

Pasupathy, R. 2010. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Operations Research* **58** 889–901.

Pasupathy, R., B. W. Schmeiser. 2009. Retrospective-approximation algorithms for the multidimensional stochastic root-finding problem. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* **19** 5.

Polak, E. 1971. *Computational methods in optimization: A unified approach*, vol. 77. Academic Pr.

Powell, M. J. D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal* **7** 155–162.

Prékopa, A. 1988. Numerical solution of probabilistic constrained programming problems. *Numerical techniques for stochastic optimization* 123–139.

Prékopa, Andras. 2003. Probabilistic programming. A. Ruszczynski, A. Shapiro, eds., *Stochastic Programming*, *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier, 267 – 351. doi:http://dx.doi.org/10.1016/S0927-0507(03)10005-9. URL http://www.sciencedirect.com/science/article/pii/S0927050703100059.

Raik, E. 1975. The differentiability in the parameter of the probability function and optimization of the probability function via the stochastic pseudogradient method. *Eesti NSV Teaduste Akdeemia Toimetised. Füüsika-Matemaatika* **24** 860–869.

Rani, D., M. M. Moreira. 2010. Simulation–optimization modeling: A survey and potential application in reservoir systems operation. *Water resources management* **24** 1107–1138.

Regis, R. G., C. A. Shoemaker. 2010. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing* **21** 411.

Rios, L. M., N. V. Sahinidis. 2012. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization* 1–47.

Robbins, H., S. Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 400–407.

Robinson, S. M. 1996. Analysis of sample-path optimization. *Mathematics of Operations Research* **21** 513–528.

Rockafellar, R. T., J. O. Royset. 2010. On buffered failure probability in design and optimization of structures. *Reliability Engineering & System Safety* **95** 499–510.

Rockafellar, R. T., S. Uryasev. 2002. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance* **26** 1443–1471.

Rosenbrock, H. 1960. An automatic method for finding the greatest or least value of a function. *The Computer Journal* **3** 175–184.

Rubinstein, R. Y., D. P. Kroese. 2011. *Simulation and the Monte Carlo method*, vol. 707. Wiley.

Rubinstein, Reuven Y., A. Shapiro. 1993. *Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method*, vol. 346. Wiley New York.

Sacks, J., W. J. Welch, T. J. Mitchell, H. P. Wynn. 1989. Design and analysis of computer experiments. *Statistical science* **4** 409–423.

Semini, M., H. Fauske, J. O. Strandhagen. 2006. Applications of discrete-event simulation to support manufacturing logistics decision-making: a survey. *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 1946–1953.

Shapiro, A. 2001. Monte carlo simulation approach to stochastic programming. *Proceedings of the 33nd conference on Winter simulation*. IEEE Computer Society, 428–431.

Shapiro, A. 2003. Monte carlo sampling methods. *Handbooks in Operations Research and Management Science* **10** 353–425.

Shapiro, A., D. Dentcheva, A.j P Ruszczyński. 2009. *Lectures on stochastic programming: modeling and theory*, vol. 9. SIAM.

Shapiro, A., T. Homem-de Mello. 1998. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming* **81** 301–325.

Spendley, W., G. R._ Hext, F. R. Himsworth. 1962. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics* **4** 441–461.

Sriver, T. A., J. W. Chrissis, M. A. Abramson. 2009. Pattern search ranking and selection algorithms for mixed variable simulation-based optimization. *European journal of operational research* **198** 878–890.

Stein, A., L. C. A. Corsten. 1991. Universal kriging and cokriging as a regression procedure. *Biometrics* 575–587.

Swann, W. H. 1972. *Numerical methods for unconstrinaed optimization*, chap. Direct search methods. Acamedic Press, London, New York, 13–28.

Swisher, James R., P. D. Hyden, S. H. Jacobson, L. W. Schruben. 2000. A survey of simulation optimization techniques and procedures. *Simulation Conference, 2000. Proceedings. Winter*, vol. 1. IEEE, 119–128.

Torczon, V. 1991. On the convergence of the multidirectional search algorithm. *SIAM journal on Optimization* **1** 123–145.

Torczon, V. 1997. On the convergence of pattern search algorithms. *SIAM Journal on optimization* **7** 1–25.

Ubi, E. 1977. Statistical investigation of stochastic programming problems and method for their solution. *Proceedings of Tallinn Polytechnical Institute* **26** 57–76.

Uryas' ev, S. 1989. A differentiation formula for integrals overseas given by inclusion. *Numerical Functional Analysis and Optimization* **10** 827–841.

Vicente, Luís. N., A. L. Custódio. 2012. Analysis of direct searches for discontinuous functions. *Mathematical programming* **133** 299–325.

Vikram, N. S. 1995. A new smooth step function for adams. Tech. rep., ADAMS.

Xu, H., D. Zhang. 2009. Smooth sample average approximation of stationary points in nonsmooth stochastic optimization and applications. *Mathematical Programming* **119** 371–401.

Yamamoto, J. K. 2000. An alternative measure of the reliability of ordinary kriging estimates. *Mathematical Geology* **32** 489–509.

Zangwill, W. I. 1967. Minimizing a function without calculating derivatives. *The Computer Journal* **10** 293–296.

Appendix A

Summary of Test Problems

- Beale (BE)
  - Dimension: 2
  - Domain: $(-4.5, 4.5)^2$.
  - Definition: $BE(x) = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2 + 1$.
  - Local minima: none.
  - Global minima: $x^* = (3, 0.5)$, $BE(x^*) = 1$.
- Branin (BR)
  - Dimension: 2.
  - Domain: $x_l = (-5, 0)$, $x_u = (10, 15)$.
  - Local minima: none.
  - Global minima: $x^* = (-\pi, 12.275; \pi, 2.275; 9.42478, 2.475)$; $BR(x^*) = 0.397887$.
- Colville (CV)
  - Dimension: 4.
  - Domain: $[-10, 10]^4$.
  - Definition: $CO(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + \ldots 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$.
  - Local minima: none.
  - Global minima: $x^* = (1, 1, 1, 1)$, $CO(x^*) = 0$.
- Easom (EA)
  - Dimension: 2
  - Domain: $[-10, 10]^2$.

  - Definition: $EA(x) = -\cos(x_1)\cos(x_2)\exp(-x_1-\pi)^2-(x_2-\pi)^2$.
  - Local minima: none.
  - Global minima: $x^* = (\pi,\pi)$, $EA(x^*) = -1$.
- Price Transistor (PT)
  - Dimension: 9
  - Domain: $x_l = (0,0,0,0,0,0,0,0,0), x_u = (10,10,10,10,10,10,10,10,10)$. $x \in [x_l,\ x_u]$.
  - Definition: Define

$$g = \begin{pmatrix} 0.485 & 0.752 & 0.869 & 0.982 \\ 0.369 & 1.254 & 0.703 & 1.455 \\ 5.2095 & 10.0677 & 22.9274 & 20.2153 \\ 23.3037 & 101.779 & 111.461 & 191.267 \\ 28.5132 & 111.8467 & 134.3884 & 211.4823 \end{pmatrix}, \text{ and sumsqr} = 0.$$

For $k = 0:3$

$$\begin{aligned} \alpha &= (1.0-x_1x_2x_3)(\exp(x_5(g(1,k+1)-0.001g(3,k+1)x_7 \\ &\quad -0.001x(8)\cdot g(5,k+1))-1.0)) \\ &\quad -g(5,k+1)+g(4,k+1)x_2; \\ \beta &= (1.0-x_1x_2)x_4(\exp(x_6(g(1,k+1)-g(2,k+1) \\ &\quad -0.001g(3,k+1)x_7+0.001g(4,k+1)x_9))-1.0) \\ &\quad -g(5,k+1)x_1+g(4,k+1); \\ \text{sumsqr} &= \text{sumsqr}+\alpha^2+\beta^2; \end{aligned}$$

End for loop.

$PT(x) = (x_1x_3-x_2x_4)^2+\text{sumsqr}$.
  - Local minima: None;
  - Global minima: $x^* = (0.9, 0.45, 1, 2, 8, 8, 5, 1, 2)$, $PT(x^*) = 0$.
- Rosenbrock (RO)
  - Dimension: user defined d.
  - Domain: $[-5,5]^d$.

  - Definition: Initiate $RO(x) = 0$.

    For $i = 1 : d-1$

    $$RO(x) = RO(x) + (1-x_i)^2 + 100(x_{i+1} - x_i^2)^2;$$

    End for loop.

    $RO(x) = RO(x) + 1$.
  - Local minima: none.
  - Global minima: $x_i^* = 1, i = 1, \cdots, d$; $RO(x^*) = 0$.

- Shekel5 (SH)
  - Dimension: 4.
  - Domain: $[1, 10]^4$.
  - Definition: Define

$$A = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}, \; SH(x) = 0, \text{ and } c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}.$$

    For $i = 1 : 5$

    $$z = x - A(i, :);$$

    $$SH(x) = SH(x) - 1/(z \times z' + c(i));$$
  - Local minima: 760.
  - Global minima: 18. **End** for loop.

- Singular (SI)
  - Dimension: 4.
  - Domain:$[-100, 100]^4$.
  - Definition: $SI(x) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3))^4 + 10(x_1 - x_4)^4 + 1$;