

SIMULATION-BASED OPTIMIZATION

By

Geng Deng

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

(MATHEMATICS AND COMPUTATION IN ENGINEERING)

at the

UNIVERSITY OF WISCONSIN – MADISON

2007

© Copyright by Geng Deng 2007

All Rights Reserved

Abstract

Computer simulations are used extensively as models of real systems to evaluate output responses. The choice of optimal simulation parameters can lead to improved operation, but configuring them well remains a challenging problem. Simulation-based optimization is an emerging field which integrates optimization techniques into simulation analysis. The parameter calibration or optimization problem is formulated as a stochastic programming problem whose objective function is an associated measurement of an experimental simulation. Due to the complexity of the simulation, the objective function is typically (a) subject to various levels of noise, (b) not necessarily differentiable, and (c) computationally expensive to evaluate.

Contemporary simulation-based optimization methods include response surface methodology, heuristic methods and stochastic approximation. Our optimization methods proposed in the dissertation are adapted from the derivative-free optimization approach, which does not try to utilize or directly estimate the gradient value. Accordingly, we can avoid the sensitive gradient estimation process. Another important feature of our methods is to use replicated samples to reduce the noise level. The idea is similar to that of the sample-path optimization method, except that we have applied Bayesian inference tools in a novel fashion to compute variable numbers of

replications for different points.

The dissertation work is comprised of both optimization algorithm design and real-world applications. We have formulated the simulation-based optimization as a stochastic minimization problem, with the expectation form of the stochastic function. Simple constraints of parameters and separate situations concerning implementing Common Random Numbers (CRN) are considered. We concentrate on algorithms for problems where the number of simulation parameters is small.

We propose a two-phase optimization framework for simulation-based optimization. Phase I is a global exploration step over the entire domain. One of our methods employs classification tools to facilitate the global search process. By learning a surrogate from existing data the approach identifies promising regions for optimization. Another Phase I method is an extension of the DIRECT (DIviding RECTangles) optimization algorithm. Similar to the classification-based global search, the method returns a collection of promising regions which are represented by unions of rectangles. As a phase transition module linking the two phases, a nonparametric statistical method is applied to determine regions for multistart Phase II optimizations.

Phase II is a collection of local trust-region derivative-free optimizations based on the UOBYQA (Unconstrained Optimization BY Quadratic Approximation) algorithm. The methods apply Bayesian techniques to guide appropriate sampling strategies while simultaneously enhancing algorithmic

efficiency to obtain solutions of a desired accuracy. The statistically accurate scheme determines the number of simulation runs and guarantees the global convergence of the algorithm. In specific cases, we outline adaptations of the termination criteria for our algorithms.

A key guiding principle in this thesis is the use of the distribution information from the Bayesian analysis to instrument existing optimization codes to make them more robust to noisy function evaluations. In particular, the space partitioning scheme in DIRECT is significantly enhanced using a Monte Carlo validation based on Bayesian posterior distribution. More simulation runs are allocated when it is necessary to meet a desired accuracy level. In the extensions of UOBYQA algorithms, we use Bayesian tools to estimate the uncertainty in the quadratic model construction and even in solution distribution, thus we are able to make correct decisions within the steps of the algorithms.

A major deliverable of the research is a sophisticated implementation of this algorithm (WISOPT), combined with documentation to allow application experts to independently use this code. The methodologies have been applied in various applications. One project with Biomedical Engineering researchers explores the optimal design for a coaxial sleeve antenna for hepatic tumor ablation. Finite element (FE) simulation models are used to generate the electromagnetic (EM) radiation field in liver given a particular design. In addition to maximize the desired performance for the antenna, we take

into account the varying physical properties of tissue amongst individuals.

Another collaborative project is with researchers in the Medical School on the Wisconsin Breast Cancer Epidemiology simulation model. The model uses detailed individual-woman level discrete event simulation to replicate breast cancer incidence rates according to the SEER Program data. The two-phase optimization approach was applied to determine the parameter vector that minimizes a scoring function associated with the simulation model.

Besides applying the WISOPT package for black-box noisy output functions, we also consider solving a special type of simulation problem in Dynamic Programming (DP). Such a simulation problem typically includes state information (at each time stage) and state transition processes (along time stages). Normally, the objective function is associated with intermediate costs during state transitions plus a cost measurement of the final state. We consider a similar Monte Carlo simulation analysis using the Bayesian method within Neuro-Dynamic Programming (NDP), which is a class of approximation methods for solving complex DP problems. One of the NDP algorithms, the rollout method, approximates the optimal cost-to-go function by a heuristic cost function which is calculated via simulations. We efficiently manage the simulation resources in the rollout method to obtain accurate stochastic optimal controls. An example of scheduling radiation treatments is provided to illustrate the effectiveness of our new algorithm. We believe this demonstrates the applicability of the ideas in this thesis in

another application domain and points to possible generalizations for other optimization procedures.

Acknowledgements

I would like to express sincere thanks to my advisor Dr. Michael Ferris, who has led me to the splendid field of optimization and offered me tremendous help and advice through the dissertation process. The completion of this dissertation would not have been possible without his encouragement and guidance.

Many thanks to my other dissertation committee members, Dr. Stephen Wright, Dr. Robert Meyer, Dr. Thomas Kurtz, and Dr. Fabian Waleffe, for their thoughtful comments and suggestions, which continually improved the quality of this dissertation. I am indebted to Dr. Wright for teaching me courses on linear programming and nonlinear programming and Dr. Meyer for his courses on integer programming and network flow.

I am very grateful to researchers in the Biomedical Engineering Department: Punit Prakash, Dr. Mark Converse and Dr. John Webster. The collaboration with them in the microwave antenna project was joyful and fruitful.

My family members and friends have contributed to this research endeavor in substantive ways. Most importantly, I thank my wife Yongping for her constant encouragement and understanding, and her patiently enduring the project's evolution. I thank my parents, Songdian and Kejian, for setting me on the path toward a graduate education.

This research was partially supported by Air Force Office of Scientific Research Grant FA9550-07-1-0389, and National Science Foundation Grants DMS-0427689 and IIS-0511905.

Contents

Abstract	i
Acknowledgements	vi
1 Introduction	1
1.1 Simulation-Based Optimization	1
1.1.1 Simulation-Based Optimization Methods	4
1.1.2 White Noise vs. Common Random Numbers	10
1.2 Bayesian Analysis in Simulation	12
1.2.1 Selecting the Best System	14
1.2.2 Constructing Bayesian Posterior Estimations	16
1.2.3 The Bayesian Method for Selecting the Best System	20
1.3 The Two-Phase Optimization Framework	21
1.3.1 The WISOPT Structure	23
1.3.2 Introduction to the Methodologies	25
1.4 Outline of the Thesis	33
2 The Phase I Methods	36
2.1 Classification-Based Global Search	38
2.1.1 The Idea of Classification-Based Global Search	38

2.1.2	A Voting Scheme to Assemble Multiple Classifiers . . .	40
2.1.3	Handling Imbalanced Dataset	43
2.1.4	General Procedure	47
2.1.5	Numerical Examples	47
2.2	The Noisy DIRECT Algorithm	51
2.2.1	The DIRECT Optimization Algorithm	53
2.2.2	Modifications	58
2.2.3	Numerical Examples	65
2.3	The Phase Transition Module	73
3	Phase II Methods	80
3.1	The Deterministic UOBYQA Algorithm	81
3.1.1	The Core Algorithm	82
3.1.2	Interpolating Quadratic Model Properties	85
3.2	The VNSP-UOBYQA Algorithm	86
3.2.1	The Bayesian VNSP Scheme	89
3.2.2	Convergence Analysis of the Algorithm	103
3.2.3	Numerical Results	111
3.3	The Noisy UOBYQA Algorithm	118
3.3.1	Modifications	119
3.3.2	Numerical Results	128
4	Applications	136

4.1	The Wisconsin Breast Cancer Epidemiology Simulation	137
4.1.1	Introduction	137
4.1.2	Methods and Results	139
4.2	The Coaxial Antenna Design in Microwave Ablation	145
4.2.1	Introduction	145
4.2.2	Methods	148
4.2.3	Results	154
4.2.4	Conclusions	162
4.3	Ambulance Base Problem	164
5	Monte Carlo Simulation Efficiency in Neuro-Dynamic Programming	168
5.1	Introduction	169
5.2	Evaluating Rollout Policy Accuracy	178
5.2.1	Bayesian Posterior Estimation	180
5.2.2	Computing the <i>PCS</i>	181
5.3	Allocating Simulation Resource	184
5.3.1	The Resource Allocation Scheme	184
5.3.2	Special Case – Finite State Space	186
5.3.3	An Extension – Group Similar <i>M</i> by Policy	188
5.4	A Fractionated Radiotherapy Problem	190
5.5	Conclusions	197

6	Conclusions	198
A	User Guide to WISOPT	203
	Bibliography	214

List of Tables

1	Confusion matrix	42
2	The performance of the Noisy DIRECT algorithm for the Goldstein Price function, with $\sigma^2 = 10$	67
3	Comparison of the Noisy DIRECT algorithm and other algorithms for the Goldstein Price function. (Results are based on 10 replications of each algorithm).	70
4	The performance of the Noisy DIRECT algorithm for the 10-dimensional perm function, with $\sigma^2 = 1$	71
5	The performance of the new algorithm for the noisy Rosenbrock function, with $n = 2$ and $\sigma^2 = 0.01$	112
6	Averaged sample-path solution with different sample number N	114
7	Statistical summary	117
8	The performance of the Noisy UOBYQA algorithm for the Rosenbrock function, with $n = 2$ and $\sigma^2 = 0.01$	130
9	Suggested values to define N_{max}	131
10	Apply the Noisy UOBYQA algorithm to the Rosenbrock function (results are based on 10 replications of the algorithms).	133

11	Optimization results of the pricing model (over average value of 10 runs), where the real solution for $M = 2$ is 23.23 and for $M = 10$ is 68.28.	135
12	Design metrics in the sleeve antenna.	149
13	Objective metrics in the sleeve antenna design.	153
14	Comparison of dimensions of the original and optimized antennas.	160
15	Comparison of the objective metrics of the original and optimized antennas.	161
16	The output of Noisy DIRECT in the ambulance base problem	165
17	Variable sample sizes $M_{\hat{u}_l}$ for all the stages.	195
18	Sample-mean Q -factors for all the stages.	196
19	Options with default values	213

List of Figures

1	Structure of the floating sleeve antenna.	2
2	Mechanism of the sample-path optimization method.	12
3	The two-phase WISOPT structure.	26
4	Predicated local subregions of the Griewank function. The function has local optimums in each subregion (circles) and the global optimum at $[0, 0]$	28
5	The DIRECT optimization algorithm on the Goldstein Price function. The contour plot of the Goldstein Price function is shown in Figure 15.	29
6	In the model-based approach, a sequence of quadratic models are constructed to approximate the objective function.	32
7	In Phase I, classifiers determine new refined samples in a promising subregion.	41
8	Cleaning the dataset with Tomek links	45
9	The classification-based global search	48
10	Estimated level sets based on noisy data.	50
11	Apply the classification-based global search on the FR function.	52
12	Partitioning hyperrectangles	55

13	Identifying potentially optimal hyperrectangles	57
14	Generate multiple replications at each point and derive Bayesian posterior distributions	61
15	Contour plot of the Goldstein Price function	66
16	Replication number plot in Goldstein Price function optimiza- tion. The center region is magnified for a clear view.	68
17	Pdf function of a triangular distribution	74
18	Positions of the ambulance bases	74
19	Produce the initial point set \mathcal{I}	78
20	Mechanism of the new sample-path method with the VNSP scheme.	87
21	Choose the correct N_k and move the next iterate along the averaged sample function \hat{f}^{N_k}	89
22	Illustration of the subsequence $\{k_{j'}\}$	108
23	Compare changes of N_k with different levels of noise.	115
24	The trial solutions within a trust region are projected to each coordinate direction and the standard deviations of the pro- jected values are evaluated.	122
25	An envelop captures the variation of the incident curves in the four stages.	140
26	Structure of the floating sleeve antenna. Wall thicknesses of fixed dimensions are labeled in the figure.	147

27	Objective metrics for assessing size and shape of a $Q(\mathbf{r})$ profile	153
28	Histograms of the objective values	157
29	Variations of individual objective metrics (frequency plot). (a) Lesion radius, (b) Axial ratio - 0.5 , (c) S_{11} , (d) Probe radius	160
30	Comparing simulated $Q(\mathbf{r})$ profiles using the average dielectric properties (43.03,1.69). (a) antenna design presented by Yang et al (b) optimal design derived by our algorithm	162
31	A distribution of emergency calls	165
32	Two possible patterns of ambulance bases	167
33	Positions of the ambulance bases.	167
34	Four types of delivery error in hypo-fraction treatment	172
35	$M_{x_{k+1}}$ sample trajectories are simulated to evaluate the sample costs starting from the state x_{k+1}	182
36	The posterior distribution plot of Q -factors. The rollout policy corresponds to the best Q -factor.	184
37	Finite state space. M_{s_j} sample trajectories are simulated to evaluate the cost-to-go starting form state s_j . State s_j at stage $k + 1$, derived from any policy u_k , use the same replication number.	187

- 38 $M_{\hat{u}_l}$ sample trajectories are simulated to evaluate the sample costs starting from the state $x_{k+1}^{\hat{u}_l}$ which is derived from the policy \hat{u}_l . All the states x_{k+1} derived from the policy \hat{u}_l share the same replication number. 189
- 39 A simple one-dimension problem. x_k is the dose distribution over voxels in the target: voxels 3, 4, ..., 13. 190

Chapter 1

Introduction

1.1 Simulation-Based Optimization

Computer simulations are used extensively as models of real systems to evaluate output responses. Applications of simulation are widely found in many areas including supply chain management, finance, manufacturing, engineering design and medical treatment [42, 48, 60, 83, 96]. The choice of optimal simulation parameters can lead to improved operation, but configuring them well remains a challenging problem. Historically, the parameters are chosen by selecting the best from a set of candidate parameter settings. *Simulation-based optimization* [3, 40, 41, 47, 69, 90] is an emerging field which integrates optimization techniques into simulation analysis. The corresponding objective function is an associated measurement of an experimental simulation. Due to the complexity of the simulation, the objective function may be difficult and expensive to evaluate. Moreover, the inaccuracy of the objective function often complicates the optimization process. Deterministic optimization tools may lead to inaccurate solutions. Indeed, derivative information

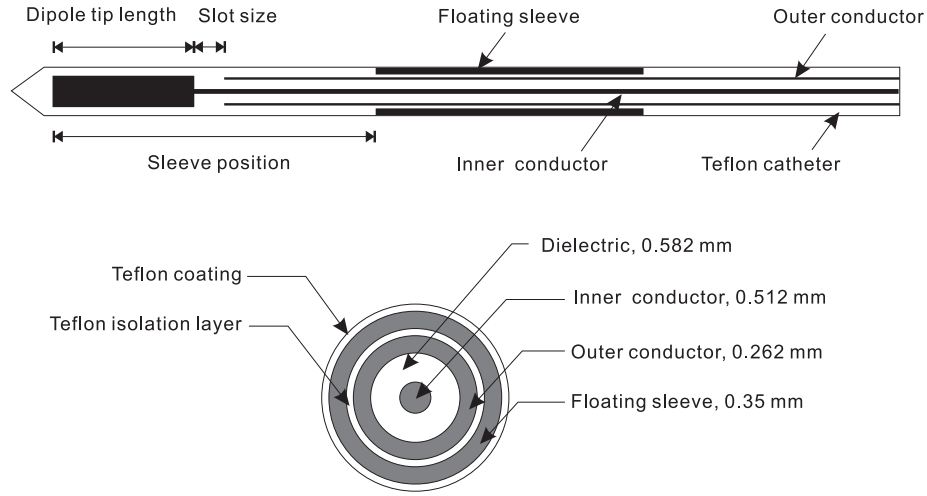


Figure 1: Structure of the floating sleeve antenna.

is typically unavailable, so many derivative-dependent methods are not applicable to these problems.

A particular example is an engineering design problem for an microwave coaxial antenna in hepatic tumor ablation. (More details can be found in Section 4.2.) We aim to determine the optimal dimensions of the antenna, such as the slot size, the dipole tip length, and the thicknesses of the Teflon coatings (see Figure 1 for the antenna illustration) to yield a desired treatment performance. Finite element models (MultiPhysics 3.2) are used to simulate the electromagnetic (EM) field distribution in liver given a particular design. Since dielectric properties of the tissue are assumed to vary within $\pm 10\%$ of average properties, it is important to ensure that the antenna used in ablation is robust, i.e., relatively insensitive to the variations in physical properties of the tissue.

Although real world problems have many forms, in the thesis we consider the following bounded stochastic formulation:

$$\min_{x \in \Omega} f(x) = \mathbb{E} [F(x, \xi(\omega))], \quad (1.1)$$

where

$$\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}.$$

Here, l and u are the lower and upper bounds for the input parameter x , respectively. We focus on the case where the parameter x comes from a continuous set Ω , but not necessarily a finite set. $\xi(\omega)$ is a random vector defined on a probability space. The sample response function $F : \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R}$ takes two inputs, the simulation parameters $x \in \mathbb{R}^n$ and a random sample of $\xi(\omega)$ in \mathbb{R}^d . Note that the variables x correspond to the design parameters (e.g., slot size, dipole tip length, etc.) in the above example, and the random realizations correspond to different dielectric properties of individuals. As in the example, our approaches concentrate on cases where the number of design parameters is small.

$F(x, \xi(\omega))$ corresponds to a particular evaluation using the finite element model. Given a random realization ξ_i of $\xi(\omega)$, $F(x, \xi_i)$ can be evaluated via a single simulation run. The underlying objective function $f(x)$ is computed by taking an expectation over the sample response function and has no explicit form. (Different formulations other than expectation are acceptable; for example, maximum, minimum, or quantiles of the sample objectives [4, 32]). A

basic assumption requires that the expectation function $f(x)$ is *well defined* (for any $x \in \mathbb{R}^n$ the function $F(x, \cdot)$ is measurable, and either $\mathbb{E}[F(x, \xi(\omega))_+]$ or $\mathbb{E}[F(x, \xi(\omega))_-]$ is finite, see page 57 of [91]).

1.1.1 Simulation-Based Optimization Methods

There are several excellent review papers on the subject of simulation-based optimization methods, such as [3, 40, 41]. These papers also provide a good survey on optimization add-ons for discrete-event simulation software, which have been undergoing fast development over the last decade; for example, OptQuest (www.optquest.com) for Arena (www.arena.com) and for SIMUL8 (www.simul8.com). A detailed summary of commercial simulation software and optimization add-ons can be found in Table 1 of [41]. Contemporary methods for continuous simulation-based optimization, which are implemented in these add-ons, are classified in several categories.

Response surface methodology The idea of *Response Surface Methodology* [12, 90] (RSM) is to construct one (or multiple) mathematical model A , which is called a surrogate model, to approximate the underlying function f , so that it is can be easily and cheaply evaluated at each parameter point x . Optimization procedures are executed over the inexpensive model A , instead of the expensive cost function. Moreover, the approximating model A could have estimated gradient values that enable the application of more efficient

optimization algorithms. We determine the model $A(\cdot)$ by minimizing the difference of $A(\cdot)$ and the function $F(\cdot)$ over a representative set of points \mathcal{S} :

$$\begin{aligned} \min \quad & \sum_{x_i \in \mathcal{S}} \Theta(F(x_i, \xi_i) - A(x_i)) \\ \text{s.t.} \quad & A(\cdot) \in \mathcal{A} \end{aligned} \tag{1.2}$$

Here $\Theta(\cdot)$ is a merit function, which is typically chosen as an l^2 -norm; ξ_i is one sample realization; and the set \mathcal{A} is a class of tunable functional forms. The most popular choices of \mathcal{A} are linear and quadratic models. Polynomial models are also compact forms and easy to evaluate, but they often have limited capacity to model complex functions of arbitrary shape. Splines are alternative tools, which are more flexible in fitting and are capable of handling large scale data. But in high dimensional approximation, the (Tensor product) spline requires sample data on a grid mesh for efficient implementation. Kriging is a type of interpolative tool which was originally developed in geo-statistics. It is flexible enough to be implemented on high-dimensional spatial data and has been used in many engineering problems. However, the performance is limited by the total number of design points; for example, increasing the number of points in \mathcal{S} can quadratically increase the Kriging model construction time.

Heuristic methods Heuristic methods have proven to be practically useful in many real-world applications. We will briefly introduce the three most popular methods: genetic algorithms, tabu search and simulated annealing.

Genetic algorithms [13, 94, 95] are inspired from the process of biological evolution. The algorithm is initialized with a finite set of potential solutions called the population. Each potential solution, referred to as an individual, may be coded as a binary string or a real-coded integer or taken from a fixed alphabet of characters. These solutions are evaluated by a fitness function (normally the objective function) and the fit individuals are assigned a high probability to “reproduce” in the next generation of solutions, a sort of survival of the fittest scheme. In the reproducing process, new offspring inherit traits from parents (crossover) and are subject to mutation changes. The new offspring are accepted using various criteria to form a new population of candidate solutions. The algorithm proceeds to generate more favorable solutions in each iteration and eventually converges to a population with a distribution of good solutions.

Tabu search [45, 46] is a metaheuristic based on the *local search method*, which iteratively moves the current iterate to a neighbor solution, until certain criteria are satisfied. The algorithm allows a move to a neighbor solution that has a worse objective value. Meanwhile, in order to prevent circling to the previous solutions or infinite loops, a list of tabu or forbidden moves are kept and updated at each iteration. A short-term memory tabu list is the most used type and is often referred to as a *tabu tenure*.

Simulated annealing [23, 31] searches local moves randomly from a list of candidate neighbor points. If a better neighbor point is encountered, it

replaces the current iterate with probability one, or if a worse point is found, it replaces the iterate with a probability value strictly less than one. The appropriate probability value is determined by the difference of the objective values. For the algorithm to converge, the probability of moving towards a worse point should decrease along the iterations according to the decrement of a certain ‘temperature’ value, which changes based on a *cooling schedule*. All of the three algorithms are considered as global optimization methods since they are able to move iterates out of regions where locally optimal solutions reside.

Stochastic approximation Stochastic approximation [67, 104] falls into the category of gradient-based approaches. Typically, the method iteratively updates the current solution by

$$x_{k+1} = x_k + a_k \nabla \tilde{f}(x_k),$$

where the estimated gradient $\nabla \tilde{f}(x_k)$ can be calculated by various gradient estimation tools and a_k is the step length. Since the method is an extension of the line search method, it obviously is a local optimization method. Under proper conditions, such as when the error in the gradient approximation and the step length converges to zero as a certain rate, the stochastic approximation method can be shown to converge to a local optimum of the underlying function.

The approximate gradient $\nabla \tilde{f}(x_k)$ is estimated using sample responses F .

Popular gradient estimation methods include perturbation analysis and the likelihood ratio method [43]. Spall's lab develops simultaneous perturbation (SP) and finite difference (FD) based stochastic approximation algorithms: SPSA and FDSA. The lab's web site (www.jhuapl.edu/SPSA) provides a good source of references on these two algorithms.

Derivative-free optimization methods Derivative-free optimization methods [80] are a class of methods that do not try to utilize or directly estimate the gradient value, thus are a good fit for the optimization problem (1.1). Compared to stochastic approximation algorithms, the derivative-free methods avoid the gradient estimation step, which is sensitive and crucial to the convergence of these algorithms. In many practical examples, we find that the gradient estimation tools often become incorrect and problematic when the gradient value gets close to zero (i.e., when near a local solution).

There are two categories of derivative-free methods: the so-called model-based approach [20, 21] and the pattern or geometry-based approach. The model-based approach typically constructs a chain of local models that approximate the objective function, and the algorithm proceeds based on model predictions; an alternative to the model-based approach is the pattern-based approach, which directly uses the functional output at locations specified by geometric arguments, such as the pattern search method [70].

One of the optimization methods we apply in the thesis is the UOBYQA

algorithm [85] (Unconstrained Optimization BY Quadratic Approximation), which is a model-based method. It is designed for solving nonlinear problems with a moderate number of dimensions. The method shares similarities to response surface methodology (RSM): both of them construct a series of models to the simulation response during the optimization process. However, many features in UOBYQA, such as the quadratic model update and local region shift have advantages over the classical RSM, and UOBYQA is more mathematically rigorous in convergence.

Dynamic programming and neuro-dynamic programming Dynamic Programming (DP) [6] problems are a special type of simulation-based optimization problems with internal time stages and state transitions. The objective function is not a single black-box output, but typically is a combination of intermediate costs during state transitions plus a cost measurement of the final state. Appropriate controls are determined at each time stage, typically in a sequential favor.

Neuro-Dynamic Programming (NDP) [9, 47, 106] is a class of reinforcement learning methods to solve complex DP problems. The central idea of NDP is to approximate the optimal cost-to-go function using simple structured functions, such as neuro networks or simulation evaluations. NDP obtains sub-optimal controls, trading off expensive computational costs. This feature distinguishes NDP methods from standard DP methods.

1.1.2 White Noise vs. Common Random Numbers

To handle stochastic functional output, we adopt a simple and effective approach - sample multiple replications per point and take the average to reduce the uncertainty. Algorithms without such a procedure can obtain a good solution but not the exact solution, because a single-replication observation is inevitably affected by noise. The tricky question of setting the replication number is actually algorithm dependent. We have applied tools from Bayesian inference in a novel way to facilitate this computation process.

We separate our optimization approaches into two cases, using an important property of the simulation system, which is based on properties of the replications.

The **white noise** case corresponds to the situation when simulation outputs $F(x, \xi(\omega))$ are independent for different simulation runs, whatever the input x is. The random component $\xi(\omega)$ of the function is not fixable. The simulation outputs are observed to be biased by white noise. The second case is the implementation of **Common Random Numbers** (CRN), which assumes the random factor $\xi(\omega)$ can be fixed; for example, this can be achieved by fixing the random seed in the computer simulation process. One of the advantageous properties of using CRN is that given a realized sample ξ_i , the sample response function $F(x, \xi_i)$ is a deterministic function. The introduction of CRN significantly reduces the level of randomness and can facilitate comparisons between different x .

When CRN is used, the expected value function $f(x)$ in (1.1) can be approximated by a deterministic averaged sample response function

$$f(x) \approx \hat{f}^N(x) := \frac{1}{N} \sum_{i=1}^N F(x, \xi_i), \quad (1.3)$$

where N is the number of samples. This is the core idea of the *sample-path method* [35, 49, 50, 83, 84, 89], which is a well-recognized method in simulation-based optimization. The sample-path method is sometimes called the *Monte Carlo sampling approach* [99] or the *sample average approximation method* [52, 53, 63, 98, 100, 101]. The sample-path method has been applied in many settings, including buffer allocation, tandem queue servers, network design, etc. A wealth of deterministic optimization techniques can be employed to solve the sample-path problem

$$\min_{x \in \Omega} \hat{f}^N(x), \quad (1.4)$$

which serves as a substitute for (1.1). An optimal solution $x^{*,N}$ to the problem (1.4) is then treated as an approximation of x^* , the solution of (1.1). Note that the method is not restricted to bounded problems, but in more general settings it requires appropriate deterministic tools (i.e., constrained optimization methods) to be used.

Convergence proofs of the sample-path method are given in [89, 97]. Suppose there is a unique solution x^* to the problem (1.1), then under the assumption that the sequence of functions $\{\hat{f}^N\}$ epiconverges¹ to the function

¹The functions \hat{f}^N epiconverges to a function f if the epigraphs of the function \hat{f}^N converges to the epigraph of the function f [91].

f , the optimal solution sequence $\{x^{*,N}\}$ converges to x^* almost surely for all sample paths. See Figure 2 for the illustration of the sample-path optimization method. Starting from x_0 , for a given N , a deterministic algorithm is applied to solve the sample-path problem. Invoking the above analysis then allows us to use the solution $x^{*,N}$ as an approximation to the true solution x^* .

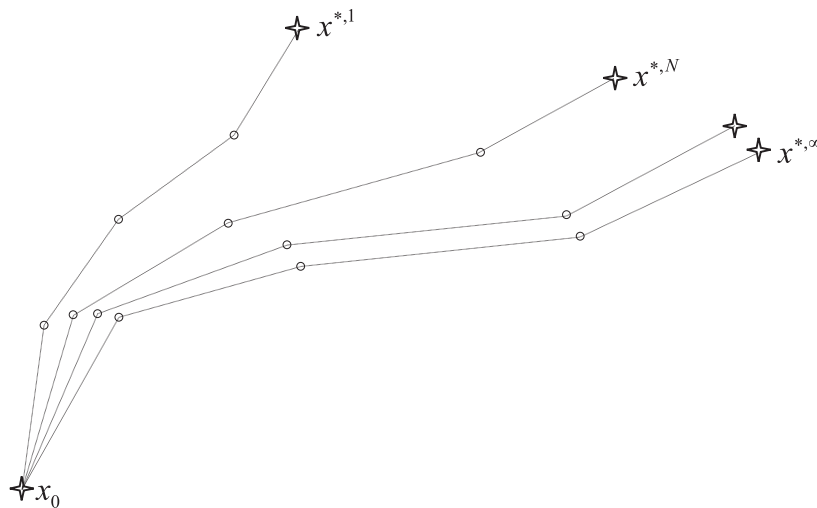


Figure 2: Mechanism of the sample-path optimization method.

1.2 Bayesian Analysis in Simulation

We have extensively used Bayesian analysis in our optimization methods. Our analysis is inspired by its application in *selecting the best system*, which is one of the fundamental problems in discrete simulation-based optimization.

In Bayesian analysis, a posterior distribution for the simulation output, including both mean and variance information, is derived. Since the posterior distribution has a parametric form, it can be readily assembled and used in optimization methods.

Another important usage of Bayesian analysis is *Monte Carlo validation*. When Bayesian posterior distributions cannot be used directly in the algorithms, the Monte Carlo validation step serves as an alternative. This idea is motivated by using Monte Carlo simulation of random variables to construct statistical estimators for unknown quantities. For example, in the computation of the expectation of a random variable, a simple way to accomplish this is to generate a large number of realizations from the random variable and take the arithmetic mean (or compute the sample mean). This provides a so called *Monte Carlo estimator* for the unknown expectation. Note that one crucial part of the Monte Carlo method necessitates an efficient way to generate random samples from a given distribution.

We adopt the same idea in Monte Carlo validation. Assume that Bayesian posterior distributions can capture the real uncertainty of the unknown random quantity well from a practical standpoint. In order to valid a given variance criterion, we can extract a vast amount of samples from the derived posterior distributions and plug them into the criterion to validate it. For example, we may observe that $1 - \alpha$ of all the samples satisfy the criterion,

therefore, we can assert that such a criterion can be satisfied with an accuracy of $1 - \alpha$. The Monte Carlo validation step avoids using the forms of the Bayesian distributions directly.

We will introduce the problem of selecting the best system in Section 1.2.1 and the role which Bayesian analysis plays. In Section 1.2.2, we describe the construction of Bayesian posterior estimations and in Section 1.2.3, we illustrate how the analysis is applied to a simple selecting the best system problem.

1.2.1 Selecting the Best System

The goal of selecting the best system is to identify the best system from a collection of candidate systems. The type of problem is encountered many times (in various forms) in our algorithmic designs, as will be shown later. Selecting the best system is equivalent to solving a discrete optimization problem:

$$\arg \min_i E(Y_i), \quad (1.5)$$

where Y_i is a random variable representing the output of system i , $i = 1, 2, \dots, K$. (Without loss of generality, we assume that a desired system has the smallest expected mean.) Let μ_i be the unknown mean of Y_i and $\mu_{[1]} \geq \mu_{[2]} \geq \dots \geq \mu_{[K]}$ be the ordered sequence of the means μ_i , but such a sequence is unknown. We prefer to determine the index $[k]$ and there is a

probability of correct selection (*PCS*) value that quantifies the correctness:

$$PCS = Pr(\text{select } Y_{[K]} | \mu_{[j]} \geq \mu_{[K]}, j = 1, 2, \dots, K - 1). \quad (1.6)$$

The difficulty of selecting the best system is to design a statistically accurate procedure that identifies the best system with the condition

$$PCS \geq 1 - \alpha,$$

where α is a threshold value, while using the least number of realized samples of Y_i .

The standard approach to selecting the best system includes *Indifference-Zone Ranking and Selection* (IZ-R&S). An indifference-zone parameter δ should first be prescribed. When the means of two systems are within a tolerance δ , it is indifferent to choose either system. One of the most influential IZ-R&S approaches is a two-stage sampling strategy, described by Rinott [88] in 1978. Firstly, r_0 replications of each system are simulated to estimate the preliminary sample variance. Secondly, it evaluates additional replications for each system, whereby numbers of replications are determined by the corresponding variance information. The best system is then chosen as the one having the smallest sample mean. Kim and Nelson [61] include a screen step that is designed to eliminate a set of inferior systems from consideration at an early stage. A good survey on the IZ-R&S procedures can be found in [62].

The Bayesian method is an alternative to the standard IZ-R&S [16, 17, 19]. The idea is to construct posterior distributions for the underlying mean μ_i using observations and acquired knowledge. The selection accuracy is accordingly estimated via joint posterior distributions. There are many practically efficient procedures including Chen’s *OCBA* [16] (Optimal Computer Budget Allocation) and Chick’s 0-1(\mathcal{B}) [17, 19]. The report [56] offers an empirical comparison of these methods, as well as Rinott’s two-stage IZ-R&S method.

The Bayesian approach has advantages over the traditional IZ-R&S approach. First, it is not necessary to set the indifference-zone parameter δ . In fact, it deals with the ranking and selection problem (1.5) directly. Moreover, it utilizes both the mean and variance information of the data, while IZ-R&S only uses the variance information directly.

1.2.2 Constructing Bayesian Posterior Estimations

The Bayesian approach described in this subsection is standard and close to that presented in Chick’s papers [17, 19].

Given a set of points $\{x^1, x^2, \dots, x^L\}$, we assume the simulation output at these points

$$\mathbf{F} = (F(x^1, \xi(\omega)), F(x^2, \xi(\omega)), \dots, F(x^L, \xi(\omega)))$$

is a multivariate normal variable, with mean $\boldsymbol{\mu} = (\mu(x^1), \dots, \mu(x^L))$ and

covariance matrix Σ :

$$\mathbf{F} \sim N(\boldsymbol{\mu}, \Sigma). \quad (1.7)$$

In the white noise case, since simulation outputs are independent, Σ should be a diagonal matrix; while for the CRN case, it is typically not. Suppose we evaluate N replications of simulation runs at each point, the existing data X can be accumulated as an $N \times L$ matrix, with

$$X_{i,j} = F(x^j, \xi_i), i = 1, \dots, N, j = 1, \dots, L.$$

Let $\bar{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ denote the sample mean and sample covariance matrix of the data. For simplicity, we introduce the notation $\mathbf{s}_i = (F(x^1, \xi_i), \dots, F(x^L, \xi_i))$, $i = 1, \dots, N$, so that

$$X = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_N \end{bmatrix}.$$

The sample mean and sample covariance matrix are calculated as

$$\begin{aligned} \bar{\boldsymbol{\mu}} &= \frac{1}{N} \sum_{i=1}^N \mathbf{s}_i \\ &= (\hat{f}^N(x^1), \dots, \hat{f}^N(x^L)), \end{aligned} \quad (1.8)$$

and

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{s}_i - \bar{\boldsymbol{\mu}})^T (\mathbf{s}_i - \bar{\boldsymbol{\mu}}). \quad (1.9)$$

Since we do not have any prior assumption for the distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we assign non-informative prior distributions for them. In doing this, the joint posterior distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are derived as

$$\begin{aligned}\boldsymbol{\Sigma}|X &\sim \text{Wishart}_L(\hat{\boldsymbol{\Sigma}}, N + L - 2), \\ \boldsymbol{\mu}|\boldsymbol{\Sigma}, X &\sim N(\bar{\boldsymbol{\mu}}, \boldsymbol{\Sigma}/N).\end{aligned}\tag{1.10}$$

Here the Wishart distribution $\text{Wishart}_p(\boldsymbol{\nu}, m)$ has covariance matrix $\boldsymbol{\nu}$ and m degrees of freedom [25]. The Wishart distribution is a multivariate generalization of the χ^2 distribution.

The distribution of the mean value $\boldsymbol{\mu}$ is of most interest to us. When the sample size is large, we can replace the covariance matrix $\boldsymbol{\Sigma}$ in (1.10) with the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$, and asymptotically derive the posterior distribution of $\boldsymbol{\mu}|X$ as

$$\boldsymbol{\mu}|X \sim N(\bar{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}/N).\tag{1.11}$$

It should be noted that, with an exact computation, the marginal distribution of $\boldsymbol{\mu}|X$ inferred by (1.10) (eliminating $\boldsymbol{\Sigma}$) is,

$$\boldsymbol{\mu}|X \sim \text{St}_L(\bar{\boldsymbol{\mu}}, N\hat{\boldsymbol{\Sigma}}^{-1}, N - 1),\tag{1.12}$$

where a random variable with Student's t-distribution $\text{St}_L(\boldsymbol{\mu}, \boldsymbol{\kappa}, m)$ has mean $\boldsymbol{\mu}$, precision $\boldsymbol{\kappa}$, and m degrees of freedom. The normal formulation (1.11) is more convenient to manipulate than the t-version (1.12). We have applied both forms in our algorithms when appropriate, but typically prefer the simpler normal form.

Particularly, in the white noise case, the distribution in (1.12) is separable.

We have the component posterior distribution

$$\mu(x^i)|X \sim N(\bar{\mu}(x^i), \hat{\sigma}^2(x^i)/N),$$

where $\hat{\sigma}^2(x^i)$ is the sample mean for the point x^i and is the i th component in the diagonal of the matrix $\hat{\Sigma}$.

While the multivariate normal assumption (1.7) is not always valid, several relevant points indicate that it is likely to be satisfied in practice [18].

- The form (1.7) is only used to derive the (normal) posterior distribution $\boldsymbol{\mu}|X$.
- Other types of distribution assumptions may be appropriate in different circumstances. For example, when a simulation output follows a Bernoulli 0-1 distribution, then it would be easier to perform parameter analysis using beta prior and posterior distributions. The normal assumption (1.7) is the more relevant to continuous simulation output with unknown mean and variance.
- The normal assumption is asymptotically valid for many applications. Many regular distributions, such as distributions from the exponential family, are normal-like distributions. The analysis using normal distributions is asymptotically correct.

1.2.3 The Bayesian Method for Selecting the Best System

As an introductory application, we consider a preliminary case in selecting the best system when there are only two systems involved, $K = 2$. A simplified analysis on how to compute the *PCS* is presented as follows. Interested readers can seek details (e.g., multiple comparisons and procedures) in [17, 19].

Let $X = \{y_{i,j}, i = 1, 2, j = 1, 2, \dots, r_i\}$ denote two sequences of output of both systems. The sample mean $\bar{\mu}_i$ and sample variance $\hat{\sigma}_i^2$ are defined as $\bar{\mu}_i = \sum_{j=1}^{r_i} y_{i,j}/r_i$ and

$$\hat{\sigma}_i^2 = \sum_{j=1}^{r_i} (y_{i,j} - \bar{\mu}_i)^2 / (r_i - 1), \text{ for } i = 1, 2.$$

A decision is drawn by directly comparing the two sample means. The system evidenced with a smaller average output is selected:

$$\text{Choose system} \begin{cases} 1, & \text{if } \bar{\mu}_1 \leq \bar{\mu}_2; \\ 2, & \text{otherwise.} \end{cases}$$

Without loss of generality, we assume that we observe the order of the sample means $\bar{\mu}_1 \leq \bar{\mu}_2$ and select the first system as the winner.

In the Bayesian framework, the posterior distribution of $\mu_i|X$ can be asymptotically estimated as

$$\mu_i|X \sim N(\bar{\mu}_i, \hat{\sigma}_i^2/r_i). \tag{1.13}$$

Following our earlier assumption, the *PCS* corresponds to the probability of event $\{\mu_1 \leq \mu_2\}$,

$$PCS \sim Pr(\mu_1 \leq \mu_2 | X) = Pr(\mu_1 | X - \mu_2 | X \leq 0). \quad (1.14)$$

The difference of two normal random variables $\mu_1 | X$ and $\mu_2 | X$ remains a normal random variable. It is straightforward to show:

$$\mu_1 | X - \mu_2 | X \sim N(\bar{\mu}_1 - \bar{\mu}_2, \hat{\sigma}_1^2/r_1 + \hat{\sigma}_2^2/r_2).$$

Therefore, the *PCS* defined in (1.14) is a tail probability of a normal distribution:

$$PCS \sim Pr(N(\bar{\mu}_1 - \bar{\mu}_2, \hat{\sigma}_1^2/r_1 + \hat{\sigma}_2^2/r_2) \leq 0). \quad (1.15)$$

The computation corresponds to a single cdf evaluation of a one-dimensional normal distribution. The value can be computed either by looking up in a standard cdf table or using routines in mathematics software, i.e., Matlab's function `normcdf`.

In general, for formulation (1.15), one may expect the *PCS* value to become higher as the numbers of replications r_1 and r_2 increase.

1.3 The Two-Phase Optimization Framework

The thesis focuses on a two-phase optimization framework for solving (1.1), which is motivated by response surface methodology. Recalling the model construction step (1.2), the overall error in approximation comes from two

sources: random error and model error. Random error is the type of error directly induced by the uncertainty $\xi(\omega)$ in the sample response; model error is due to an inappropriate choice of functional forms to fit the data. For example, fitting a cubic curve with a quadratic function will result in a large discrepancy, when the domain of interest is not appropriately restricted. This type of error exists independent of the random term $\xi(\omega)$. In the global view, the model error dominates the random error; therefore, constructing a surrogate function aims at reducing the model error to the maximum extent. However, in the local view, reducing the random error becomes the primary consideration. For this reason, we design a two-phase framework for simulation-based optimization which addresses distinct goals:

1. Phase I is a global exploration and rough search step. The algorithm explores the entire domain and proceeds to determine potentially good subregions for future investigation. We assume the observed randomness in the subregion detection process is insignificant and can be ignored. Appropriate criteria to determine when a good subregion has been identified are required.
2. Phase II is a local exploitation step. Local optimization algorithms are applied in each subregion to determine the exact optimum. The algorithms are required to deal with the randomness explicitly, since in local subregions, random error is considered as the dominating feature.

Phase I typically produces one or multiple good points representing centers of good local subregions. These points are used as starting points for the Phase II local search algorithms. Alternatively, Phase I may generate geometric information for the location and shape of the subregions, which are more interesting for analyzing the significance, interactions and patterns of individual dimensions.

1.3.1 The WISOPT Structure

We design an optimization package WISOPT (which stands for WIsconsin Simulation OPTimization) incorporating different optimization methodologies, based on the two-phase framework. See Figure 3 for a flow chart.

Phase I is a global exploration step over the entire domain. Algorithms in Phase I should be able to generate (and evaluate) densely distributed samples in promising subregions and sparsely distributed samples in inferior subregions. The entire set of samples will be passed to a phase transition procedure between Phase I and Phase II, which implements a non-parametric statistical method to determine starting points and surrounding subregions for multistart Phase II optimizations.

One of our Phase I methods employs classification tools to facilitate the global search process. By learning a surrogate from existing data the approach identifies promising subregions and generates dense samples in the

subregions. Additional features of the method are: (a) more reliable predictions obtained using a voting scheme combining the options of multiple classifiers, (b) a data pre-processing step that copes with imbalanced training data. Another Phase I method is the Noisy DIRECT (DIviding RECTangles) algorithm, which is an extension of the DIRECT optimization algorithm to the noisy case. As with the classification-based global search, the method returns a collection of promising points together with surrounding rectangles.

Phase II performs local derivative-free optimization based on the UOBYQA (Unconstrained Optimization BY Quadratic Approximation) algorithm, in each of the subregions identified. We do consider whether we can implement CRN in the simulator, corresponding to the VN-SP-UOBYQA (Variable-Number Sample-Path UOBYQA) algorithm for the CRN case and the Noisy UOBYQA algorithm for the white noise case. Both algorithms apply Bayesian techniques to guide appropriate sampling strategies while simultaneously enhancing algorithmic efficiency to obtain solutions of a desired accuracy. The statistically accurate scheme determines the number of simulation runs and guarantees the global convergence of the algorithm.

A key component in the extended algorithms is to incorporate distribution information provided by Bayesian estimations. Bayesian inference is an effective tool in input modeling and uncertainty analysis. In particular, in order to prevent unnecessarily exploring hyperrectangles in inferior regions,

DIRECT tests the accuracy of selecting hyperrectangles for future partitioning using a Monte Carlo validation process. Trial samples are extracted from the Bayesian posterior distributions to validate a predefined variance criterion. In UOBYQA algorithms, we derive the posterior volatility for the local model construction step, and therefore control the uncertainty of solutions in the trust region subproblems.

Certainly, the variety of methods in both phases is not restricted to what we present. Additional methods that satisfy the purposes of both phases may work as new modules and can be plugged into the two-phase framework.

1.3.2 Introduction to the Methodologies

We give a brief introduction to the methodologies we use in WISOPT (refer to Figure 3). Details will be discussed in later chapters.

Classification-based global optimization A good surrogate model for the entire space (often noted as a metamodel) may require a large amount of simulations and can be very expensive to compute, but it may be capable of approximating global behavior of the underlying function f . Since Phase I only attempts to determine promising subregions of the search space, the model can be constructed in a coarser manner.

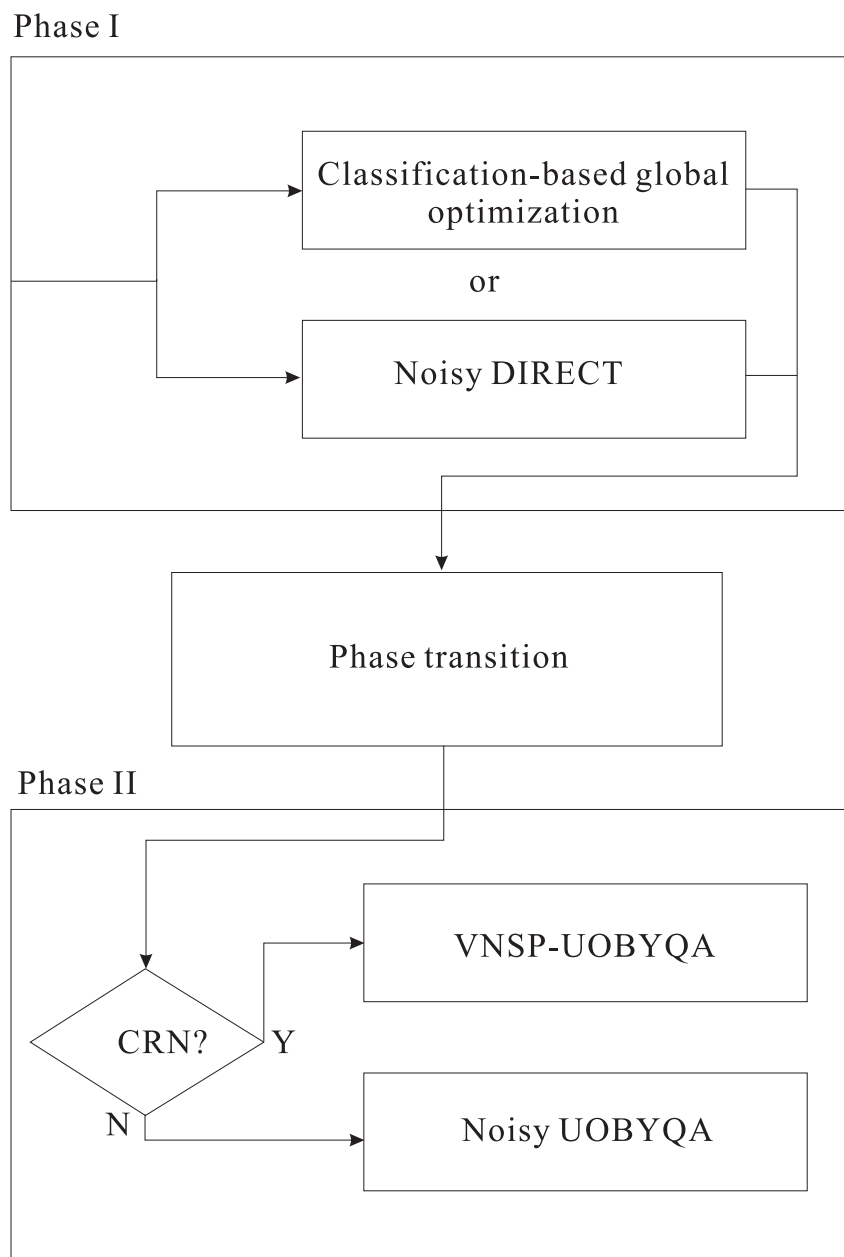


Figure 3: The two-phase WISOPT structure.

We are indeed interested in the behavior of a simple indicator function:

$$I(x) = \begin{cases} 1, & \text{for } x \text{ in a promising subregion} \\ 0, & \text{otherwise,} \end{cases} \quad (1.16)$$

where promising subregions in the method correspond to level sets. This function gives sufficient information to determine where a subregion is located. Approximating the indicator function I is simpler than approximating the underlying function f , especially in a high dimension case. Normally, we sample dense designs in the subregion and sparse designs out of the subregion.

In the classification-based global search, we utilize the predicting power of a classifier: the classifier works as a surrogate function for the indicator function I , which reveals the location of promising subregions. A classifier is a cheap mechanism to predict whether new samples are in promising subregions or not, thus we can generate a dense collection of points in these subregions. Figure 4 illustrates the local regions (the dotted circles and the level sets) and the samples (the ‘+’s). The method fits well to the theme of the Phase I optimization, which is to identify local regions. In fact, the method is relatively insensitive to noise, because the simplification step (1.16) smoothes out the occurrence of noise. That is reason we normally do not use replicated samples in training the classifiers. Setting the algorithm may require specific parameters; for example, we have to potentially understand the proper number of samples to use in training the classifiers.

Classification forms a major branch in in *machine learning/data mining* [51, 77]. In the past couple of years, there has been increasing interest on techniques interfacing optimization and machine learning. For example, Kim and Ding [60] have implemented data mining tools in an engineering design optimization problem. Mangasarian [74] has enhanced optimization robustness in support vector machines.

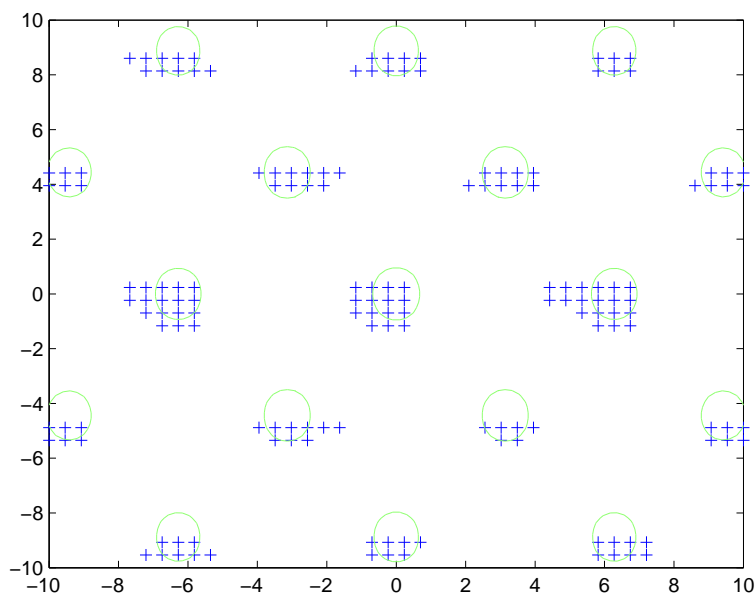


Figure 4: Predicated local subregions of the Griewank function. The function has local optimums in each subregion (circles) and the global optimum at $[0, 0]$.

The Noisy DIRECT algorithm The DIRECT optimization method [37, 38, 58, 59] is a deterministic global optimization algorithm for bound-constrained

problems. The algorithm, first motivated by Lipschitz optimization [59], has proven to be effective in a wide range of application domains. The algorithm centers around a space-partitioning scheme that divides large hyperrectangles into small ones. Promising hyperrectangles are subject to further division. Figure 5 provides an illustration of the algorithm on the Goldstein Price function. The algorithm therefore proceeds to gradually explore promising subregions.

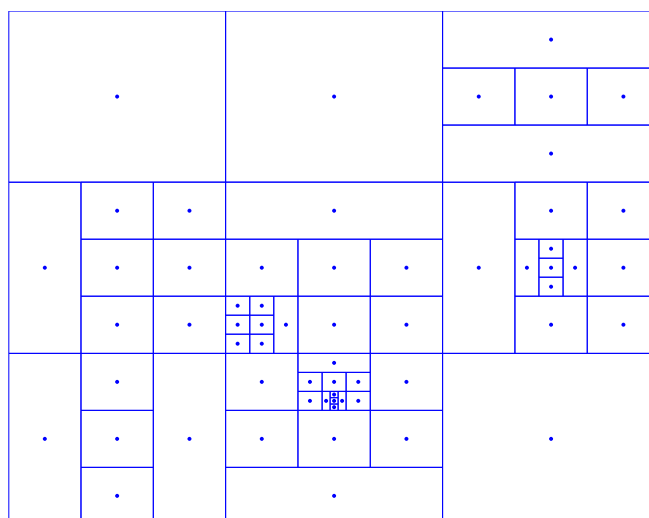


Figure 5: The DIRECT optimization algorithm on the Goldstein Price function. The contour plot of the Goldstein Price function is shown in Figure 15.

When the objective function is subjected to uncertainty, some crucial operational steps of the DIRECT algorithm are affected. For example, the choice of potentially optimal hyperrectangles becomes incorrect because of

the noisy function values, possibly misleading the algorithm to search in inferior regions. We modify the original DIRECT algorithm using a simple approach - multiple replications are sampled to reduce output uncertainty. We expect the algorithm to proceed correctly as in the deterministic case. However, we must face the issue of handling the tradeoff between two design goals: *efficiency of the algorithm* versus *total computational effort*. Since the objective function is often computationally expensive to evaluate, we must be very cautious in using function evaluations. On the other hand, we need to maintain a certain precision in the functions for correctness of the algorithm. In our modification, we apply Bayesian techniques to derive a posterior distribution for the function output at each point, and incorporate the distribution information into the algorithm to determine an appropriate number of replications to be used.

The parameters for this algorithm are easy to set. Since deterministic DIRECT is designed for both global and local optimization, one should note that it is desirable to terminate the algorithm at a reasonable time, at which sufficient information of local regions can be identified. This can avoid unnecessary function evaluations for handling comparisons that are really dominated by noise.

The phase transition Using the evaluated samples in Phase I, the phase transition procedure consists of a non-parametric local quadratic regression

method to determine the appropriate subregion size. Being different from regular regression methods, which use the entire set of samples in the domain to construct one model, local regression makes a prediction (at a point) using a local model based on samples within a ‘window size’, thus the approach values the local behavior of a function more. ‘Non-parametric’ means the regression model is not from a single parametric family. It is presumed that the samples outside the local region have a slight relevance to the current prediction. In our procedure, we treat the resulting ‘window size’ as our subregion radius.

A sequence of good starting points is generated, satisfying the criteria: (a) each starting point is the center of a subregion, (b) the subregions are mutually separated. The sequence of starting points and the subregion sizes are passed to Phase II for local processing, possibly in a parallel setting.

Extended UOBYQA algorithms In Phase II, the deterministic UOBYQA algorithm is applied as the base local search method and is extended for noisy function optimization. The method is an iterative algorithm in a trust region framework [80], but it differs from a classical trust region method in that it creates quadratic models by interpolating a set of sample points instead of using the gradient and Hessian values of the objective function (thus making it a derivative-free tool). Besides UOBYQA, other model-based software include WEDGE [75] and NEWUOA [86]. We choose UOBYQA, because it

is a self-contained, efficient and robust algorithm.

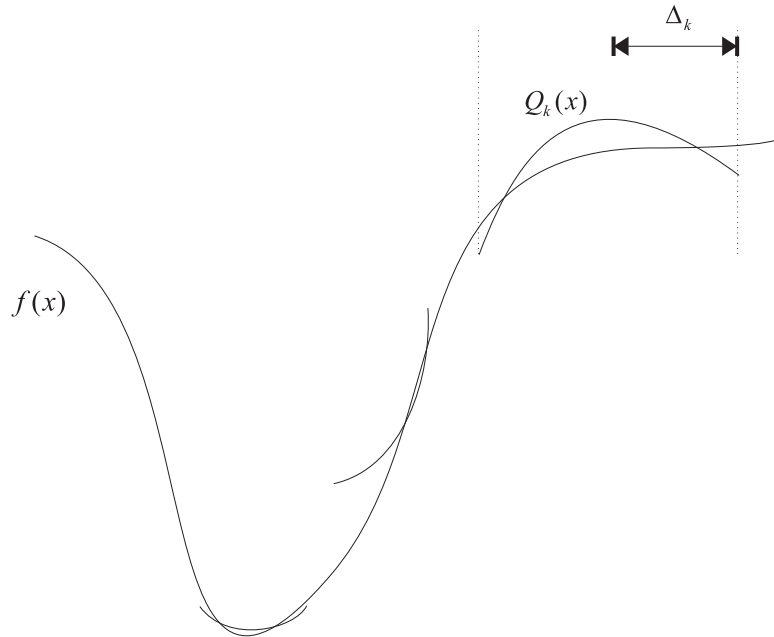


Figure 6: In the model-based approach, a sequence of quadratic models are constructed to approximate the objective function.

We developed variants of the original UOBYQA, called the VNSP-UOBYQA and the Noisy UOBYQA, that are adapted for noisy optimization problems. The extension idea is similar to that of the Noisy DIRECT algorithm. We sample multiple replications per point to reduce variance and apply Bayesian techniques to guide appropriate sampling strategies to estimate the objective function. The two algorithms employ different mechanisms in the sampling process. The VNSP-UOBYQA determines appropriate replication numbers by whether sufficient reduction is identified in the trust-region subproblem, while the Noisy UOBYQA determines the number by whether the quadratic

models can be shown to be stable or not. Generally speaking, when CRN is implemented, the noise is relatively easy to handle because it is correlated among sites. Therefore, it is shown that the VNSP-UOBYQA has better convergence properties.

1.4 Outline of the Thesis

The general theme of thesis centers around the WISOPT package, which is designed based on the two-phase optimization framework. Phase I techniques are described in Chapter 2 and Phase II techniques are described in Chapter 3.

More specifically, the first section of Chapter 2 presents the classification-based global search. The detailed procedures of applying several types of classifiers are included, together with two special features: a voting scheme to assemble multiple classifiers and imbalanced data handling. The second section introduces and explains the development of the Noisy DIRECT algorithm. We will first describe the deterministic DIRECT algorithm, then analyze how to enhance the algorithm under uncertain conditions. Several numerical examples and a simulation problem are presented. The third section is on the procedure of the phase transition.

Chapter 3 contains the details about the Phase II noisy versions of the UOBYQA algorithm. The ideas of the extensions of both the VNSP-UOBYQA

and the Noisy UOBYQA are similar: the Bayesian posterior distributions for the parameters of the quadratic model are derived and further we can estimate the stability of the algorithms. Since the VN-SP-UOBYQA is in the CRN setting, the corresponding section is written in a more rigorous manner, with the convergence proof of the algorithm provided.

We show in Chapter 4 two real-world simulation optimization examples. We aim to fine tune the simulation parameters in the Wisconsin Breast Cancer Epidemiology model, which is from a collaborative project with researchers in the Medical School at the University of Wisconsin. In the second example, we optimize the shape of a type of microwave coaxial antenna for hepatic tumor ablation, which is a current ongoing project with the Biomedical Engineering Department at the University of Wisconsin.

In Chapter 5 we demonstrate a special simulation-based problem in dynamic programming. This type of simulation problem has an internal time structure – the objective function is not a pure black-box stochastic function, but is constituted of a sequence of costs along the time stages. We modify the rollout algorithm from neuro-dynamic programming, using a similar Bayesian analysis to that outline above to improve the simulation efficiency in training and in deriving optimal controls at each stage. We illustrate the effectiveness of our new algorithm using an example in fractionated radiotherapy treatment. This approach to using Bayesian analysis in neuro-dynamic programming effectively generalizes the methods of this thesis to time domain

problems, and leads to further possible applications in other optimization contexts.

Chapter 2

The Phase I Methods

Methods in Phase I provide a global search over the entire domain Ω (refer to Figure 3), with the goal of identifying several promising subregions for Phase II local optimization. Two Phase I methods – the classification-based global search and the Noisy DIRECT method, are presented in Section 2.1 and Section 2.2, respectively. Both methods are designed to generate densely distributed samples in promising subregions. All of the evaluated samples are passed to a phase transition module, described in Section 2.3, to identify the locations and sizes of the subregions.

Compared with the Noisy DIRECT method, the classification-based global search has shown more robustness to noise. The simplification step of the complex objective function to an indicator function smoothes out the noisy effects and reduces the occurrence of uncertainty. Moreover, the method can handle high dimensional problems, because it works with a simplified model. The drawback of applying the method is the trickiness in setting the parameters; for example, choosing a proper number of samples for training

the classifiers. The more samples that are available, the better the performance of the method. Insufficient samples may lead to missing promising subregions, especially the small, hard to detect subregions.

The Noisy DIRECT algorithm is easy to implement, with a standard setting of parameters. It generates samples gradually during the exploration of the domain. The density of samples is in proportion to the goodness of the objective values. In this sense, the method is able to save computational effort because the smooth density distribution is better than the two-level density distribution as in the classification-based global search. The drawback of the method is that it becomes inefficient when handling high dimensional problems, i.e., the dimension $n > 30$.

When the problem is of moderate and small dimensions, we suggest the Noisy DIRECT method. For the reasons stated above, the method is more economical in function evaluations and it has a succinct way to operate. For high dimensional problems, the classification-based global search is a better choice.

For both of these methods, we do not distinguish the cases of CRN and white noise. In fact, there are no special mechanisms implemented to take advantage of CRN. For the classification-based global search, we use a single replication (the default value) for each sample, because we observe that the method handles noise well. If we use multiple replications (with a replication number greater than 1) for all the samples in the method (which is the

sample-path optimization idea), the algorithm may perform better due to a reduction in the variance of the noise. If CRN is implemented as well, the method may perform even better because of the reduction of covariance among the samples. For the Noisy DIRECT method, the multiple replications approach is employed and the replication number is automatically determined by the algorithm.

2.1 Classification-Based Global Search

Classification, as a major branch in machine learning/data mining, uses statistical or artificial intelligence tools to classify objects into two (or multiple) categories based on available information. Our method of classification-based global search utilizes the predicting power of classifiers: the classifier acts in the role of an indicator function which reveals locations of promising regions. Additional features of the method are: (a) more reliable predictions obtained using a voting scheme combining the options of multiple classifiers, (b) a data pre-processing step that copes with imbalanced training data.

2.1.1 The Idea of Classification-Based Global Search

Since the goal of Phase I is to locate promising subregions rather than to determine the exact solution, one may not care about how an objective function f behaves over the whole space, instead, caring about the behavior of a

simple indicator function:

$$I(x) = \begin{cases} 1, & \text{for } x \text{ in a promising subregion;} \\ 0, & \text{otherwise.} \end{cases}$$

This function provides sufficient information to determine where subregions are located. Approximating the indicator function I is much simpler than approximating the underlying function f , especially in high dimensional cases.

In our approach, a classifier works as a surrogate function for the indicator function I . A classifier is a cheap mechanism to predict whether new samples are in a promising subregion or not. The target promising subregion is often defined as a certain level set

$$L(c) = \{x \mid f(x) \leq c\}, \quad (2.1)$$

where c is an adjustable parameter that quantifies the volume of the set. The value of c may be determined, for example, as a quantile value of the responses.

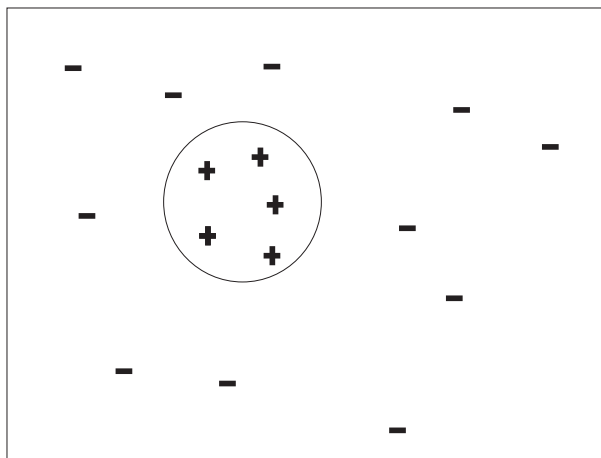
One important component of the classification-based approach is to prepare the training set. We generate spacing-filling samples (points) all over the domain. A straightforward way to do this is by mesh grid sampling; other advanced techniques include the *Latin Hypercube Sampling* (LHS) [92]. The training set is further divided into two classes: a positive class for points in $L(c)$ and a negative class for the rest (as illustrated in Figure 7(a)). The lower the value of c , the smaller the size of the level set $L(c)$. Adjusting it can balance portions of samples in both classes.

The classifier is built on the training data to create appropriate decision rules. (Details of the training process will be discussed later.) The decision rule is then applied to predict the membership of a set of more refined space-filling points – the evaluation set (see Figure 7(b)). As a consequence, the classification implicitly partitions the domain space into positive and negative zones. Typically, we expect the process to greatly increase the chance of generating refined points in promising subregions.

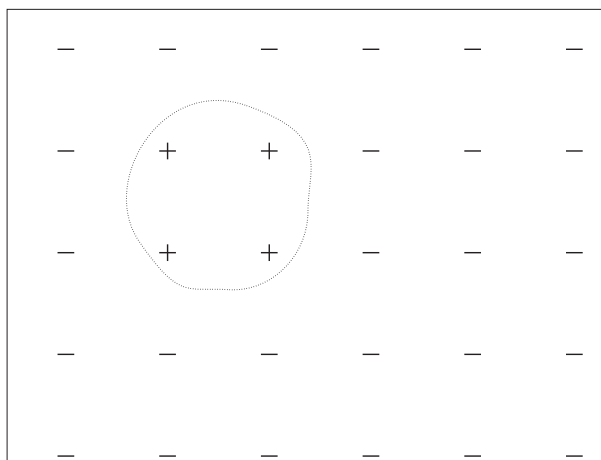
The identified promising points are then evaluated by the function f and passed to the phase transition module.

2.1.2 A Voting Scheme to Assemble Multiple Classifiers

Given a particular training set, a variety of classifiers can be generated, including typical classifiers such as *Support Vector Machine* (SVM) [73], *Decision Trees* [87], and *k-Nearest Neighbor* (k-NN) [77]. Unfortunately, it is almost impossible for any individual classifier to perform satisfactorily for all classification tasks. We propose a voting scheme to combine the predictions of various classifiers to build a single robust classifier, which typically increases the prediction power over any component classifier. The voting idea of classifiers is addressed in many research works, such as Kotsiantis and Pintelas’s *Agent-Based Knowledge Discovery* (ABKD) [64] and Optiz et al.’s bagging and boosting [81].



(a) For the training set, samples in the level set $L(c)$ are classified as positive and the others are classified as negative. The solid circle represents the level set $L(c)$.



(b) Classification is performed on the evaluation set, which consists of more refined space-filling samples. As a result, four points are predicted as positive and rest are negative.

Figure 7: In Phase I, classifiers determine new refined samples in a promising subregion.

We investigate the use of 6 candidate classifiers: 4 SVMs with linear, quadratic, cubic and Gaussian kernels, C4.5 (a decision tree classifier) and 5-NN. The performance measure of each classifier is summarized in a *confusion matrix* (Table 1); for example, the accuracy of a classifier is defined as $acc = \frac{a+d}{a+b+c+d}$. Similarly, the accuracy on positive samples is $acc^+ = \frac{a}{a+b}$ and the accuracy on negative samples is $acc^- = \frac{d}{c+d}$. Kubat et al. [65] provide the geometric-mean (g-mean) measure, a more robust measure taking considerations on both classes:

$$g = \sqrt{acc^+ \cdot acc^-} = \sqrt{\frac{a}{a+b} \cdot \frac{d}{c+d}}.$$

Inclusion of each classifier to the ensemble is selective based on its performance evaluations. The ones that fail a performance test are excluded, e.g., failing a test criterion requiring the g-mean $g \geq 0.5$.

Table 1: Confusion matrix

		Predicted:	
		Positive	Negative
True:	Positive	True Positive (a)	False Positive (b)
	Negative	False Negative (c)	True Negative (d)

We describe the voting scheme for multiple classifiers as follows:

Procedure 1. *The voting scheme:*

1. *Split the input training set T into two subsets, denoted as training subset T_1 and testing subset T_2 . For example, we may use 75% of randomly*

selected samples to make the training subset T_1 , and rest are treated as the testing subset T_2 .

- 2. Perform a prior performance test: train each classifier on the training subset T_1 and evaluate it with the samples in the testing subset T_2 . If the classification accuracy is not assured, we will discard the classifier.*
- 3. Classifiers that pass the performance test are trained on the original training set T . In the evaluation process, assign new samples to the class which is majorally voted.*

2.1.3 Handling Imbalanced Dataset

The aforementioned level set parameter c in (2.1) plays a crucial role in setting up the training set. We intend to drop c as much as possible, so that the level set $L(c)$ only represents exceptionally good points. For example, we may choose c as the 10% quantile value of the observed responses, such that $L(c)$ represents the top 10% region in the domain. However, $L(c)$ cannot be very small, because the dataset that naturally follows from such a choice can significantly hinder the classifier performance. When the negative class heavily outnumbered the positive class, the under-represented positive class has a low accuracy of being correctly classified. The class imbalance problem is prevalent in real-world settings, e.g., the detection of fraud credit card usage [14] and the detection of oil spills in satellite radar images [65].

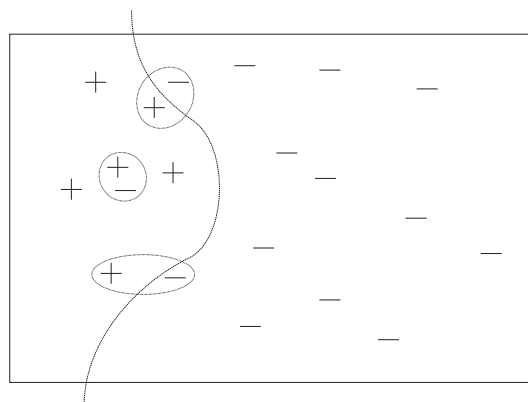
There are two mainstream techniques to cope with the class imbalance problem: (a) resize the training set, including under-sampling the majority class [66] and over-sampling the minority class [15], and (b) assign different misclassification costs [28]. We adopt the approach to carrying out both under- and over-sampling. The one-sided selection procedure [66] is applied for under-sampling and positive samples are simply duplicated once for over-sampling. The one-sided selection constitutes detection and removal of the negative samples that participate in so called *Tomek links* [107] (see Figure 8). These samples are generally considered border samples or samples affected by noise, and are therefore detrimental to classifier fitting. Denoting $d(x, y)$ as the distance between two points x and y , a Tomek link is defined as a pair of two points x and y from opposite classes, satisfying that there is no third sample z such that $d(x, z) < d(x, y)$ or $d(y, z) < d(x, y)$.

Our pre-processing procedure to adjust the imbalanced data is described as follows.

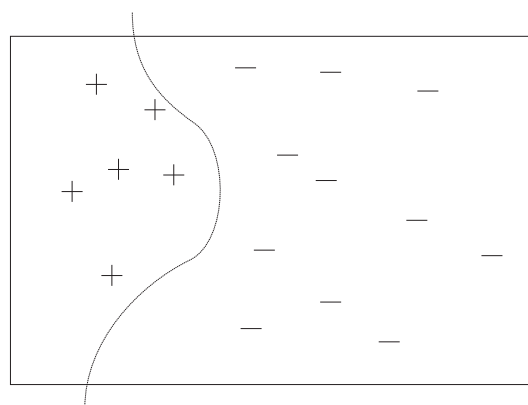
Procedure 2. *The data pre-processing step:*

1. *Under-sample the negative class using one-sided selection:*

- (a) *Keep all the positive samples unchanged. To obtain a consistent subset C of the original training set T , train 1-NN classifier with the positive samples plus one randomly chosen negative sample. Test the 1-NN rule on the rest of the samples in the set T . The new*



(a) Determine the pairs of Tomek links



(b) Remove the negative samples participating Tomek links

Figure 8: Cleaning the dataset with Tomek links

subset C will consist of the misclassified samples plus the samples used for training. In doing this, we derive a consistent subset C of T such that all the samples in T can be correctly predicted using the 1-NN rule on C .

(b) Detect the Tomek links in C and remove the associated negative samples.

2. Over-sample the positive class by duplicating all the positive samples once.

The above pre-processing procedure typically returns a much more balanced training set. Indeed, carrying out the under-sampling sampling step (Step 1) multiple times can significantly reduce the negative samples, but our numerical examples show that it is not necessary to do so, because one under-sampling step often performs well enough. A small example on the Rosenbrock function shows the effectiveness of our data balancing procedure. Without using the procedure in the training data, we obtained the classifier accuracies $acc^+ = 0.20$ and $acc^- = 0.99$, which resulted a g-mean value 0.4450; while with the procedure implemented, we had $acc^+ = 0.89$ and $acc^- = 0.90$, which is equivalent to a much better g-mean value 0.8950.

2.1.4 General Procedure

The general procedure is summarized in detail (refer to the flow chart in Figure 9).

Procedure 3. *The classification-based global search:*

1. *Generate coarse space-filling samples and evaluate them via simulation. Choose an appropriate value of c , and split the samples into positive (points in $L(c)$) and negative samples (points outside $L(c)$). Typically, we suggest to set c as the 10% quantile value.*
2. *Use the pre-processing procedure (Procedure 2) to generate a balanced dataset. 6 classifiers are considered, but those passing the performance test are used.*
3. *Given the training set, derive an ensemble of classifiers (Procedure 1).*
4. *Generate a fine space-filling evaluation set either by the grid sampling or the LHS. Determine the membership points by classification.*
5. *For those points that are predicted to lie in $L(c)$, evaluate them via simulation.*

2.1.5 Numerical Examples

The classification-based global search is relatively insensitive to noise. The accuracy of the method is highly related to the accuracy of the training set,

Phase I- Classification-based global search

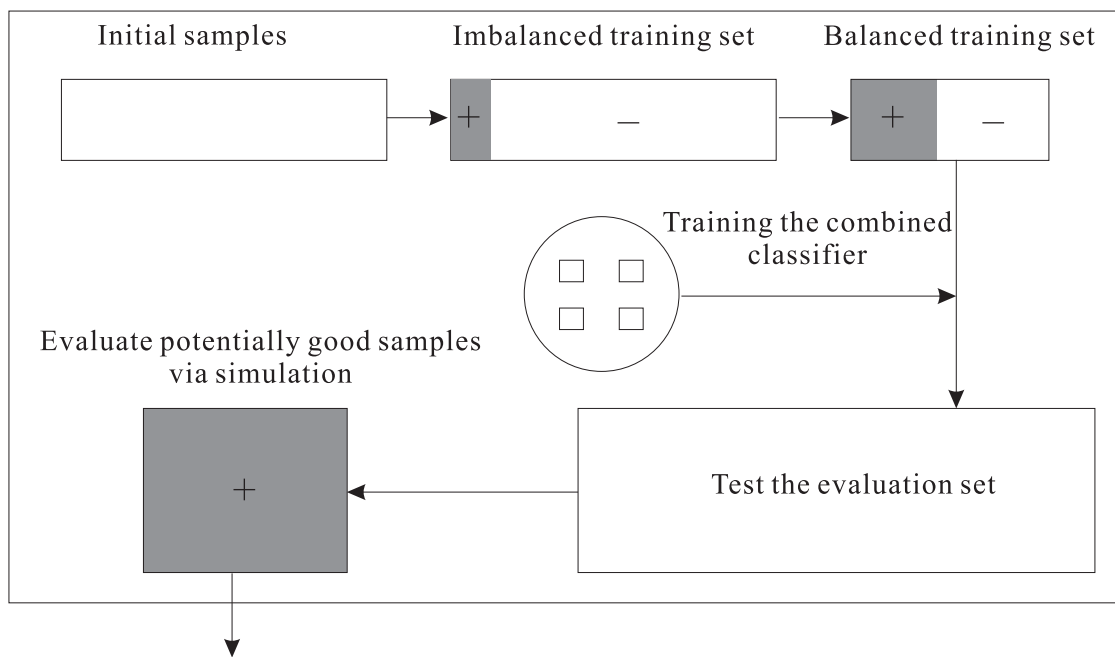


Figure 9: The classification-based global search

i.e., whether a positive sample is indeed a positive sample, which is equivalent to the positive sample being located in a subregion of the underlying objective function. We show here that the estimated level sets (represented by positive points) are quite insensitive to the noisy data.

We plot several estimated level sets of a small example in Figure 10. The test function was the Gauss function with additive noise:

$$F(x, \xi(\omega)) = 1 - \exp(-20(x_1 - 0.25)^2 - 2(x_2 - 0.25)^2) + \xi(\omega).$$

$\xi(\omega)$ here is a noise term distributed as $N(0, \sigma^2)$. We applied the grid sampling method to generate 400 samples in the range $[0, 0.8] \times [0, 0.8]$. Of all the samples, the top 10% were considered as positive samples and plotted as ‘+’, and the rest were considered as negative samples and plotted as ‘.’. As we observed, when we simplified all the samples as positive or negative, most samples (in the first two figures) were correctly labeled. When the noise was intense, i.e., the third figure, it could produce a biased training set.

As a second example, we applied the entire classification-based global search to the FR function

$$F(x, \xi(\omega)) = (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 \\ + (-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2)^2 + \xi(\omega),$$

which has two local subregions with two local minimizers. $\xi(\omega)$ is a zero-mean normal quantity with variance $\sigma^2 = 0.1$. Following the steps in Procedure 3, we first generated 400 grid samples which were labeled positive and negative

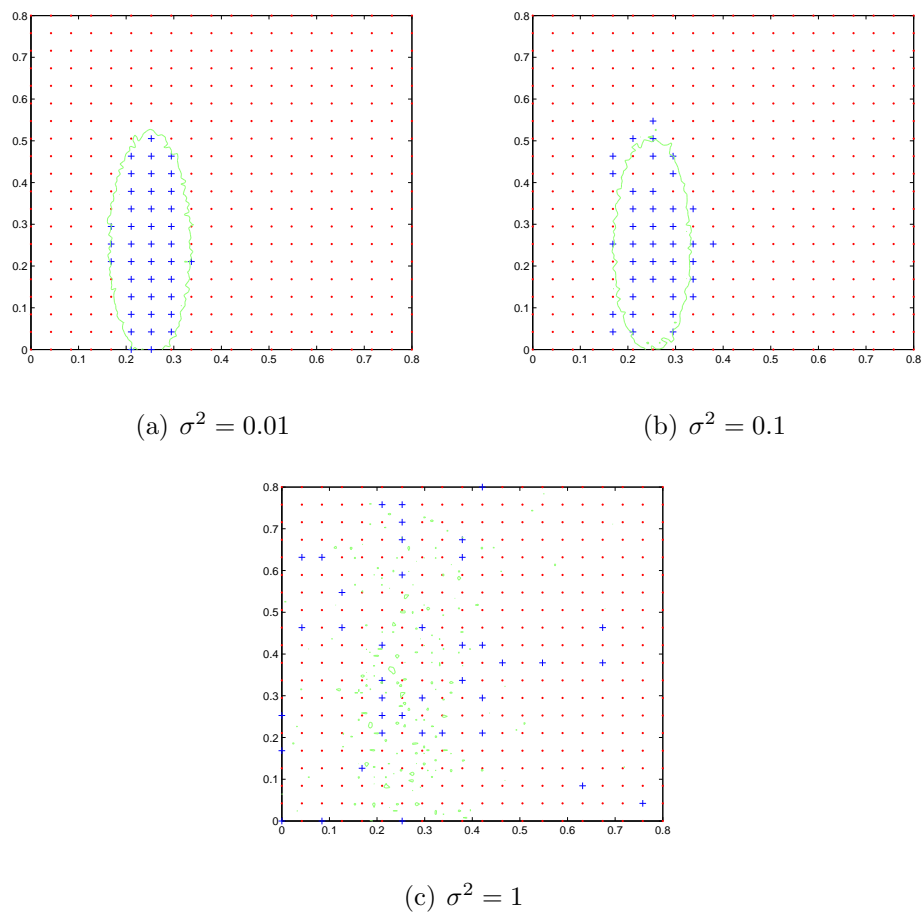


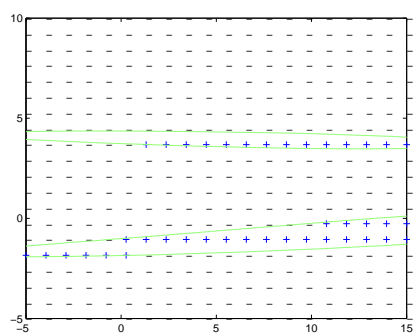
Figure 10: Estimated level sets based on noisy data.

(Figure 11 (a)). The green curves represent the real level sets (the real local subregions). After we applied the data pre-processing step, the data set was much more balanced. A significant amount of negative samples were removed from the training data set while all the positive samples were retained and duplicated (Figure 11 (b)). Note that the negative samples removed were mostly likely in inferior regions. The ensemble of classifiers was trained and used to predict a vast amount of new samples (10,000 samples), generated by the LHS sampling method. Once the ensemble of classifiers was trained, the prediction process was very fast. In the figures, we plot only the new positive samples identified. Figure 11 (c) shows that all the predicted samples that have been evaluated in the domain. We finally were able to generate a dense sample set in the promising subregions.

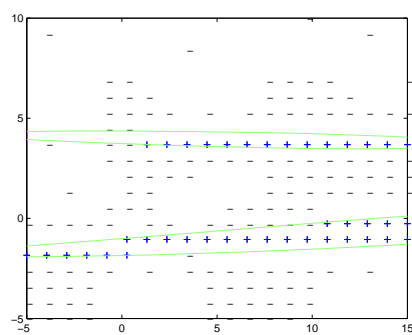
2.2 The Noisy DIRECT Algorithm

DIRECT (DIviding RECTangles) is a deterministic global optimization algorithm for bound-constrained problems. The algorithm, based on a space-partitioning scheme, performs both global exploration and local exploitation. We modify the deterministic DIRECT algorithm to handle simulation-based optimization problem (1.1)

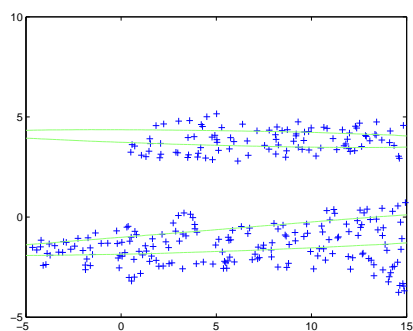
$$\min_{x \in \Omega} f(x) = \mathbb{E} [F(x, \xi(\omega))],$$



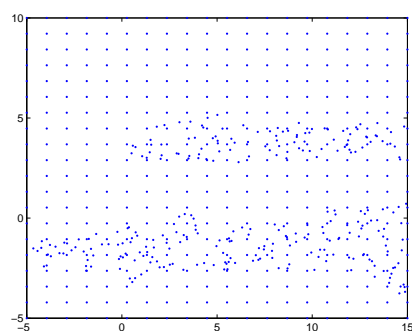
(a) Original training set



(b) Balanced training set



(c) Predicted samples



(d) All samples

Figure 11: Apply the classification-based global search on the FR function.

where

$$\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}.$$

We adopt a simple approach that replicates multiple function evaluations per point and takes an average to reduce functional uncertainty. Particular features of the DIRECT method are modified using acquired Bayesian sample information to determine appropriate numbers of replications.

The extended DIRECT algorithm is suited for noisy function optimization problems. The algorithm is a sampling approach, that only uses objective function evaluations. We have applied the new algorithm in a number of noisy global optimizations, including an ambulance base simulation optimization problem.

The remainder of the section is arranged as follows. In Section 2.2.1 we will describe the DIRECT optimization algorithm. In Section 2.2.2 we focus on the modifications of the algorithm, including a variance controlling rule to determine the replication numbers. In Section 2.2.3, we will present test examples and comparison of the algorithm with other algorithms.

2.2.1 The DIRECT Optimization Algorithm

The DIRECT algorithm is defined for Lipschitz continuous objective function, a class that includes some non-smooth functions. Bounds on the range of the simulation parameter x are required in the design of the algorithm.

The feasible region starts as a single hyperrectangle that is internally normalized to a unit hyperrectangle. The algorithm partitions the hyperrectangle into a collection of smaller hyperrectangles and evaluates the objective function at their center points (Figure 5). Potentially optimal hyperrectangles are identified and passed to the next iteration for further partitioning and investigation. The DIRECT algorithm will converge to the global optimum of the objective function for dense enough sampling, but the search process may consume a large amount of function evaluations. For the algorithm to be effective, the typical dimension of a problem, as cited in [59], should be less than 30.

We will give a brief description of the DIRECT algorithm, including two major component steps: (a) partitioning hyperrectangles, and (b) identifying potentially optimal hyperrectangles.

Partitioning hyperrectangles For each hyperrectangle, let \mathcal{D} be the coordinate directions corresponding to the largest side lengths, δ be one third of the largest length, and c be the center point. The function will explore the objective values at the points $c \pm \delta e_i$, for all $e_i \in \mathcal{D}$, where e_i is the i th unit vector. The hyperrectangle will be trisected along the dimensions in \mathcal{D} , first along dimensions whose objective values are better. The procedure continues until each point $c \pm \delta e_i$ occupies a single hyperrectangle.

In a two-dimensional case, two possible partitioning scheme are illustrated

as follows. Since we only perform trisections, the length of any side in the unit hyperrectangle can possibly be 3^{-k} , $k = 1, 2, \dots$

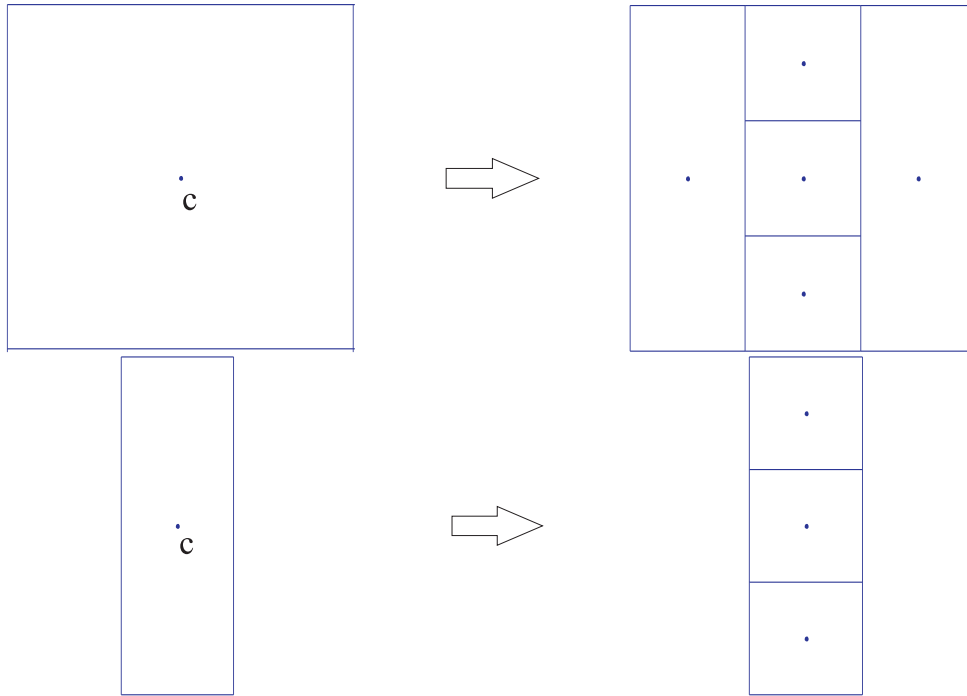


Figure 12: Partitioning hyperrectangles

Identifying potentially optimal hyperrectangles Selection of potentially optimal hyperrectangles combines the purposes of both global and local searches. Let \mathcal{H} be the index set of existing hyperrectangles. For each hyperrectangle $j \in \mathcal{H}$, we evaluate the function value at the center representing point $f(c_j)$ and note the size of the hyperrectangle α_j . The size α_j is computed as the distance from the center point to the corner point. A hyperrectangle $j \in \mathcal{H}$ is said to be potentially optimal ($j \in \mathcal{S}$) if there exists

a constant \tilde{K} such that

$$f(c_j) - \tilde{K}\alpha_j \leq f(c_i) - \tilde{K}\alpha_i, \forall i \in \mathcal{H}, \quad (2.2)$$

$$f(c_j) - \tilde{K}\alpha_j \leq f_{min} - \epsilon|f_{min}|. \quad (2.3)$$

In the above expressions, f_{min} is the lowest function value available and ϵ is a parameter that balances between global and local search. The parameter is typically nonsensitive and set as 0.0001.

An equivalent interpretation of the process of selecting potential optimal rectangles is illustrated in Figure 13. First sort the hyperrectangles in groups according to the size α . Each hyperrectangle is plotted in the figure as a black dot in accordance with its center function value $f(c_j)$ and size α_j . Criteria (2.2) and (2.3) correspond to selecting rectangles on the lower convex hull of the graph (hyperrectangles that are selected are denoted as white dots). The introduction of ϵ may result in exclusions of good hyperrectangles in the smaller size groups. Thus ϵ is considered as a balancing parameter between local and global search. As noted from the figure, the best hyperrectangle in the largest size group is always selected. The author claims that the algorithm will eventually converge to the global optimum because the maximum size $\max_j \alpha_j$ decreases to zero and the entire search space is thoroughly explored.

With the two key component steps available, the process of the DIRECT algorithm is straightforward and we summarize the steps of the algorithm

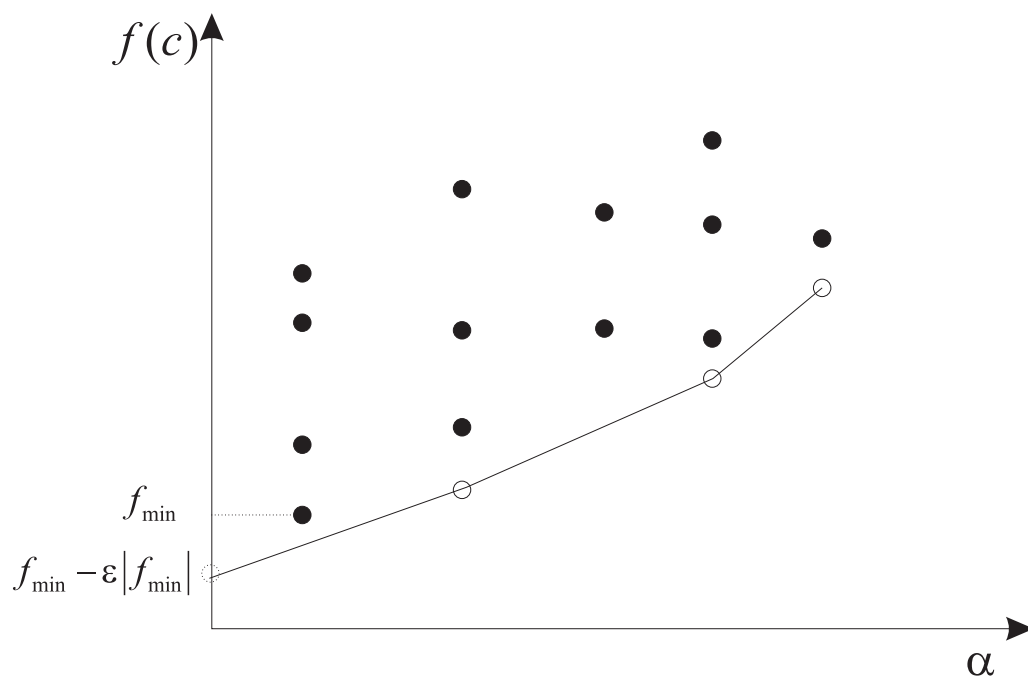


Figure 13: Identifying potentially optimal hyperrectangles

below. More details on this algorithm can be found in [59].

Procedure 4. *The DIRECT optimization algorithm*

Given the lower bound l and upper bound u for the optimization problem.

1. *Normalize the domain as a unit hyperrectangle.*
2. *For iterations $k = 1, 2, \dots$*
 - (a) *Identify the potentially optimal hyperrectangle set $\mathcal{S} \subset \mathcal{H}$.*
 - (b) *Divide the hyperrectangles in \mathcal{S} according to the partitioning scheme.*
3. *Terminate the algorithm when the total number of function evaluations hits a maximum limit or no improvement of the objective function values is observed after a number of iterations.*
4. *Return the best point found.*

2.2.2 Modifications

We will focus on describing our modifications to the deterministic DIRECT algorithm. When noise is present in the objective function output, there are two problematic points in the original DIRECT algorithm.

- **Incorrect potentially optimal hyperrectangle set \mathcal{S}** As we have discussed in Section 2.2.1, $\mathcal{S} \subset \mathcal{H}$ is determined according to the function values $f(c_j), j \in \mathcal{H}$ and size α_j (see the equivalent process in

Figure 13). When the objective values are volatile, the resulting set \mathcal{S} is unstable. Therefore, the algorithm working with the unstable set will possibly miss important search regions or waste computational effort in redundant regions.

- **Biased solution** The DIRECT algorithm always keeps track of the best point and corresponding best objective value f_{min} . If the objective function is noisy, we cannot assert that the point with the lowest objective value f_{min} is the best point. In such cases, the algorithm may return a solution that is incorrect due to bias from noise.

As we have mentioned above, our primary approach is to sample multiple replications per point and use the averaged value in the algorithm. In order to choose the appropriate number of replications, we first construct a Bayesian posterior distribution for the functional output at each point, and incorporate the distribution information to help determine the appropriate number of replications (see Figure 14). Based on the replication samples, the Bayesian posterior distribution provides not only mean but also variance information of the functional output that can be utilized by the algorithm. At each iteration of the algorithm, our modification focuses on determining and evaluating a necessary number of replications at each point, such that the algorithm can maintain a certain accuracy. The accuracy here explicitly addresses the two problematic issues: the accuracy of the identified promising set \mathcal{S} and the

accuracy of the final output of the algorithm.

Based on the mean and variance information, our sampling scheme may generate different numbers of samples for different points. For example, in Figure 14, since Point 1 has a large mean function value (or the volatility is relatively small), 3 replications are enough to obtain the desired accuracy. The number 3 is the minimum number of replications in order to construct the posterior distribution. For Point 3 and Point 5, since their objective values are small (or volatilities are relatively large), more replications are needed.

Following the discussion in Section 1.2, in the Bayesian framework, the unknown mean $\mu(c_j)$ and variance $\sigma^2(c_j)$ of $F(c_j, \xi(\omega))$ are considered as random variables. Suppose we have r_j replicated samples at the point c_j , we can derive two versions of posterior distributions, the normal version

$$\mu(c_j)|X \sim N(\bar{\mu}(c_j), \hat{\sigma}^2(c_j)/r_j). \quad (2.4)$$

and the t version

$$\mu(c_j)|X \sim St(\bar{\mu}(c_j), r_j/\hat{\sigma}^2(c_j), r_j - 1). \quad (2.5)$$

In the above expressions, $\bar{\mu}(c_j)$ and $\hat{\sigma}^2(c_j)$ are the sample mean and sample variance at the point c_j , respectively.

Monte Carlo validation The goal of our approach is to quantify how the randomness in the objective function affects the identification of the

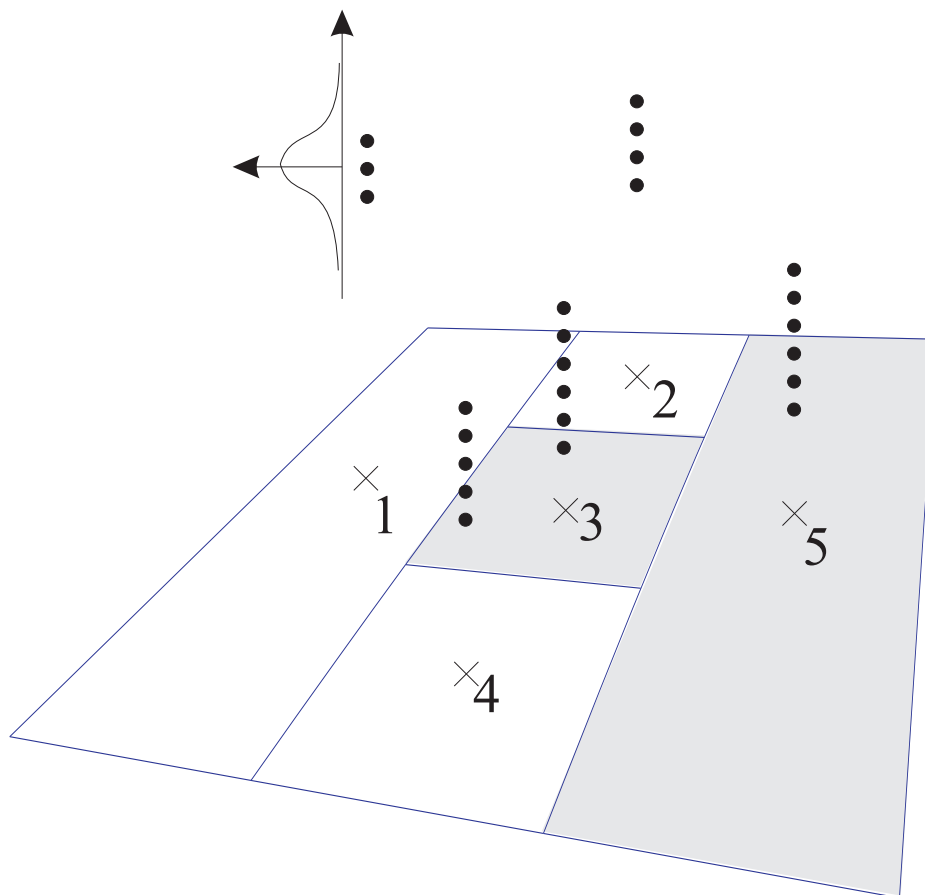


Figure 14: Generate multiple replications at each point and derive Bayesian posterior distributions

promising hyperrectangle set \mathcal{S} . This is done by Monte Carlo validations. Note that the set \mathcal{S} is derived using the observed sample means $\bar{\mu}(c_j)$ in place of $f(c_j)$. As we know, increasing the number of replications r_j at c_j can reduce the uncertainty of the set \mathcal{S} , but in our procedure we want to control the total number of function evaluations $\sum_j r_j$. We do this by sampling the posterior distribution and use the samples to determine if further replications are needed.

Suppose we carry out N_t Monte Carlo trial experiments. In the i th experiment $i = 1, 2, \dots, N_t$, we sample objective values $\tilde{f}_i(c_j)$ from the derived posterior distribution $\mu(c_j)|X, j \in \mathcal{H}$ (cf., (2.4), (2.5)) and plug them into the potentially optimal set identification process (e.g., Figure 13) to generate a trial set $\tilde{\mathcal{S}}_i$. On completion of all the experiments (corresponding to multiple executions of one step of DIRECT), we come up with a collection of trial sets $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2, \dots, \tilde{\mathcal{S}}_{N_t}$. We then test the difference between the set $\tilde{\mathcal{S}}_i$ and the original set \mathcal{S} that is suggested by the default execution of DIRECT. Here, we introduce a volatility controlling rule that requires the trial sets to be close to the set \mathcal{S}

$$\frac{1}{N_t} \sum_i^{N_t} \frac{\text{card}(\tilde{\mathcal{S}}_i \cap \mathcal{S})}{\text{card}(\mathcal{S})} \geq \beta, \quad (2.6)$$

where β is a threshold value that is normally set as 90%. The function card returns the cardinality of a set. The rule is a Monte Carlo validation criterion that guarantees a sufficiently large overlap between $\tilde{\mathcal{S}}_i$ and \mathcal{S} ($\tilde{\mathcal{S}}_i$ and \mathcal{S} are

90% alike on average). Increasing r_j should help reduce the volatility of \mathcal{S} , and thus increase the possibility of satisfying the rule (2.6).

On the other hand, satisfying the above rule necessitates an appropriate number of replication r_j . We adopt a sequential resource allocation procedure. That is, we gradually increase the number r_j until the rule (2.6) is satisfied. In doing this, we attempt to minimize the total number of function evaluations. We will selectively increase

$$r_j := \gamma \cdot r_j$$

for $j \in \mathcal{R}$, where γ is an inflation factor and

$$\mathcal{R} = \bigcup_i^{N_t} \left((\tilde{\mathcal{S}}_i / \mathcal{S}) \cup (\mathcal{S} / \tilde{\mathcal{S}}_i) \right). \quad (2.7)$$

The index set \mathcal{R} is the union of the possible potential optimal rectangles in $\tilde{\mathcal{S}}_i$ and \mathcal{S} , but excluding the intersection of them. That implies we only increase r_j for hyperrectangles that are likely to be potentially optimal, but for which the Monte Carlo experiments give differing output. Note that if rule (2.6) is not satisfied, the set \mathcal{R} is nonempty. The rule (2.6) will be eventually satisfied because when r_j increase to infinity, Monte Carlo samples $\tilde{f}_i(c_j)$ are nonvolatile (generated from delta distributions), and the trial sets $\tilde{\mathcal{S}}_i$ become stable.

Procedure 5. *Determine a stable set \mathcal{S} :*

Given an initial sample size r_0 , a threshold value β , and a number N_t of Monte Carlo experiments. At each iteration of the DIRECT algorithm, the

following procedures should be executed to identify the potentially optimal set \mathcal{S} .

1. Generate r_0 function evaluations for each point c_j for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j \in \mathcal{H}$.
2. Carry out N_t Monte Carlo experiments to generate $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2, \dots, \tilde{\mathcal{S}}_{N_t}$.
3. While the test rule (2.6) is not satisfied, increase $r_j \rightarrow \gamma \cdot r_j$, for $j \in \mathcal{R}$. \mathcal{R} is defined in (2.7). Repeat Step 2.
4. Return the potentially optimal set \mathcal{S} .

Previously evaluated function values can certainly be reused.

Our modification does not change the fact that one of the rectangles in the largest size group is divided. Therefore, the division process will finally generate a dense collection of points filling the entire domain. At the return of the algorithm, the point with the lowest averaged sample mean (or averaged sample response function value) is selected. We cannot guarantee the selected point is the real best point (in term of the underlying function $f(x)$) among the explored points, because the sample mean is still subject to noise. In fact, choosing the best point from the vast candidate points is itself a difficult problem [62]. However, compared to the single replication case in the original method, our approach of returning the point with the best sample mean is much more reliable.

2.2.3 Numerical Examples

In this subsection, several numerical experiments of the Noisy DIRECT algorithm are reported, including a particular simulation optimization problem. We also consider comparing the Noisy DIRECT algorithm against the standard DIRECT and the Snobfit (Stable Noisy Optimization by Branch and Fit) optimization algorithm [55], both of which are well known global optimization methods.

Numerical functions

For a numerical objective function $F(x, \xi(\omega))$, we consider a special case where $\xi(\omega)$ is an additive ‘white noise’ term, thus the form of the objective function becomes

$$F(x, \xi(\omega)) = f(x) + \xi(\omega). \quad (2.8)$$

We assume the noise term is a normally distributed variable $N(0, \sigma^2)$, therefore, the expectation form of the objective is consistent with model (1.1). Using this formulation, we are able to compare to the known solutions of the deterministic function $f(x)$.

The first test function we employed was the Goldstein Price function; see Figure 15 for a contour plot. The function is a two-dimensional function with a global optimum at $(0, -1)$. The global objective value at this point is 3. We used the bounds $[-2, 2] \times [-2, 2]$ in the optimization methods. The iterates of DIRECT on the noiseless Goldstein Price function were shown in

Figure 5. We used the following parameter settings for the Noisy DIRECT algorithm: the initial sample number $r_0 = 3$, the threshold value $\beta = 90\%$, the number of Monte Carlo trial experiments $N_t = 100$, the inflation factor $\gamma = 1.3$, and the maximum number of replications per point $N_{max} = 100$.

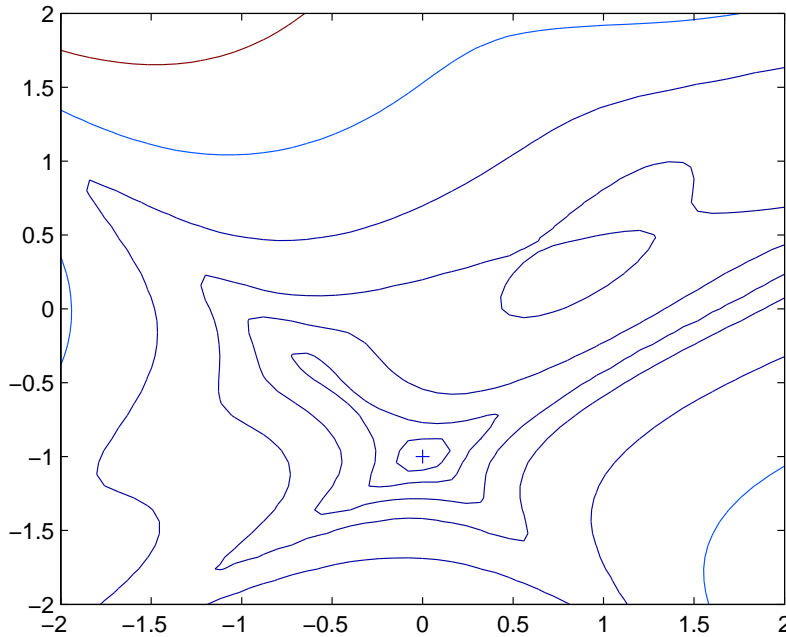


Figure 15: Contour plot of the Goldstein Price function

Table 2 shows the performance of the Noisy DIRECT algorithm for the case $\sigma^2 = 10$. We terminated the algorithm when the total number of function evaluations reached 3000. The column $\bar{F}(x_k)$ records the best value of the averaged sample response function at iteration k , and $f(x_k)$ records the

corresponding best underlying objective function value. In the earlier iterations, the algorithm used relatively few function evaluations. We found only 3 replications were used per point in iterations 1-6. As the iterates got close to the real solution, more and more replications were used to maintain the accuracy of the algorithm. Several points that were near the global solution $(0, -1)$ hit the maximum replication number 100. We generated a replication number plot for the whole optimization process, as shown in Figure 16. It can be shown in this figure that replication numbers increase in promising regions.

Table 2: The performance of the Noisy DIRECT algorithm for the Goldstein Price function, with $\sigma^2 = 10$.

Iteration (k)	$f(x_k)$	$\bar{F}(x_k)$	FN
1	200.54	201.03	15
2	200.54	201.03	21
3	14.92	12.68	39
4	14.92	12.68	63
5	3.64	7.16	81
6	3.64	7.16	111
7	5.84	-0.74	298
8	4.55	2.13	380
9	3.55	3.89	1270
10	3.55	3.72	3010

Based on the same Goldstein Price example, we compared the noisy version DIRECT algorithm against the deterministic DIRECT and Snobfit. See Table 3 for the comparison results. We present both the normal and t version

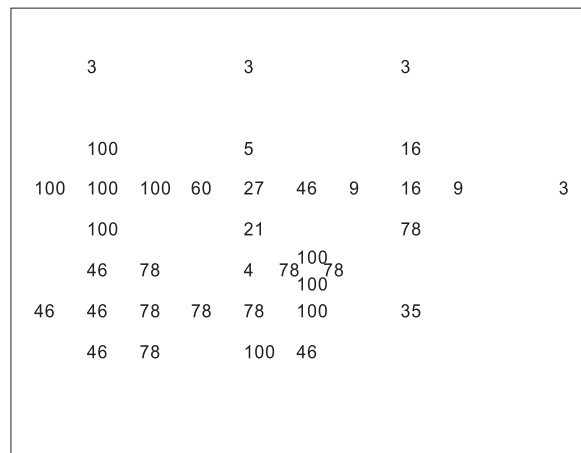
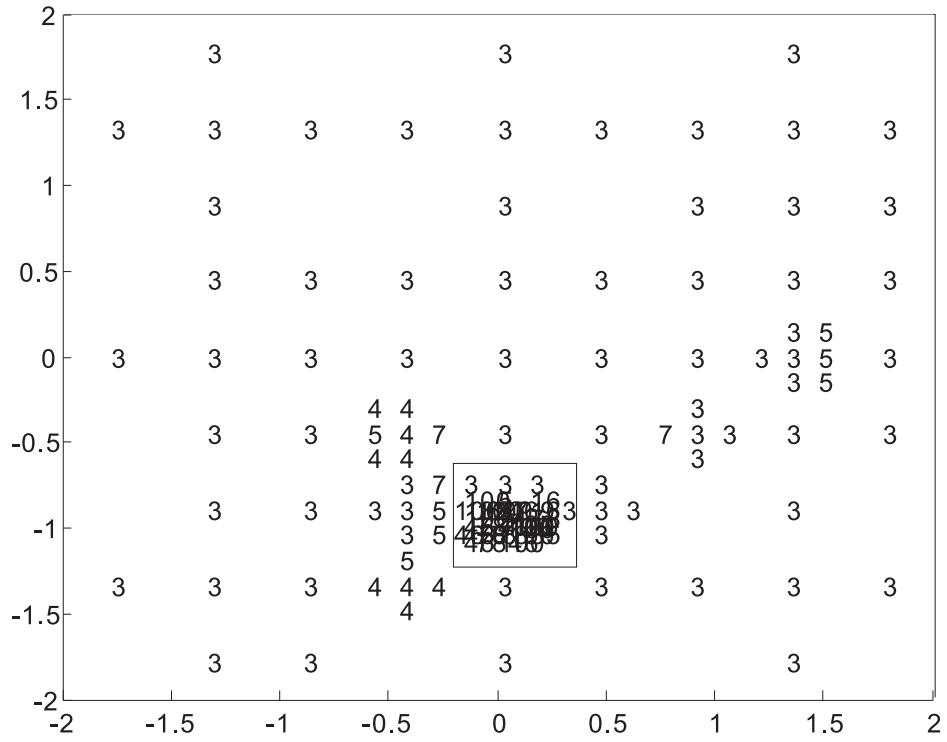


Figure 16: Replication number plot in Goldstein Price function optimization. The center region is magnified for a clear view.

Bayesian approaches. In order to show the advantage of our automatic replication number selection scheme, we used different fixed replications numbers for other algorithms, for example, 1, 5, 10, 50, and 100. We generated 10 runs of each algorithm and computed the averaged difference in terms of the objective value and the position of solution. (y^* and x^* are the optimal objective value and solution to the Goldstein Price function.) Our Noisy DIRECT algorithm performed the best among all the algorithms. As we observed in the table, DIRECT with 1 replication generated worse solutions because the best point is biased by noise, similar to our analysis in Section 2.2.2. DIRECT with 50 fixed replications performed very close to our algorithm, and DIRECT with 100 fixed replications performed slightly worse because of early termination of the algorithm (not enough iterations). The automatic scheme indeed saved computation effort in the earlier iterations of the algorithm compared to the fixed replications approaches (cf., Figure 16). The Snobfit algorithm showed the same issues. The Snobfit algorithm in general needed more function evaluations. As we used 3000 maximum function evaluations, Snobfit with 50 and 100 fixed replications cases terminated very early, thus the solutions were significantly worse. We expect the algorithm would perform better given more function evaluations.

We also tested the algorithm on higher dimensional problems. Although the author claims that the suggested dimension of problem should be less than 30, the test problems in [59] have dimensions ranging from 4-6. Here

Table 3: Comparison of the Noisy DIRECT algorithm and other algorithms for the Goldstein Price function. (Results are based on 10 replications of each algorithm).

	Replication #	Mean $ f(x_{end}) - f^* $	Mean $ x_{end} - x^* $
Noisy DIRECT (normal)	Auto	0.3787	0.0288
Noisy DIRECT (t)	Auto	0.3445	0.0283
DIRECT	1	2.4570	0.0694
	5	1.5045	0.0601
	10	0.6119	0.0432
	50	0.4073	0.0296
	100	0.6474	0.0370
Snobfit	1	2.3231	0.0688
	5	2.0332	0.0802
	10	0.9761	0.0536
	50	11.683	0.1805
	100	44.456	0.5695

we introduce the ‘perm’ function, which has an adjustable dimension n

$$f(x) = \sum_{k=1}^n \left(\sum_{i=1}^n [i^k + \theta] [x_i^k - (1/i)^k] \right)^2.$$

This function has a global solution at $x_i = 1/i, i = 1, 2, \dots, n$, at which the global objective value attains 0. The value θ was set at 0.5, and the range of the region was $[0, 1]^n$. In our experiment, we allowed a total of 100,000 function evaluations. Increasing the dimension n made the problem significantly more difficult to solve. In fact, we were not able to obtain a satisfactory solution for 20 dimensional problems. A single run of a 10 dimensional problem (Table 4) showed the same pattern as in the Goldstein Price case.

Table 4: The performance of the Noisy DIRECT algorithm for the 10-dimensional perm function, with $\sigma^2 = 1$.

Iteration (k)	$f(x_k)$	$\bar{F}(x_k)$	FN
1	372.5	372.6	63
5	45.9	45.3	435
10	5.45	5.84	1299
15	2.3	1.52	2598
20	0.87	0.82	14139
25	0.55	0.28	50131
30	0.30	0.16	107593

A Simulation Problem

We applied the Noisy DIRECT algorithm to solve the ambulance base problem, one of the testbed problems provided in [82]. The problem aims to determine the optimal locations of ambulance bases $p(i), i = 1, 2, \dots, d$, where d is the number of bases. The emergency calls follow a Poisson arrival process at the rate of λ_a , and all calls are initiated in the region $[0, 1]^2$. The location of a call follows a joint triangle distribution function $g(x_1, x_2) = \hat{g}(x_1)\hat{g}(x_2)$, where $\hat{g}(x)$ is a triangle distribution $\hat{g}_{a,b,c}(x)$ (Figure 17).

We assume each ambulance travels at a constant speed v , thus the routing time is twice the distance between the call location and the base divided by the speed v (round trip). The scene time of an ambulance follows an exponential distribution with a location parameter λ_s . The total time of an ambulance at work will be the routing time plus the scene time.

When a call arrives, the nearest free ambulance should respond to the call. If no ambulance is available, the call will be put on wait, and the first ambulance to be freed will respond to this call. The waiting calls are added in FIFO order. In the problem, we aim to find the optimal positions of the ambulance bases such that the total response time is minimized. The response time is consisted of the one-way routing time plus the possible waiting time. We considered two objectives (a) minimize the expected response time, and (b) minimize the maximum response time in a fixed period.

A particular model had the following parameter settings. We considered

a simulation period time 500 hours, and $\lambda_a = 0.5$, thus around 1000 calls occurred in each simulation run. $\hat{g}(x)$ used parameters $a = 0, b = 1, c = 0.8$, thus the distribution of calls centered around the location $(0.8, 0.8)$. The vehicle speed was 0.5 and parameter $\lambda_s = 0.1$. The simulation showed that roughly 10% of calls were put on wait.

We allowed a maximum of 20,000 simulation runs. The Noisy DIRECT algorithm ended up with 25 iterations and the final positions of the ambulance bases are plotted in Figure 18. Since the calls were symmetrically distributed about the diagonal of the unit square, the final positions were also observed approximately symmetric about the diagonal. The solution we obtained shows an averaged response time of an ambulance is 0.3307 hour. For a comparison, we ran an exhaustive computation using a refined model with a simulation period of 25,000 hours, which is equivalent to replicating the previous coarse model 50 times. The best solution we obtained yielded an average response time 0.3291 hour, about a 0.48% improvement over our solution. We note however that this solution while being close to symmetric again, is not simply a refinement to the solution our DIRECT code generates.

2.3 The Phase Transition Module

Phase II methods, such as the extended UOBYQA algorithms, normally require an initial point x_0 and a trust region radius Δ , such that a model is

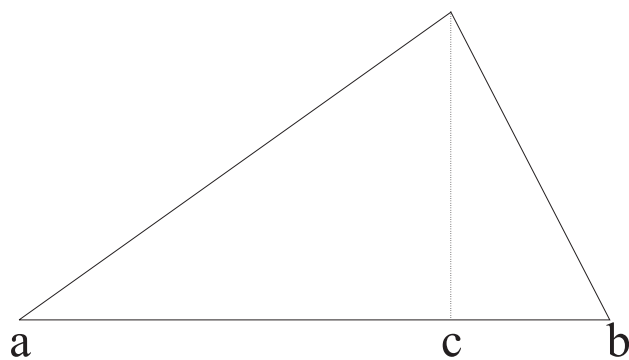


Figure 17: Pdf function of a triangular distribution

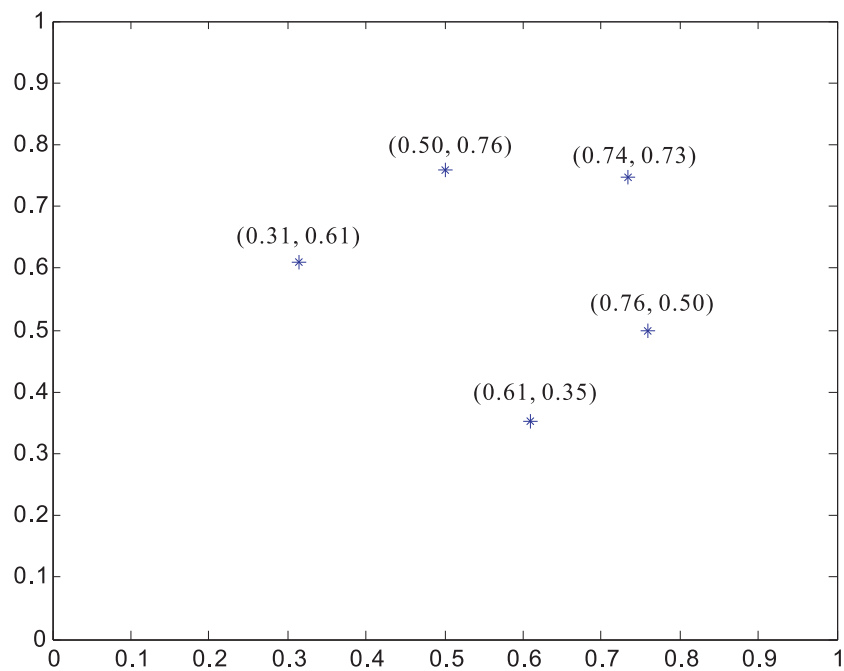


Figure 18: Positions of the ambulance bases

centered at $x_0^{(i)}$ and the trust region is defined as

$$\{x \mid \|x - x_0^{(i)}\|_2 \leq \Delta\}.$$

In Phase II, we perform M multiple optimizations based on a set of trail starting points $\mathcal{I} = \{x_0^{(1)}, x_0^{(2)}, \dots, x_0^{(M)}\}$ (M is also the cardinality of \mathcal{I}). For simplicity in computation, we set a uniform local region size Δ for all $x_0^{(i)}$ and require that the associated subregions should be mutually separated by Δ . When defining a proper subregion size, we assume that the underlying function f should perform as or close to a quadratic function within the subregion.

A non-parametric regression is performed to determine the uniform trust region radius Δ . Being different from regular regression methods, which use the entire samples in the domain to construct one model, local regression predicts the value at a point using a local model based on the samples within a ‘window size’. It is presumed that the samples outside the local region have only slight relevance to the current prediction. ‘Non-parametric’ means that there is no single model in a setting of parameters that can explain the underlying response function. In our procedure, we treat the optimal ‘window size’ as our subregion radius.

Denote by \mathcal{P} the points evaluated in Phase I. To compute the best ‘window size’ for non-parametric local regression, we consider the problem

$$\begin{aligned} \Delta &\in \arg \min_h sse(h) \\ &= \arg \min_h \sum_{y \in \mathcal{P}} (F(y, \xi(\omega)) - Q_h^y(y))^2, \end{aligned} \quad (2.9)$$

where $sse(h)$ is the sum of squares error of knock-one out prediction (in local quadratic regression). Given a window-size h and a point y , the knock-one out predicted value is $Q_h^y(y)$, where

$$Q_h^y(x) = c + g^T(x - y) + \frac{1}{2}(x - y)^T H(x - y)$$

is a regression model constructed using the data point set $\{x \mid \|x - y\| < h\}$. Note that knock-one out prediction means that the point y is excluded from the point set.

We apply the following procedures to generate the initial point set \mathcal{I} of cardinality at most $M = 10$:

Procedure 6. *Phase transition procedure:*

1. Use non-parametric regression method (2.9) to determine the initial trust region radius Δ .
2. Sort the points in \mathcal{P} by their sample response values (values may be biased by noise) and put the best point into the initial point set \mathcal{I} .

3. For each x taken in ascending order of sample response values from the point set \mathcal{P} , compute the shortest distance from the point to the initial point set

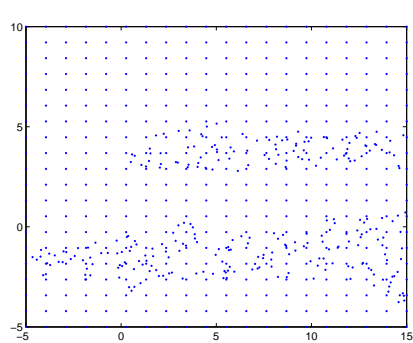
$$dist = \min_{y \in \mathcal{I}} \|y - x\|.$$

If $dist > \Delta$, add the point to the initial point set $\mathcal{I} := \mathcal{I} \cup \{x\}$.

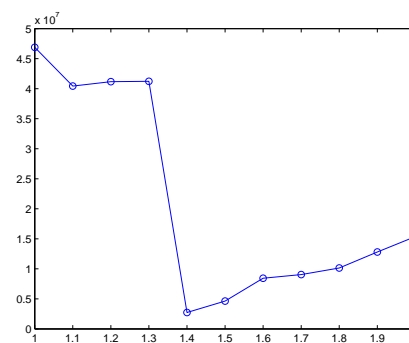
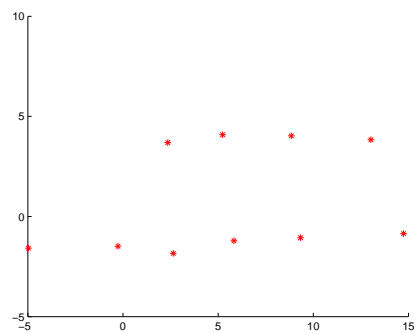
4. Stop if $card(\mathcal{I}) \geq M = 10$ or all the points have been enumerated.

For a numerical example, we continue the discussion in Section 2.1.5. Figure 19 (a) shows all the evaluated samples generated by the classification-based global search. We applied Procedure 6 to yield the initial point set \mathcal{I} for Phase II. First, the non-parametric regression method was executed with different values of h . We used 11 values of h from 1–2, with a 0.1 step length. The corresponding function $sse(h)$ is plotted in Figure 19 (b), showing that $h^* = 1.4$ was the best window size. The uniform region size Δ was then set at 1.4 and applying Procedure 6, we generated a set of 10 points that were mutually separated by Δ . (Note that in other examples the number of initial points may be less than 10.)

We provide an approach for generating a set of initial starting points and the region sizes. Our approach is well adapted to problems in various noisy situations. When the underlying function is flat and smooth and has few subregions of interest, we observe our approach generates a large subregion size (thus a small number of initial points). When the underlying function



(a) The entire available samples

(b) Plot of the function $sse(h)$ (c) Plot of the initial point set \mathcal{I} Figure 19: Produce the initial point set \mathcal{I} .

is bumpy, our approach accordingly generates a small subregion size. The level of noise affects this approach: when noise is significant, it makes local regressions difficult. Since we assume the regions are mutually separated, we obtain sparsely distributed starting points. One disadvantage is that we use the same region size for every subregion, which might not be valid in practice. Further research should be done to investigate how to determine variable region sizes.

Chapter 3

Phase II Methods

In Phase II we apply local optimization algorithms to solve the exact optimum in each promising region that is identified in Phase I. Powell's UOBYQA algorithm is modified to handle the local optimization task. Uncertainty in the objective function becomes the major concern. Our modifications apply Bayesian techniques to guide appropriate sample strategies to estimate the objective function and thus determine the proper number of replications to use. We aim to make the underlying UOBYQA algorithm proceed efficiently while simultaneously controlling the amount of computational effort.

According to the implementation of CRN (common random numbers) in the simulation system, the modified algorithms are the VNSP-UOBYQA algorithm for the CRN case, and the Noisy UOBYQA algorithm for the white noise case. Sections are arranged as follows. In Section 3.1 we introduce the deterministic UOBYQA algorithm, together with the core part on constructing quadratic models. In Section 3.2 we describe the algorithm with the VNSP extension and focus on the sufficient reduction criterion in the trust region method that potentially guarantees the convergence proof. In

Section 3.3, we discuss the Noisy UOBYQA algorithm, which provides an alternative approach especially designed for the white noise case.

3.1 The Deterministic UOBYQA Algorithm

We apply Powell’s UOBYQA (*Unconstrained Optimization BY Quadratic Approximation*) algorithm [85] as our base Phase II local optimization solver. The algorithm is a derivative-free approach and thus is a good fit for the simulation-based optimization problem (1.1). It is designed to solve nonlinear problems with a moderate number of dimensions. The general structure of UOBYQA follows a *model-based approach* [20, 21], which constructs a chain of local quadratic models that approximate the objective function (Figure 6). The method is an iterative algorithm in a trust region framework [80], but it differs from the classical trust region method in that it creates quadratic models by interpolating a set of sample points instead of using the gradient and Hessian values of the objective function (thus making it a derivative-free tool). Besides UOBYQA, other model-based software include WEDGE [75] and NEWUOA [86].

3.1.1 The Core Algorithm

A general framework for the model-based approach is given by Conn and Toint [21], and convergence analysis is presented in [20]. We present an algorithm outline based on the general model-based approach, omitting specific details of UOBYQA. Interested readers may refer to Powell's paper [85] for further details.

Starting the algorithm requires an initial trial point x_0 and an initial trust region radius Δ_0 . As in a classical trust region method, a new promising point is determined from a subproblem:

$$\min_{s \in \mathbb{R}^n} Q_k(x_k + s), \quad \text{subject to } \|s\| \leq \Delta_k. \quad (3.1)$$

The new solution s^* is accepted (or not) by evaluating the ‘degree of agreement’ between f and Q_k :

$$\rho_k = \frac{f(x_k) - f(x_k + s^*)}{Q_k(x_k) - Q_k(x_k + s^*)}. \quad (3.2)$$

If the ratio ρ_k is large enough, which indicates a good agreement between the quadratic model Q_k and the function f , the point $x_k + s^*$ is accepted into the set \mathcal{I}_k .

It may happen that the quadratic model becomes inadequate after a potential step. Accordingly, UOBYQA first checks and improves the adequacy of \mathcal{I}_k before the trust region radius is updated following standard trust region rules. Whenever a new point x^+ enters (the point x^+ may be the solution

point $x_k + s^*$ or a replacement point to improve the geometry), the agreement is rechecked to determine the next iterate.

The notion of adequacy of the interpolation points in a ball

$$\mathcal{B}_k(d) := \{x \in \mathbb{R}^n \mid \|x - x_k\| \leq d\}$$

is defined in [20]. Implicitly, adequacy relates to good conditioning of an underlying matrix, which enables the interpolation model to work well. For example, if the interpolation points lie on a line, then it is difficult to construct the quadratic model (off the line) based on these points. Improving the adequacy of the point set involves replacing a subset of points with new ones. The paper [20] shows a mechanism that will generate adequate interpolation points after a finite number of operations. UOBYQA applies a heuristic procedure, which may not guarantee these properties, but is very effective in practice. Since this point is unrelated to the issues we address here, we state the theory in terms of adequacy to be rigorous, but use the UOBYQA scheme for our practical implementation.

We now present an outline of the UOBYQA algorithm. The constants associated with the trust region update are:

$$0 < \eta_0 \leq \eta_1 < 1, 0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2, \epsilon_1 > 0 \text{ and } \epsilon_2 \geq 1.$$

Procedure 7. *The UOBYQA Algorithm:*

Choose a starting point x_0 , an initial trust region radius Δ_0 and a termination trust region radius Δ_{end} .

1. Generate initial trial points in the interpolation set \mathcal{I}_k . Determine the first iterate $x_1 \in \mathcal{I}_k$ as the best point in \mathcal{I}_k .

2. For iterations $k = 1, 2, \dots$

(a) Construct a quadratic model Q_k of the form (3.5) which interpolates points in \mathcal{I}_k . If $\|g_k\| \leq \epsilon_1$ and \mathcal{I}_k is inadequate in $\mathcal{B}_k(\epsilon_2\|g_k\|)$ (see Section 3.1.2), then improve the quality of \mathcal{I}_k .

(b) Solve the trust region subproblem (3.1). Evaluate f at the new point $x_k + s^*$ and compute the agreement ratio ρ_k in (3.2).

(c) If $\rho_k \geq \eta_1$, then insert $x_k + s^*$ into \mathcal{I}_k . If a point is added to the set \mathcal{I}_k , another element in \mathcal{I}_k should be removed to maintain the cardinality $|\mathcal{I}_k| = L$. If $\rho_k < \eta_1$ and \mathcal{I}_k is inadequate in \mathcal{B}_k , improve the quality of \mathcal{I}_k .

(d) Update the trust region radius Δ_k :

$$\Delta_{k+1} \begin{cases} \in [\Delta_k, \gamma_2\Delta_k], & \text{if } \rho_k \geq \eta_1; \\ \in [\gamma_0\Delta_k, \gamma_1\Delta_k], & \text{if } \rho_k < \eta_1 \text{ and } \mathcal{I}_k \text{ is adequate in } \mathcal{B}_k(\Delta_k); \\ = \Delta_k, & \text{otherwise.} \end{cases} \quad (3.3)$$

(e) When a new point x^+ is added into \mathcal{I}_k , if

$$\hat{\rho}_k = \frac{f(x_k) - f(x^+)}{Q_k(x_k) - Q_k(x_k + s^*)} \geq \eta_0, \quad (3.4)$$

then $x_{k+1} = x^+$, otherwise, $x_{k+1} = x_k$.

(f) Check whether any of the termination criteria is satisfied, otherwise repeat the loop. The termination criteria include $\Delta_k \leq \Delta_{end}$ and hitting the maximum limit of function evaluations.

3. Evaluate and return the final solution point.

Note that in the algorithm a *successful* iteration is claimed only if the new iterate x_{k+1} satisfies the condition

$$\hat{\rho}_k \geq \eta_0,$$

otherwise, the iteration is called *unsuccessful*.

3.1.2 Interpolating Quadratic Model Properties

At every iteration of the algorithm, a quadratic model

$$Q_k(x) = c_k + (g_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T G_k (x - x_k), \quad (3.5)$$

is constructed by interpolating a set of adequate points (see explanation below) $\mathcal{I}_k = \{y^1, y^2, \dots, y^L\}$,

$$Q_k(y^i) = f(y^i), \quad i = 1, 2, \dots, L. \quad (3.6)$$

The point x_k acts as the center of a trust region, the coefficient c_k is a scalar, g_k is a vector in \mathbb{R}^n , and G_k is an $n \times n$ real symmetric matrix. The interpolation model is expected to approximate f well around the base point x_k , such that the parameters c_k, g_k and G_k approximate the Taylor

series expansion coefficients of f around x_k . Thus, g_k is used as a derivative estimate for f . To ensure a unique quadratic interpolator, the number of interpolating points should satisfy

$$L = \frac{1}{2}(n+1)(n+2). \quad (3.7)$$

Note that the model construction step (3.5) does not require evaluations of the gradient or the Hessian.

3.2 The VNSP-UOBYQA Algorithm

The sample-path method is one of the most important tools in simulation-based optimization. The basic idea of the method is to approximate the expected simulation output by the average of sample observations with a common random number sequence. Instead of directly solving problem (1.1), we consider an approximate problem

$$\min_{x \in \mathbb{R}^n} \hat{f}^N(x) := \frac{1}{N} \sum_{i=1}^N F(x, \xi_i), \quad (3.8)$$

Our purpose of this section is to introduce a *Variable-Number Sample-Path* (VNSP) scheme, an extension of sample-path optimization. The classical sample-path method is criticized for its excessive simulation evaluations: in order to obtain a solution point $x^{*,N}$, one has to solve an individual optimization problem (3.8) and at each iterate x_k of the algorithm $\hat{f}^N(x_k)$ is required (with N large). The new VNSP scheme is designed to generate different numbers of samples (N) at each iteration. Denoting N_k as the number

of samples at iteration k , the VNSP scheme integrates Bayesian techniques to determine a satisfactory N_k , which accordingly ensures the accuracy of the approximation of $\hat{f}^N(x)$ to $f(x)$. The numbers $\{N_k\}$ form a non-decreasing sequence within the algorithm, with possible convergence to infinity. The new approach is briefly described in Figure 20. Starting from x_0 , the algorithm generates its iterates across different averaged sample functions. In an intermediate iteration k , it first computes a satisfactory N_k which guarantees certain level of accuracy, then an optimization step is taken exactly the same as in problem (3.8), with $N = N_k$. The algorithm has a globally convergent solution x^{*,N_∞} , where $N_\infty := \lim_{k \rightarrow \infty} N_k$. The convergence is almost sure for all the sample paths, which correspond to different runs of the algorithm. The solution, we will prove later, matches the solution $x^{*,\infty}$. Significant computational savings accrue when k is small.

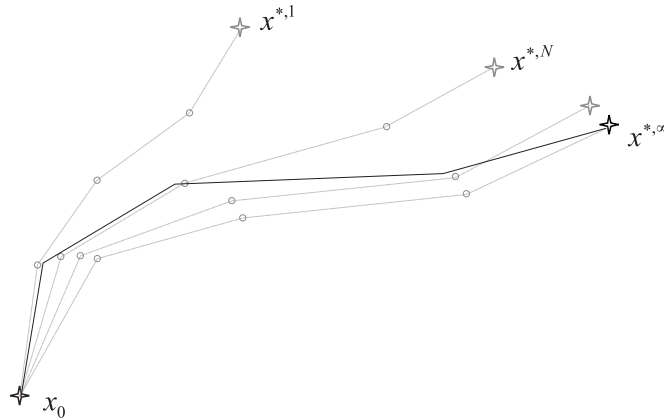


Figure 20: Mechanism of the new sample-path method with the VNSP scheme.

Another ‘variable-sample’ scheme for sample-path optimization is proposed by Homem-de-Mello in [52]. The work proposes a framework for iterative algorithms that use, at iteration k , an estimator f^{N_k} of the true function f constructed via the sample average of N_k samples. It is shown in [52] that, if the convergence of such an algorithm requires that $f^{N_k}(x) \rightarrow f(x)$ almost surely for all sample paths, then it is necessary that $N_k \rightarrow \infty$ at a certain rate. Our VNSP scheme is significantly different: N_k in our scheme is validated based on the uncertainty of the iterate x_k . We require $x_k \rightarrow x^*$ almost surely, but we do not impose the convergence condition $\hat{f}^{N_k} \rightarrow f$. As a consequence, $\{N_k\}$ is a non-decreasing sequence with the limit value N_∞ being either finite or infinite. Here is a toy example showing that the limit sample number N_∞ in our algorithm can be finite. Consider a simulation system with only additive noise:

$$F(x, \xi(\omega)) = \phi(x) + \xi(\omega),$$

where $\phi(x)$ is a deterministic function and $\xi(\omega) \sim N(0, \sigma^2)$. As a result, the minimizer of each piece $F(x, \xi_i) = \phi(x) + \xi_i$ coincides with the minimizer of $f(x) = \phi(x)$ (thus the solutions of \hat{f}^k are: $x^{*,1} = x^{*,2} = \dots = x^{*,\infty}$). In this case, our VNSP scheme turns out to use a constant sequence of sample numbers $N_k : N_1 = N_2 = \dots = N_\infty < +\infty$. We obtain $\lim_{k \rightarrow \infty} x_k = x^{*,N_1} = \dots = x^{*,N_\infty} = x^*$, but obviously $\lim_{k \rightarrow \infty} \hat{f}^{N_k} \neq f$. However, the ‘variable-sample’ scheme in [52] still requires $\lim_{k \rightarrow \infty} N_k = \infty$ on this example. More details about this toy example can be found in the numerical example section.

3.2.1 The Bayesian VNSP Scheme

The goal of the VNSP scheme is to determine the suitable sample number N_k to be applied at iteration k . Such a step should be inserted before constructing the quadratic model Q_k . See the UOBYQA Algorithm - Procedure 7. As a consequence, the algorithm, performing on averaged sample function \hat{f}^{N_k} , produces solutions x_k that converge to $x^{*,N_\infty} = x^{*,\infty}$ (see Figure 21).

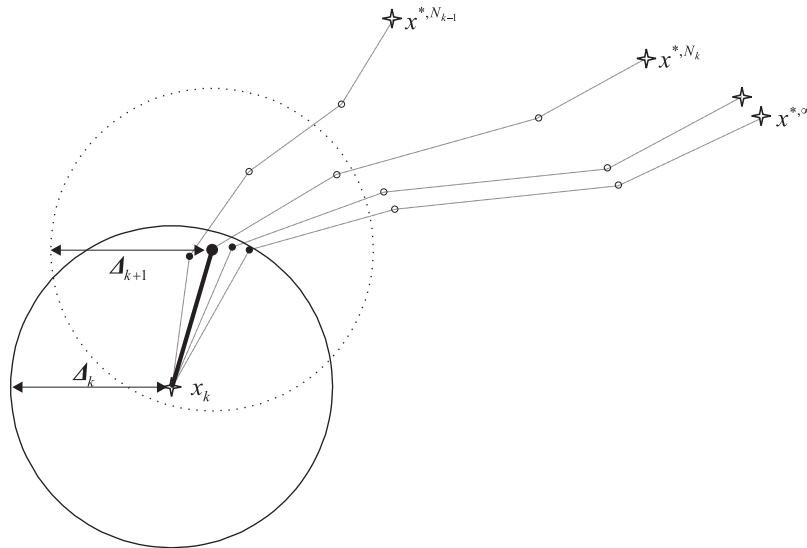


Figure 21: Choose the correct N_k and move the next iterate along the averaged sample function \hat{f}^{N_k} .

Revisiting the Quadratic Models

When we consider the sample response function $F(x, \xi(\omega))$, we inherit several basic assumptions regarding the nature of the objective function from [20]. We will also discover properties associated with the quadratic model $Q_k^N(x)$

that interpolates the function \hat{f}^N .

Assumption 1. For a fixed $y \in \mathbb{R}^d$ the function $F(\cdot, y)$ is twice continuously differentiable and its gradient and Hessian are uniformly bounded on $\mathbb{R}^n \times \mathbb{R}^d$. There exist constants $\kappa_{Fg} > 0$ and $\kappa_{Fh} > 0$, such that the following inequalities hold:

$$\sup_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \left\| \frac{\partial F(x, y)}{\partial x} \right\| \leq \kappa_{Fg} \text{ and } \sup_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \left\| \frac{\partial^2 F(x, y)}{\partial^2 x} \right\| \leq \kappa_{Fh}.$$

Assumption 2. For a given $y \in \mathbb{R}^d$, the function $F(\cdot, y)$ and the underlying function $f(\cdot)$ are bounded below on \mathbb{R}^n .

Let Q_k^N be the quadratic model interpolating the sample average function \hat{f}^N at the point set \mathcal{I}_k . We assume that its Hessian is bounded uniformly.

Assumption 3. The Hessian of the quadratic function Q_k^N is uniformly bounded for all x in the trust region, i.e., there exists a constant $\kappa_{Qh} > 0$ such that

$$\|G_k^N\| \leq \kappa_{Qh}, \text{ for all } x \in \{x \in \mathbb{R}^n \mid \|x - x_k\| \leq \Delta_k\}.$$

As a key component of the analysis, Conn, Scheinberg, and Toint address the difference of using the classical Taylor expansion model

$$\hat{Q}_k^N(x) = \hat{f}^N(x_k) + \nabla \hat{f}^N(x_k)^T (x - x_k) + \frac{1}{2} (x - x_k)^T \nabla^2 \hat{f}^N(x_k) (x - x_k)$$

and the interpolative quadratic model Q_k^N . The model \hat{Q}_k^N shares the same gradient $\nabla \hat{f}^N(x_k)$ at x_k with the underlying function, while for the interpolative model Q_k^N , its gradient g_k^N is merely an approximation. The error in

this approximation is shown in the following lemma to decrease quadratically with the trust region radius. As an implication of the lemma, within a small trust region, the model Q_k^N is also a decent approximation model.

Lemma 3.1. *(Theorem 4 in [20]) Suppose Assumptions 1-3 hold and \mathcal{I}_k is adequate in the trust region $\mathcal{B}_k(\Delta_k)$. Furthermore, if at iteration k , Q_k^N is the interpolative approximation model for the function \hat{f}^N , then assume the bias of the function value and the gradient are bounded within the trust region. Then there exist constants κ_{em} and κ_{eg} , for each $x \in \mathcal{B}_k(\Delta_k)$, the following inequalities hold*

$$|\hat{f}^N(x) - Q_k^N(x)| \leq \kappa_{em} \max[\Delta_k^2, \Delta_k^3] \quad (3.9)$$

and

$$\|\nabla \hat{f}^N(x) - g_k^N\| \leq \kappa_{eg} \max[\Delta_k, \Delta_k^2]. \quad (3.10)$$

In fact, the proof of Lemma 3.1 is associated with manipulating Newton polynomials instead of the Lagrange functions that UOBYQA uses. Since the quadratic model is unique via interpolation (by choice of L), the results are valid regardless of how the model is constructed.

We have seen that Q_k^N interpolates the function \hat{f}^N at the points in \mathcal{I}_k . Let Q_k^∞ be the ‘expected’ quadratic model interpolating the function f at the same points. The following lemma provides convergence of Q_k^N to Q_k^∞ .

Lemma 3.2. *$Q_k^N(x)$ converges pointwise to $Q_k^\infty(x)$ with probability 1 (w.p.1) as $N \rightarrow \infty$.*

Proof. The *Law of Large Numbers* (LLN) guarantees the pointwise convergence of $\hat{f}^N(x)$ to $f(x)$ w.p.1 [91]. By solving the system of linear equations (3.6), each component of the coefficients of Q_k^N , $c_k^N, g_k^N(i), G_k^N(i, j), i, j = 1, 2, \dots, n$, is uniquely expressed as a linear combination of $\hat{f}^N(y^i), \hat{f}^N(y^i)\hat{f}^N(y^j), i, j = 1, 2, \dots, L$. (The uniqueness of solution requires the adequacy of the interpolation points.) Therefore, as $N \rightarrow \infty$ the coefficients c_k^N, g_k^N, G_k^N converge to $c_k^\infty, g_k^\infty, G_k^\infty$ w.p.1 because the values $\hat{f}^N(y^i)$ converge to $f(y^i), i = 1, 2, \dots, L$, w.p.1. Finally, for a fixed value $x \in \mathbb{R}^n$, $Q_k^N(x)$ converges to $Q_k^\infty(x)$ w.p.1. □ □

Posterior Estimations for the Quadratic Model

Q_k^∞ (unknown) is of most interest to us. We derive the posterior distributions of the coefficients of Q_k^∞ as follows. Assume the simulation output at points of \mathcal{I}_k

$$\mathbf{F} = (F(y^1, \xi(\omega)), F(y^2, \xi(\omega)), \dots, F(y^L, \xi(\omega)))$$

has mean $\boldsymbol{\mu} = (\mu(y^1), \mu(y^2), \dots, \mu(y^L))$ and covariance matrix $\boldsymbol{\Sigma}$. According to the Bayesian analysis in Section 1.2, we have

$$\boldsymbol{\mu} | X^N \sim N(\bar{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}/N), \tag{3.11}$$

where $\bar{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are the sample mean and sample covariance respectively. X^N is used instead of X to indicate the number of replications N used.

We delve into the detailed steps of quadratic model construction in the

UOBYQA algorithm. The quadratic model Q_k^∞ is expressed as a linear combination of Lagrange functions $l_j(x)$,

$$Q_k^\infty(x) = \sum_{j=1}^L f(y^j)l_j(x) = \sum_{j=1}^L \mu(y^j)l_j(x), \quad x \in \mathbb{R}^n. \quad (3.12)$$

Each piece of $l_j(x)$ is a quadratic polynomial from \mathbb{R}^n to \mathbb{R}

$$l_j(x_k + s) = c_j + g_j^T s + \frac{1}{2}s^T G_j s, \quad j = 1, 2, \dots, L,$$

that has the property

$$l_j(y^i) = \delta_{ij}, \quad i = 1, 2, \dots, L,$$

where δ_{ij} is 1 if $i = j$ and 0 otherwise. It follows from (3.5) and (3.12) that the parameters of Q_k^∞ are derived as

$$\begin{aligned} c_k^\infty &= \mathbf{c}\boldsymbol{\mu}^T, \quad g_k^\infty = \mathbf{g}\boldsymbol{\mu}^T, \\ \text{and } G_k^\infty &= \sum_{j=1}^L \mu(y^j)G_j, \end{aligned} \quad (3.13)$$

where $\mathbf{c} = (c_1, \dots, c_L)$ and $\mathbf{g} = (g_1, \dots, g_L)$. Note that the parameters c_j , g_j , and G_j in each Lagrange function l_j are uniquely determined when the points y^j are given, regardless of the function f .

Combining (3.11) and (3.13), it follows that the posterior distributions of c_k^∞ , g_k^∞ and G_k^∞ are normal-like distributions:

$$c_k^\infty | X^N \sim N(\mathbf{c}\bar{\boldsymbol{\mu}}^T, \mathbf{c}\hat{\boldsymbol{\Sigma}}\mathbf{c}^T/N), \quad (3.14)$$

$$g_k^\infty | X^N \sim N(\mathbf{g}\bar{\boldsymbol{\mu}}^T, \mathbf{g}\hat{\boldsymbol{\Sigma}}\mathbf{g}^T/N), \quad (3.15)$$

$$G_k^\infty | X^N \sim MN\left(\sum_{j=1}^L \bar{\mu}(y^j)G_j, \mathbf{P}^T\hat{\boldsymbol{\Sigma}}\mathbf{P}/N, \mathbf{P}^T\hat{\boldsymbol{\Sigma}}\mathbf{P}/N\right), \quad (3.16)$$

where the $L \times N$ matrix $\mathbf{P} = (G_1\mathbf{1}, \dots, G_L\mathbf{1})^T$. The matrix normal distribution $MN(\boldsymbol{\mu}, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2)$ has parameters mean $\boldsymbol{\mu}$, left variance $\boldsymbol{\nu}_1$, and right variance $\boldsymbol{\nu}_2$ [24]. In (3.16), because G_j are symmetric, the left variance and right variance coincide.

The Bayesian VNSP Scheme

We will develop and summarize the Bayesian VNSP scheme in this section. We first introduce the following lemma concerning the ‘sufficient reduction’ within a trust region step. This is an important but standard result in the trust region literature.

Lemma 3.3. *The solution $s_k^{*,N}$ of the subproblem (3.1) satisfies*

$$Q_k^N(x_k) - Q_k^N(x_k + s_k^{*,N}) \geq \kappa_{mdc} \|g_k^N\| \min \left[\frac{\|g_k^N\|}{\kappa_{Qh}}, \Delta_k \right] \quad (3.17)$$

for some constant $\kappa_{mdc} \in (0, \frac{1}{2})$ independent of k .

Proof. For the *Cauchy point* $x_k + s_c^N$ defined as the minimizer of the model in the trust region along the steepest decent direction, we have a corresponding reduction [78]

$$Q_k^N(x_k) - Q_k^N(x_k + s_c^N) \geq \frac{1}{2} \|g_k^N\| \min \left[\frac{\|g_k^N\|}{\kappa_{Qh}}, \Delta_k \right]. \quad (3.18)$$

Since the solution $s_k^{*,N}$ of the subproblem yields an even lower objective value of Q_k^N , we have the inequality (3.17). The complete proof can be found in [80]. □ □

Comment 1: Lemma 3.3 is generally true for models Q_k^N and Q_k^∞ .

Comment 2: There are issues concerning setting the values of κ_{mdc} and κ_{Qh} in an implementation. For κ_{mdc} , we use a safeguard value of 0.49, which is slightly smaller than $\frac{1}{2}$. This value is true for Cauchy points, so is valid for the solutions of the subproblem. For κ_{Qh} , we update it as the algorithm proceeds

$$\kappa_{Qh} := \max(\kappa_{Qh}, \|G_k^N\|), \quad (3.19)$$

that is, κ_{Qh} is updated whenever a new G_k^N is generated. Assumption 3 ensures the boundedness of the sampled Hessian and prevents the occurrence of ill-conditioned problems. It is hard to find a good value of κ_{Qh} satisfying Assumption 3, but in practice the above scheme updates the value very infrequently.

In our algorithm, $Q_k^N(x_k) - Q_k^N(x_k + s^{*,N})$ is the observed model reduction, which serves to promote the next iterate (i.e., used to compute the agreement ρ_k^N in (3.2)). The key idea for the global convergence of algorithm is that, by replacing g_k^N with g_k^∞ in (3.17), we force the model reduction $Q_k^N(x_k) - Q_k^N(x_k + s^{*,N})$ to regulate the size of $\|g_k^\infty\|$, and so drive $\|g_k^\infty\|$ to zero. We present the modified ‘sufficient reduction’ criterion:

$$Q_k^N(x_k) - Q_k^N(x_k + s^{*,N}) \geq \kappa_{mdc} \|g_k^\infty\| \min \left[\frac{\|g_k^\infty\|}{\kappa_{Qh}}, \Delta_k \right]. \quad (3.20)$$

Lemma 3.2 and 3.3 imply that increasing the replication number N lessens the bias between the quadratic models Q_k^N and Q_k^∞ , and is likely to produce

a more precise step length $s^{*,N}$, close to $s^{*,\infty}$. The criterion will be eventually satisfied when $N \rightarrow \infty$.

To ensure the ‘sufficient reduction’ criterion (3.20) is satisfied accurately, we require

$$\begin{aligned} Pr(E_k^N) &= Pr\left(Q_k^N(x_k) - Q_k^N(x_k + s^{*,N}) < \kappa_{mdc} \|g_k^\infty\| \min\left[\frac{\|g_k^\infty\|}{\kappa_{Qh}}, \Delta_k\right]\right) \\ &\leq \alpha_k, \end{aligned} \quad (3.21)$$

where the event E_k^N is defined as the failure of (3.20) for the current N and α_k is the significance level. In practice, the risk $Pr(E_k^N)$ is difficult to evaluate because 1) it requires multiple sample paths, while the available data is limited to one sample path, and 2) we do not know the explicit form of Q_k^∞ (and hence g_k^∞).

By adapting knowledge from Bayesian inference, we approximate the risk value by a Bayesian posterior estimation based on the current observations X^N

$$Pr(E_k^N) \approx Pr(E_k^N | X^N). \quad (3.22)$$

The value $Pr(E_k^N | X^N)$ is thus called *Bayes risk*, which depends on a particular sample path. In the Bayesian perspective, the unknown quantities, such as $f(x)$ and g_k^∞ , are considered as random variables, whose posterior distributions are inferred by Bayes’ rule. Given the observations X^N , we

have

$$\begin{aligned}
& Pr(E_k^N | X^N) \tag{3.23} \\
&= Pr \left(Q_k^N(x_k) - Q_k^N(x_k + s^{*,N}) < \kappa_{mdc} \|g_k^\infty\| \min \left[\frac{\|g_k^\infty\|}{\kappa_{Qh}}, \Delta_k \right] \middle| X^N \right) \\
&= Pr \left(Q_k^N(x_k) - Q_k^N(x_k + s^{*,N}) < \kappa_{mdc} \|g_k^\infty | X^N\| \min \left[\frac{\|g_k^\infty | X^N\|}{\kappa_{Qh}}, \Delta_k \right] \right).
\end{aligned}$$

The left hand side $Q_k^N(x_k) - Q_k^N(x_k + s^{*,N})$ of the inequality becomes a fixed quantity given X^N . The probability evaluation is computed with respect to the posterior distribution $g_k^\infty | X^N$. Here we show the fact:

Lemma 3.4. *The Bayes risk $Pr(E_k^N | X^N)$ converges to zero as $N \rightarrow \infty$.*

Proof. For simplicity in notation, let $A^N = \|g_k^\infty | X^N\| \min \left[\frac{\|g_k^\infty | X^N\|}{\kappa_{Qh}}, \Delta_k \right]$ be a sequence of random variables, and $b^N = Q_k^N(x_k) - Q_k^N(x_k + s^{*,N})$ be a sequence of scalars. As shown in (3.15), as $N \rightarrow \infty$ the distribution $g^\infty | X^N$ converges to a delta distribution. A^N also converges to a delta distribution A^∞ centered at $\|g_k^\infty\| \min \left[\frac{\|g_k^\infty\|}{\kappa_{Qh}}, \Delta_k \right]$. Therefore, A^∞ is essentially a constant with zero variance. We can rewrite the Bayes risk in (3.23) as follows:

$$\begin{aligned}
& Pr(E_k^N | X^N) \\
&= Pr(b^N < \kappa_{mdc} A^N) \\
&= Pr \left((b^N - b^\infty) + \left(b^\infty - \frac{1}{2} A^\infty \right) + \right. \\
&\quad \left. \left(\frac{1}{2} A^\infty - \kappa_{mdc} A^\infty \right) < \kappa_{mdc} (A^N - A^\infty) \right) \\
&= Pr \left(A^N - A^\infty > \frac{(b^N - b^\infty) + (b^\infty - \frac{1}{2} A^\infty) + (\frac{1}{2} A^\infty - \kappa_{mdc} A^\infty)}{\kappa_{mdc}} \right).
\end{aligned}$$

As $N \rightarrow \infty$, $b^N - b^\infty$ converges to zero, $b^\infty - \frac{1}{2}A^\infty \geq 0$ by Lemma 3.3, and $\frac{1}{2}A^\infty - \kappa_{mdc}A^\infty$ converges to a strictly positive value because $\kappa_{mdc} < \frac{1}{2}$. Thus the right hand side of the inequality converges to a strictly positive value. Showing the Bayes risk converges to zero is equivalent to showing the random variable A^N converges to A^∞ in probability.

If we let $a^N = \mathbb{E}[A^N]$, then $a^N \rightarrow \mathbb{E}[A^\infty] = A^\infty$ (Theorem (3.8) p17 [30]). For a given positive value $\varepsilon > 0$, there exists a large enough N' such that when $N > N'$ we have $|a^N - A^\infty| \leq \varepsilon/2$. If $N > N'$,

$$\begin{aligned}
Pr(A^N - A^\infty > \varepsilon) &\leq Pr(|A^N - A^\infty| > \varepsilon) \\
&= Pr(|A^N - a^N + a^N - A^\infty| > \varepsilon) \\
&\leq Pr(|A^N - a^N| + |a^N - A^\infty| > \varepsilon) \\
&\leq Pr(|A^N - a^N| > \varepsilon/2) \\
&\leq (2/\varepsilon)^2 var(A^N).
\end{aligned}$$

The last inequality is by the Chebyshev's inequality [30]. Because $var(A^N)$ decreases to zero, we have $Pr(A^N - A^\infty > \varepsilon)$ decreases to zero and A^N converges to A^∞ in probability. The proof of the lemma follows. \square \square

Lemma 3.4 guarantees that $Pr(E_k^N | X^N) \leq \alpha_k$ will eventually be satisfied when N is large enough.

In Section 3.2.1, we derive the posterior distributions for the parameters of Q_k^∞ . These distributions can be plugged in (3.23) to evaluate the Bayes risk.

However, the exact evaluation of the probability is hard to compute, especially involving the component $\kappa_{mdc} \|g_k^\infty | X^N\| \min \left[\frac{\|g_k^\infty | X^N\|}{\kappa_{Qh}}, \Delta_k \right]$. Instead we use the Monte Carlo method to approximate the probability value: we generate M random samples from the posterior distribution of $g_k^\infty | X^N$. Based on the samples, we check the event of ‘sufficient reduction’ and make a count on the failed cases: M_{fail} . The probability value in (3.23) is then approximated by

$$Pr(E_k^N | X^N) \approx \frac{M_{fail}}{M}. \quad (3.24)$$

The approximation becomes accurate as M increases. Normally, we use a large value $M = 500$. Note that this does not require any new evaluations of the sample response function, but instead samples from the inferred Bayesian distribution $g_k^\infty | X^N$. We actually enforce a stricter accuracy on the fraction value for reasons that will be described below:

$$\frac{M_{fail}}{M} \leq \frac{\alpha_k}{2}. \quad (3.25)$$

A complete description of our Bayesian VNSP scheme follows:

Procedure 8. *The VNSP scheme*

At the k th iteration of the algorithm, start with $N = N_{k-1}$.

Loop

1. *Evaluate N replications at each point y^j in the interpolation set \mathcal{I}_k , to construct the data matrix X^N . Note: data from previous iterations can be included.*

2. Construct the quadratic model Q_k^N and solve the subproblem for $x_k + s^{*,N}$.
3. Update the value of κ_{Qh} by (3.19).
4. Compute the Bayesian posterior distributions for the parameters of Q_k^∞ as described above.
5. Validate the Monte Carlo estimate (3.25). If the criterion is satisfied, then stop with $N_k = N$; otherwise increase N , and repeat the loop.

Since a smaller N_k is preferable, a practical approach is to sequentially allocate computing resources: starting with $N = N_{k-1}$, we decide to increase N or keep N by checking (3.25). If rejected, N is updated as

$$N := N \cdot \beta,$$

where β is an incremental factor. Otherwise, the current N is used as the sample number N_k at iteration k .

Two approximation steps (3.22) and (3.24) are employed in the computation. The following assumptions formally guarantee that risk $Pr(E_k^N)$ is eventually approximated by the Monte Carlo fraction value M_{fail}/M .

Assumption 4. *The difference between the risk $Pr(E_k^N)$ and the Monte Carlo estimation value is bounded by $\frac{\alpha_k}{2}$*

$$\left| Pr(E_k^N) - \frac{M_{fail}}{M} \right| \leq \frac{\alpha_k}{2}.$$

When $M \rightarrow \infty$, $\frac{M_{fail}}{M}$ approaches the Bayes risk $Pr(E_k^N|X^N)$. The assumption essentially guarantees the Bayes risk $Pr(E_k^N|X^N)$ is a good approximation of the real risk $Pr(E_k^N)$. Under this assumption and the criterion (3.25), it implies

$$|Pr(E_k^N)| \leq \left| Pr(E_k^N) - \frac{M_{fail}}{M} \right| + \left| \frac{M_{fail}}{M} \right| \leq \frac{\alpha_k}{2} + \frac{\alpha_k}{2} = \alpha_k,$$

which guarantees the accuracy of the ‘sufficient reduction’ criterion (3.21). The algorithm enforces (3.25) and the convergence proof can thus use the criterion (3.21).

Assumption 5. *The sequence of significance level values $\{\alpha_k\}$ satisfy the property:*

$$\sum_{k=1}^{\infty} \alpha_k < \infty. \quad (3.26)$$

The assumption necessitates a stricter accuracy to be satisfied as the algorithm proceeds, which allows the use of the *Borel-Cantelli Lemma* in probability theory.

Lemma 3.5 ((1st) Borel-Cantelli Lemma). *Let $\{E_k^N\}$ be a sequence of events, and the sum of the probabilities of E_k^N is finite, then the probability of infinitely many E_k^N occur is 0.*

Proof. See the book by Durrett [30]. □ □

Consider the event E_k^N to be the failure to satisfy the ‘sufficient reduction’ criterion (3.20). Given the error rate (3.21) and Assumption 5, the Borel-Cantelli Lemma provides that the events E_k^N only happen finitely many times

w.p.1. Therefore, if we define K as the first successful index after all failed instances, then (3.20) is satisfied w.p.1 for all iterations $k \geq K$. We will use this without reference in the sequel.

Finally, we will require the following uniformity assumptions to be valid in the convergence proof.

Assumption 6. *Given two points $x_1, x_2 \in \mathbb{R}^n$, the sample response difference of the two points is $F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))$. We assume that the 2nd and 4th central moments of the sample response difference are uniformly bounded. For simplicity, we denote the i th central moment of a random variable Z by $\varphi_i(Z)$, that is*

$$\varphi_i(Z) = \mathbb{E}[(Z - \mathbb{E}Z)^i].$$

Then the assumptions are, for any $x_1, x_2 \in \mathbb{R}^n$,

$$\varphi_2(F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) \leq \kappa_{\sigma^2} \quad (3.27)$$

$$\varphi_4(F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) \leq \kappa_{\sigma^4} \quad (3.28)$$

for some constants κ_{σ^2} and κ_{σ^4} .

Note that difference of the underlying function is the mean of the sample response difference

$$f(x_1) - f(x_2) = \mathbb{E}[F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))].$$

The assumptions in fact constrain the gap between the change of the sample response function and the change of the underlying function. The 4th central

moment exists for almost all statistical distributions. In Assumption 6, we consider two points x_1 and x_2 , because we would like to constrain their correlations (covariance, high order covariance) as well.

Moreover, for the averaged sample function $\hat{f}^N(x)$,

$$\begin{aligned}
& \varphi_4 \left(\hat{f}^N(x_1, \xi(\omega)) - \hat{f}^N(x_2, \xi(\omega)) \right) \\
&= \frac{1}{N^3} \varphi_4 (F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) + \\
&\quad \frac{3(N-1)}{N^3} \varphi_2^2 (F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) \\
&= \frac{1}{N^2} \left(\frac{1}{N} \varphi_4 (F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) + \right. \\
&\quad \left. \frac{3(N-1)}{N} \varphi_2^2 (F(x_1, \xi(\omega)) - F(x_2, \xi(\omega))) \right) \\
&\leq \frac{1}{N^2} (\kappa_{\sigma^4} + 3\kappa_{\sigma^2}^2). \tag{3.29}
\end{aligned}$$

Therefore, Assumption 6 implies that the 4th central moment of the change of averaged sample function decreases quadratically fast with the sample number N .

3.2.2 Convergence Analysis of the Algorithm

Convergence analysis of the general model-based approach is given by Conn, Scheinberg, and Toint in [20]. Since the model-based approach is in the trust region framework, their proof of global convergence follows general ideas for the proof of the standard trust region method [78, 80].

We start by showing that there is at least one stationary accumulation

point. The stationary point of a function is a point at which the gradient of the function is zero. The idea is to first show that the gradient g_k^∞ , driven by the ‘sufficient reduction’ criterion (3.20), converges to zero, and then prove that $\|\nabla f(x_k)\|$ converges to zero as well.

Lemma 3.6. *If Assumptions 1–6 hold and $\|g_k^\infty\| \geq \epsilon_g$ for all k and for some constant $\epsilon_g > 0$, then there exists a constant $\epsilon_\Delta > 0$ such that w.p.1,*

$$\Delta_k > \epsilon_\Delta, \text{ for all } k \geq K. \quad (3.30)$$

Proof. Given the condition $\|g_k^\infty\| \geq \epsilon_g$, we will show that the corresponding Δ_k cannot become too small, therefore, we can derive the constant ϵ_Δ .

Let us evaluate the following term associated with the agreement level

$$|\rho_k^{N_k} - 1| = \left| \frac{\hat{f}^{N_k}(x_k + s^{*,N_k}) - Q_k^{N_k}(x_k + s^{*,N_k})}{Q_k^{N_k}(x_k) - Q_k^{N_k}(x_k + s^{*,N_k})} \right|. \quad (3.31)$$

By Lemma 3.1, we compute the error bound for the numerator

$$\left| \hat{f}^{N_k}(x_k + s^{*,N_k}) - Q_k^{N_k}(x_k + s^{*,N_k}) \right| \leq \kappa_{em} \max[\Delta_k^2, \Delta_k^3]. \quad (3.32)$$

Note that when Δ_k is small enough, satisfying the condition

$$\Delta_k \leq \min \left[1, \frac{\kappa_{mdc} \epsilon_g (1 - \eta_1)}{\max[\kappa_{Qh}, \kappa_{em}]} \right], \quad (3.33)$$

according to the facts $\eta_1, \kappa_{mdc} \in (0, 1)$ and $\|g_k^\infty\| \geq \epsilon_g$, we deduce

$$\Delta_k \leq \frac{\|g_k^\infty\|}{\kappa_{Qh}}. \quad (3.34)$$

For the denominator in (3.31), our ‘sufficient reduction’ criterion (3.20) provides a lower bound for $Q_k^{N_k}(x_k) - Q_k^{N_k}(x_k + s^{*,N_k})$. When $k \geq K$ the inequality holds w.p.1

$$Q_k^{N_k}(x_k) - Q_k^{N_k}(x_k + s^{*,N_k}) \geq \kappa_{mdc} \|g_k^\infty\| \min \left[\frac{\|g_k^\infty\|}{\kappa_{Qh}}, \Delta_k \right] = \kappa_{mdc} \|g_k^\infty\| \Delta_k. \quad (3.35)$$

Combining (3.31), (3.32), (3.33) and (3.35), we see that the following inequality holds w.p.1 for iteration $k \geq K$

$$\begin{aligned} |\rho_k^{N_k} - 1| &= \left| \frac{\hat{f}^{N_k}(x_k + s^{*,N_k}) - Q_k^{N_k}(x_k + s^{*,N_k})}{Q_k^{N_k}(x_k) - Q_k^{N_k}(x_k + s^{*,N_k})} \right| \\ &\leq \frac{\kappa_{em} \max[\Delta_k^2, \Delta_k^3]}{\kappa_{mdc} \|g_k^\infty\| \Delta_k} \\ &\leq \frac{\kappa_{em} \Delta_k}{\kappa_{mdc} \|g_k^\infty\|} \\ &\leq 1 - \eta_1. \end{aligned} \quad (3.36)$$

The criterion $\rho_k^{N_k} \geq \eta_1$ implies the identification of a good agreement between the model $Q_k^{N_k}$ and the function \hat{f}^{N_k} , which will induce an increase of the trust region radius $\Delta_{k+1} \geq \Delta_k$ (3.3). We thus have

$$\rho_k^{N_k} \geq \eta_1 \text{ valid w.p.1 for all } k \geq K.$$

According to (3.33), it is equivalent to say that Δ_k can shrink only when

$$\Delta_k \geq \min \left[1, \frac{\kappa_{mdc} \epsilon_g (1 - \eta_1)}{\max[\kappa_{Qh}, \kappa_{em}]} \right].$$

We therefore derive a lower bound for Δ_k :

$$\Delta_k > \epsilon_\Delta = \gamma_0 \min \left[1, \frac{\kappa_{mdc} \epsilon_g (1 - \eta_1)}{\max[\kappa_{Qh}, \kappa_{em}]} \right], \text{ for } k \geq K. \quad (3.37)$$

□

□

Theorem 3.1. *If Assumptions 1–6 hold, then, w.p.1*

$$\liminf_{k \rightarrow \infty} \|g_k^\infty\| = 0. \quad (3.38)$$

Proof. We prove the statement (3.38) by contradiction. Suppose there is $\epsilon_g > 0$ such that

$$\|g_k^\infty\| \geq \epsilon_g. \quad (3.39)$$

By Lemma 3.6, we have w.p.1, $\Delta_k > \epsilon_\Delta$ for $k \geq K$.

We first show there exists only finitely many successful iterations. If not, suppose we have infinitely many successful iterations. At each successful iteration $k \geq K$, by (3.2), (3.20), (3.39) and $\Delta_k > \epsilon_\Delta$, the inequality

$$\begin{aligned} \hat{f}^{N_k}(x_k) - \hat{f}^{N_k}(x_{k+1}) &\geq \eta_0 \left[Q_k^{N_k}(x_k) - Q_k^{N_k}(x_k + s^{*,N_k}) \right] \\ &\geq \eta_0 \kappa_{mdc} \epsilon_g \min \left[\frac{\epsilon_g}{\kappa_{Qh}}, \epsilon_\Delta \right] \end{aligned} \quad (3.40)$$

holds w.p.1.

We will discuss two situations here: (a) when the limit of the sequence $\lim_{k \rightarrow \infty} N_k = N_\infty$ is a finite number, and (b) when N_∞ is infinite. Both situations are possible in our algorithm. For simplicity, we use \mathcal{S} to denote the index set of successful iterations and define

$$\epsilon_d := \eta_0 \kappa_{mdc} \epsilon_g \min \left[\frac{\epsilon_g}{\kappa_{Qh}}, \epsilon_\Delta \right],$$

the positive reduction in right hand side of (3.40).

Situation (a): If $N_\infty < \infty$, then there exists an index $\tilde{K} \geq K$ such that $N_k = N_\infty$ for $k \geq \tilde{K}$. Since $\{\hat{f}^{N_\infty}(x_k) | k \geq \tilde{K}\}$ is monotonically decreasing

$$\begin{aligned} \hat{f}^{N_\infty}(x_{\tilde{K}}) - \hat{f}^{N_\infty}(x_{\hat{K}+1}) &\geq \sum_{\substack{k \geq \tilde{K}, k \leq \hat{K}, \\ k \in \mathcal{S}}} \hat{f}^{N_\infty}(x_k) - \hat{f}^{N_\infty}(x_{k+1}) \\ &\geq t(\hat{K})\epsilon_d, \end{aligned} \quad (3.41)$$

where \hat{K} is a large index in \mathcal{S} and $t(\hat{K})$ is a count number of indexes in the summation term. Since \hat{f}^{N_∞} is bounded below (Assumption 2), we know that $\hat{f}^{N_\infty}(x_{\tilde{K}}) - \hat{f}^{N_\infty}(x_{\hat{K}+1})$ is a finite value. However, the right hand side goes to infinity because there are infinitely many indexes in \mathcal{S} w.p.1 ($t(\hat{K}) \rightarrow \infty$, as $\hat{K} \rightarrow \infty$). This induces a contradiction, therefore, there are only a finite number of successful iterations.

Situation (b): For this situation, $N_\infty = \infty$. Let us define a specific subsequence of indexes $\{k_{j'} | k_{j'} \geq K\}$ (see Figure 22), indicating where there is a jump in N_k , i.e., a truncated part of subsequence is

$$\cdots < N_{k_{j'}} = N_{k_{j'}+1} = \cdots = N_{k_{j'+1}-1} < N_{k_{j'+1}} = \cdots .$$

Let \mathcal{S}' be a subset of $\{k_{j'}\}$, including $k_{j'}$ if there is at least one successful iteration in $\{k_{j'}, \dots, k_{j'+1} - 1\}$. This implies

$$x_{k_{j'+1}} \begin{cases} \neq x_{k_{j'}}, & \text{for } k_{j'} \in \mathcal{S}'; \\ = x_{k_{j'}} \text{ (unchanged)}, & \text{for } k_{j'} \notin \mathcal{S}'. \end{cases}$$

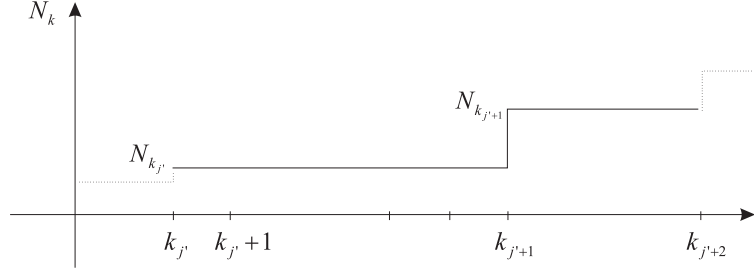


Figure 22: Illustration of the subsequence $\{k_{j'}\}$.

For $k_{j'} \in \mathcal{S}'$, sum the inequality (3.40) for $k \in \{N_{k_{j'}}, \dots, N_{k_{j'+1}-1}\}$ to derive

$$\begin{aligned} \hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'+1}}}(x_{k_{j'+1}}) &\geq \sum_{\substack{k \geq k_{j'}, k \leq k_{j'+1}-1 \\ k \in \mathcal{S}'}} \hat{f}^{N_{k_{j'}}}(x_k) - \hat{f}^{N_{k_{j'+1}}}(x_{k+1}) \\ &\geq \epsilon_d. \end{aligned} \quad (3.42)$$

We want to quantify the difference between $\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'+1}}}(x_{k_{j'+1}})$ and $f(x_{k_{j'}}) - f(x_{k_{j'+1}})$. The idea behind this is that moving from $x_{k_{j'}}$ to $x_{k_{j'+1}}$, the function $\hat{f}^{N_{k_{j'}}}$ decreases, and so does the underlying function f . Since infinitely many decrement steps for f are impossible, we derive a contradiction.

Define the event $\hat{E}_{k_{j'}}$ as the occurrence of $\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'+1}}}(x_{k_{j'+1}}) \geq \epsilon_d$

while $f(x_{k_{j'}}) - f(x_{k_{j'+1}}) \leq \frac{\epsilon_d}{2}$. The probability of event

$$\begin{aligned}
& Pr\left(\hat{E}_{k_{j'}}\right) \\
& \leq Pr\left(\left(\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})\right) - \left(f(x_{k_{j'}}) - f(x_{k_{j'+1}})\right) \geq \frac{\epsilon_d}{2}\right) \\
& \leq Pr\left(\left|\left(\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})\right) - \left(f(x_{k_{j'}}) - f(x_{k_{j'+1}})\right)\right| \geq \frac{\epsilon_d}{2}\right) \\
& = Pr\left(\left(\left(\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})\right) - \left(f(x_{k_{j'}}) - f(x_{k_{j'+1}})\right)\right)^4 \geq \left(\frac{\epsilon_d}{2}\right)^4\right) \\
& \leq \frac{16}{\epsilon_d^4} \cdot \mathbb{E}\left[\left(\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})\right) - \left(f(x_{k_{j'}}) - f(x_{k_{j'+1}})\right)\right]^4 \\
& = \frac{16}{\epsilon_d^4} \cdot \varphi_4\left(\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})\right) \\
& \leq \frac{16\left(\kappa_{\sigma^4} + 3\kappa_{\sigma^2}^2\right)}{\epsilon_d^4(N_{k_{j'}})^2}.
\end{aligned}$$

The third inequality is due to Markov's inequality [30]. The random quantity $\hat{f}^{N_{k_{j'}}}(x_{k_{j'}}) - \hat{f}^{N_{k_{j'}}}(x_{k_{j'+1}})$ has mean value $f(x_{k_{j'}}) - f(x_{k_{j'+1}})$. The last inequality is due to the implication of Assumption 6, see (3.29).

The result implies that probability of the event \hat{E}_k decreases quadratically fast with k . Since the sum of the probability values is finite

$$\sum_{\substack{j'=1 \\ k_{j'} \in \mathcal{S}'}}^{\infty} Pr\left(\hat{E}_{k_{j'}}\right) \leq \sum_{\substack{j'=1 \\ k_{j'} \in \mathcal{S}'}}^{\infty} \frac{16\left(\kappa_{\sigma^4} + 3\kappa_{\sigma^2}^2\right)}{\epsilon_d^4(N_{k_{j'}})^2} < \infty,$$

applying the Borel-Cantelli Lemma again, the event $\hat{E}_{k_{j'}}$ occurs only finitely many times w.p.1. Thus, there exists an index \bar{K} , such that

$$f(x_{k_{j'}}) - f(x_{k_{j'+1}}) \geq \frac{\epsilon_d}{2}, \text{ for all } \{k_{j'} | k_{j'} \geq \bar{K}, k_{j'} \in \mathcal{S}'\} \text{ w.p.1.}$$

Playing the same trick as before, by summing over all $k_{j'} \geq \bar{K}$, we derive

that w.p.1

$$\begin{aligned}
f(x_{\bar{K}}) - f(x_{\hat{K}+1}) &\geq \sum_{\substack{k_{j'} \geq \bar{K}, k_{j'} \leq \hat{K} \\ k_{j'} \in \mathcal{S}'}} f(x_{k_{j'}}) - f(x_{k_{j'}+1}) \\
&\geq t(\hat{K}) \frac{\epsilon_d}{2}.
\end{aligned} \tag{3.43}$$

The left hand side is a finite value, but the right hand side goes to infinity. This contradiction also shows that the number of successful iterations is finite.

Combining the two situations above, we must have infinitely many unsuccessful iterations when k is sufficiently large. As a consequence, the trust region radius Δ_k decreases to zero

$$\lim_{k \rightarrow \infty} \Delta_k = 0,$$

which contradicts the statement that Δ_k is bounded below (3.37). Thus (3.39) is false, and the theorem is proved. \square \square

Theorem 3.2. *If Assumptions 1–6 hold and*

$$\liminf_{j \rightarrow \infty} \|g_{k_j}^\infty\| = 0 \text{ w.p.1} \tag{3.44}$$

holds for a subsequence $\{k_j\}$, then we also have

$$\liminf_{j \rightarrow \infty} \|\nabla f(x_{k_j})\| = 0 \text{ w.p.1.} \tag{3.45}$$

Proof. Due to the fact $\lim_{j \rightarrow \infty} \Delta_{k_j} = 0$, Lemma 3.1 guarantees that the difference between $\|g_{k_j}^\infty\|$ and $\|\nabla f(x_{k_j})\|$ is small. Thus the assertion (3.45) follows. The details of the proof refer to Theorem 11 in [20]. \square \square

Theorem 3.3. *If Assumptions 1–6 hold, every limit point x^* of the sequence $\{x_k\}$ is stationary.*

Proof. The procedure of proof is essentially the same as given for Theorem 12 in [20]. However, we use the ‘sufficient reduction’ inequalities (3.41) when N_∞ is finite and (3.43) when N_∞ is infinite. \square \square

3.2.3 Numerical Results

We apply the VNSP-UOBYQA algorithm (Procedure 8) to several numerical examples. The noisy test functions are altered from deterministic functions with artificial randomness.

The first numerical function we employed was the well-known extended Rosenbrock function. The random term was added only to the first component of the input variable. Define

$$\hat{x}(x, \xi(\omega)) := (x_{(1)}\xi(\omega), x_{(2)}, \dots, x_{(n)})$$

and the corresponding function becomes

$$F(x, \xi(\omega)) = \sum_{i=1}^{n-1} 100(\hat{x}_{(i+1)} - \hat{x}_{(i)}^2)^2 + (\hat{x}_{(i)} - 1)^2. \quad (3.46)$$

We assume $\xi(\omega)$ is a normal variable centered at 1:

$$\xi(\omega) \sim N(1, \sigma^2).$$

As a general setting, the initial and end trust region radius Δ_0, Δ_{end} were set to 2 and $1.0e - 5$, respectively. Implementing the algorithm required a

starting value $N_0 = 3$, which was used to estimate the initial sample mean and sample covariance matrix. We believe such a value is the minimum required for reasonable estimates. Larger values of N_0 would in most cases lead to wasted evaluations. $M = 500$ (see (3.24)) trials samples were generated to evaluate the Bayes probability (3.23) in the VNSP procedure. To satisfy Assumption 5, the sequence $\{\alpha_k\}$ was pre-defined as

$$\alpha_k = 0.5 \times (0.98)^k.$$

Table 5: The performance of the new algorithm for the noisy Rosenbrock function, with $n = 2$ and $\sigma^2 = 0.01$.

Iteration k	N_k	FN	x_k	$f^{N_k}(x_k)$	Δ_k
0	3	3	(-1.0000,1.2000)	11.7019	2.0
19	3	81	(0.5002,0.2449)	0.3616	0.1
20	4	91	(0.5002,0.2449)	0.4904	0.05
21	5	102	(0.5208,0.2904)	0.4944	0.02
22	22	226	(0.5082,0.2864)	0.4018	0.02
23	22	248	(0.5082,0.2864)	0.4018	0.02
24	30	326	(0.5082,0.2864)	0.5018	0.02
29	30	476	(0.4183,0.1862)	0.4447	0.04
30	113	1087	(0.4328,0.1939)	0.4290	0.02
31	113	1200	(0.4328,0.1939)	0.4290	0.02
32	221	1848	(0.4328,0.1939)	0.4437	0.02
33	604	4750	(0.4328,0.1939)	0.4601	0.01
35	604	5958	(0.4276,0.1837)	0.4569	0.0125
36	845	8249	(0.4197,0.1774)	0.4556	0.0101
37	1183	10277	(0.4172,0.1760)	0.4616	0.0101

Table 5 presents the details about a single-run of the new algorithm on the two-dimensional Rosenbrock function with $\sigma^2 = 0.01$. The starting point

was chosen to be $(-1, 1.2)$, and the maximum number of function evaluations was 10000. We recorded the iteration number k when there was a change in N_k . For example, N_k remained at 3 in iterations 1–19, and N_k changed to 4 at iteration 20. Since in the first 19 iterations, the averaged sample function was \hat{f}^3 , all the steps were taken regarding \hat{f}^3 as the objective function. Therefore, it was observed that the iterates x_k moved toward the solution $x^{*,3}$ of the averaged sample problem (3.8) with $N = 3$. In Table 6 we present the corresponding sample-path solution of the optimization problem (3.8). For example, $x^{*,3} = (0.5415, 0.2778)$. Note that, in order to derive the solution to f in the two dimensional problem, the noisy Rosenbrock function was rearranged as

$$\begin{aligned} f(x) &= \mathbb{E} [100(\hat{x}_{(2)} - \hat{x}_{(1)}^2)^2 + (\hat{x}_{(1)} - 1)^2] \\ &= 100x_{(2)}^2 + 1 - 2x_{(1)}\mathbb{E}[\xi] + (-200x_{(2)}x_{(1)}^2 + x_{(1)}^2)\mathbb{E}[\xi^2] + 100x_{(1)}^4\mathbb{E}[\xi^4]. \end{aligned}$$

By plugging the values $\mathbb{E}[\xi] = 1$, $\mathbb{E}[\xi^2] = 1.01$, and $\mathbb{E}[\xi^4] = 1.0603$, we obtained the solution $x^{*,\infty} = (0.4162, 0.1750)$, which was different from the deterministic Rosenbrock solution $(1, 1)$. For different N_k , the averaged function \hat{f}^{N_k} might vary greatly. In Table 5, we observe that $x_{19} = x_{20} = (0.5002, 0.2449)$. The value of $\hat{f}^{N_{19}}(x_{19})$ is 0.3616, while the value of $\hat{f}^{N_{20}}(x_{20})$ is 0.4904. It shows that the algorithm actually worked on objective functions with increasing accuracy.

As shown in Table 5, the algorithm used a small N_k to generate new

Table 6: Averaged sample-path solution with different sample number N

N	$x^{*,N}$	$\hat{f}^{N_k}(x^{*,N})$
3	(0.5415,0.2778)	0.3499
4	(0.4302,0.1922)	0.4412
5	(0.4218,0.1936)	0.4395
22	(0.4695,0.2380)	0.3892
30	(0.4222,0.1896)	0.4446
113	(0.4423,0.2027)	0.4286
221	(0.4331,0.1910)	0.4427
604	(0.4226,0.1798)	0.4567
845	(0.4236,0.1807)	0.4556
1183	(0.4174,0.1761)	0.4615
∞	(0.4162,0.1750)	0.4632

iterates in the earlier iterations. Only 476 function evaluations were applied for the first 29 iterations. This implies that when noisy effects were small compared to the large change of function values, the basic operation of the method was unchanged and $N_k = N_0$ samples were used. As the algorithm proceeded, the demand for accuracy increased, therefore, N_k increased as well as the total number of function evaluations. We obtained very good solutions. At the end of the algorithm, we generated a solution $x_{37} = (0.4172, 0.1760)$, which is close to the averaged sample-path solution $x^{*,N=1183} = (0.4174, 0.1761)$ and is better than the solution $x^{*,N=845} = (0.4236, 0.1807)$. In a standard sample-path optimization method, assuming that there are around 40 iterations in the algorithm, we need $845 \times 40 = 33800$ function evaluations for the solution $x^{*,N=845}$ and $1183 \times 40 = 43720$ for the

solution $x^{*,N=1183}$. Our algorithm indeed saved a significant amount of function operations.

To study the changes of N_k , in Figure 23, we plot N_k against the iteration number for two problems. One is a high volatility case with $\sigma^2 = 1$ and the other is a low volatility case with $\sigma^2 = 0.01$. In both problems, N_k was 3 for the first 20 iterations, when the noise is not the dominating factor. In the later iterations, the noise became significant and we observe that the demand for N_k increased faster for the high volatility case. If we restricted the total function evaluations to be 10000, the high volatility case resulted in a early termination at the 34th iteration.

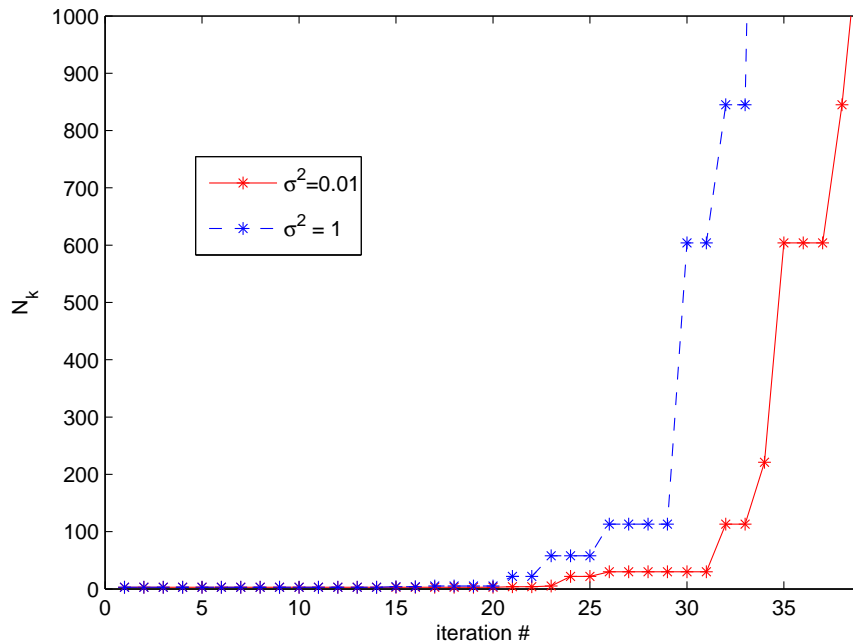


Figure 23: Compare changes of N_k with different levels of noise.

We applied the algorithm to both 2 and 10 dimensional problems. Increasing the dimension significantly increased computational burden. The problem with dimension $n = 10$ is already very hard to tackle. Even in the deterministic case, the standard UOBYQA requires around 1400 iterations to terminate at $\Delta_{end} = 0.0001$. In Table 7, we record a summary of the algorithm applied to the Rosenbrock function with different dimensions and noise levels. For comparisons, we include the result of the standard sample-path methods with fixed numbers of samples: 10, 100, and 1000. The statistical results are based on 10 replications of the algorithm. The variance of the error is small, showing that the algorithm was generally stable. For $n = 10$ and $\sigma^2 = 1$, we notice a big mean error 2.6 and a relatively small variance of error 0.10. This is due to the earlier termination of the algorithm when σ^2 is large (we used a limit of 20000 function evaluations in this case). There are two reasons why the standard sample-path methods yield relatively larger errors. 1) Methods SP(10) and SP(100) do not provide accurate averaged sample functions \hat{f}^N . 2) For a large sample number N , the iteration number of the algorithm is limited. For example, we can expect SP(100) is limited to 200 iterations and SP(1000) is limited to 20 iterations. Increasing the total number of function evaluations can significantly improve the performance of the sample path optimization methods. For example, if we allow 2,000,000 total function evaluations for the 10 dimensional case and the noise level $\sigma^2 = 1$, the mean error of SP(100) and SP(1000) are 1.6, 7.5, respectively.

The VNSP method performs better than this.

Table 7: Statistical summary

n	Noise level σ^2	VNSP		SP(10)	SP(100)	SP(1000)
		Mean error	Variance of error	Mean error	Mean error	Mean error
2	0.01	1.1e-5	1.2e-5	0.035	0.0045	7.9e-5
2	0.1	8.9e-5	3.3e-5	0.079	0.0067	4.2e-4
2	1	1.1e-4	8.2e-5	0.098	0.0088	8.9e-4
10	0.01	0.054	0.067	0.44	28	120
10	0.1	0.087	0.060	2.1	44	129
10	1	2.6	0.10	14	32	145

For another test example, we refer back to the toy example. The objective function is only affected by additive noise

$$F(x, \xi(\omega)) = \phi(x) + \xi(\omega).$$

We will show N_k is unchanged for every iteration, that is, $N_1 = N_2 = \dots = N_\infty$. At iteration k , the function outputs at points y^j in \mathcal{I}_k are entirely correlated. As a result, the sample covariance matrix $\hat{\Sigma}$ is a rank-one matrix, whose elements are all identical $\hat{\Sigma}(i, j) = a$, $i, j = 1, 2, \dots, L$, where $a = \text{var}[(\xi_1, \dots, \xi_{N_k})]$. Thus, the matrix can be decomposed as

$$\hat{\Sigma} = \mathbf{1} \cdot a \cdot \mathbf{1}^T. \quad (3.47)$$

Plug (3.47) into (3.15), we obtain the posterior covariance of g_k^∞

$$\text{cov}(g_k^\infty | X^N) = (\mathbf{g} \cdot \mathbf{1})^T \cdot a \cdot (\mathbf{g} \cdot \mathbf{1}) = (\mathbf{0})^T \cdot a \cdot \mathbf{0} = \mathbf{0}_{L \times L},$$

which implies g_k^∞ is not random and $g_k^\infty = g_k^{N_k}$. As a consequence, in the VNSP scheme, the mechanism will not increase N_k because the criterion (3.20) is always satisfied.

The fact $\mathbf{g} \cdot \mathbf{1} = \sum_{j=1}^L g_j = \mathbf{0}$ is a property of Lagrange functions. The proof is simple - the sum of Lagrange functions $\sum_{j=1}^L l_j(x)$ is the unique quadratic interpolant of a constant function $\hat{g}(x) = 1$ at the points y^j , because $\sum_{j'=1}^L l_{j'}(y^j) = 1 = \hat{g}(y^j)$, $j = 1, \dots, L$. Therefore, the gradient of the interpolant $\sum_{j=1}^L g_j = \mathbf{0}$.

In practice, the behavior of the toy example occurs rarely. We present it here to show that our algorithm indeed checks the uncertainty of each iterate x_k , but not that of objective value $\hat{f}^{N_k}(x_k)$.

3.3 The Noisy UOBYQA Algorithm

We develop a variant of the UOBYQA algorithm, called the Noisy UOBYQA algorithm, that is adapted for simulation-based optimization problem in the white noise case. The modification is close to what we present for the VNSP-UOBYQA algorithm, i.e., in the construction of quadratic models. The key difference is that we allow different numbers of samples for each point in the algorithm, including the point y^j the interpolation point set \mathcal{I}_k . For example, in our following discussion, r_j samples are evaluated for each point y^j ; while for the VNSP-UOBYQA algorithm, we use the same number N_k

for all y^j in \mathcal{I}_k . This is because we can take advantage of the correlations among samples. The sampling strategy in the Noisy UOBYQA algorithm is based on an alternative stability criterion.

3.3.1 Modifications

When noise is present, UOBYQA may behave precariously. For example, a subproblem that minimizes the quadratic model within a trust region may generate poor situations. The idea of our modification is to control the random error by averaging multiple evaluations per point, helping the algorithm to proceed appropriately. In the following subsections, we will present our modifications to the UOBYQA algorithm to deal with problematic points of the algorithm in the noisy case.

Reducing Quadratic Model Variance

When there is uncertainty in the objective function, the existence of noise can cause erroneous estimations of coefficients of the quadratic model $Q_k(x)$, say c_k, g_k, G_k , and as a result, generate an unstable solution $x_k + s^*$. To reduce the variance of the quadratic model, we consider generating multiple function values for points $y^j, j = 1, 2, \dots, L$ in the set \mathcal{I}_k and use the averaged function value for the interpolation process.

In UOBYQA, the quadratic function is constructed as a linear combination of Lagrange functions $l_j(x)$,

$$Q_k(x) = \sum_{j=1}^L f(y^j)l_j(x), \quad x \in \mathbb{R}^n. \quad (3.48)$$

The parameters of Q_k are derived as

$$\begin{aligned} c_k &= \sum_{j=1}^L f(y^j)c_j, \quad g_k = \sum_{j=1}^L f(y^j)g_j, \\ \text{and } G_k &= \sum_{j=1}^L f(y^j)G_j. \end{aligned} \quad (3.49)$$

Here, c_j , g_j and G_j are the coefficients of the quadratic function l_j . See details in Section 3.2.1.

In the Bayesian analysis, the posterior distribution of μ can be derived as

$$\mu(y^j)|X \sim N(\bar{\mu}(y^j), \hat{\sigma}^2(y^j)/r_j), \quad (3.50)$$

where r_j is the number of replications for the point y^j . According to (3.49), we treat the c_k , g_k and G_k as random variables from a Bayesian perspective. They all follow normal distributions whose means are estimated using (3.50) as (all of these are posterior estimates)

$$\begin{aligned} \mathbb{E}[c_k] &= \mathbb{E} \left[\sum_{j=1}^L f(y^j, \omega)c_j \right] \\ &= \mathbb{E} \left[\sum_{j=1}^L \mu(y^j)c_j \right] \\ &= \sum_{j=1}^L \bar{\mu}(y^j)c_j, \\ \mathbb{E}[g_k] &= \sum_{j=1}^L \bar{\mu}(y^j)g_j, \\ \mathbb{E}[G_k] &= \sum_{j=1}^L \bar{\mu}(y^j)G_j \end{aligned} \quad (3.51)$$

and the variances are estimated as

$$\begin{aligned}
\text{var}(c_k) &= \text{var} \left(\sum_{j=1}^L \mu(y^j) c_j \right) \\
&= \sum_{j=1}^L c_j^2 \hat{\sigma}^2(y^j) / r_j, \\
\text{var}(g_k(i')) &= \sum_{j=1}^L g_j^2(i') \hat{\sigma}^2(y^j) / r_j, \\
\text{var}(G_k(i', j')) &= \sum_{j=1}^L G_j^2(i', j') \hat{\sigma}^2(y^j) / r_j, \\
& \quad i', j' = 1, \dots, n.
\end{aligned} \tag{3.52}$$

As r_j increases to infinity, the variance decreases to zero.

To increase the stability of a quadratic model, we want to quantify how the randomness in the coefficients c_k , g_k , and G_k affects the solution of the subproblem (3.8). Suppose we solve N_t trial subproblems whose quadratic model coefficients are extracted from their posterior distributions, we expect that solutions $s^{*(i)}$, $i = 1, 2, \dots, N_t$ have a small overall variance (see Figure 24). Therefore, we introduce a criterion that constrains the standard deviation of solutions in each coordinate direction, requiring them to be smaller than a threshold value β times the trust region radius:

$$\max_{j=1}^n \text{std}([s^{*(1)}(j), s^{*(2)}(j), \dots, s^{*(N_t)}(j)]) \leq \beta \Delta_k. \tag{3.53}$$

Increasing r_j should help reduce the volatility of coefficients, thus reduce the standard deviations of solutions. Therefore, satisfying the above criterion necessitates a sufficiently large r_j for the point y^j .

Sequentially allocating computational resources A new resource allocation question arises (based on r_j function evaluations at site y^j) about

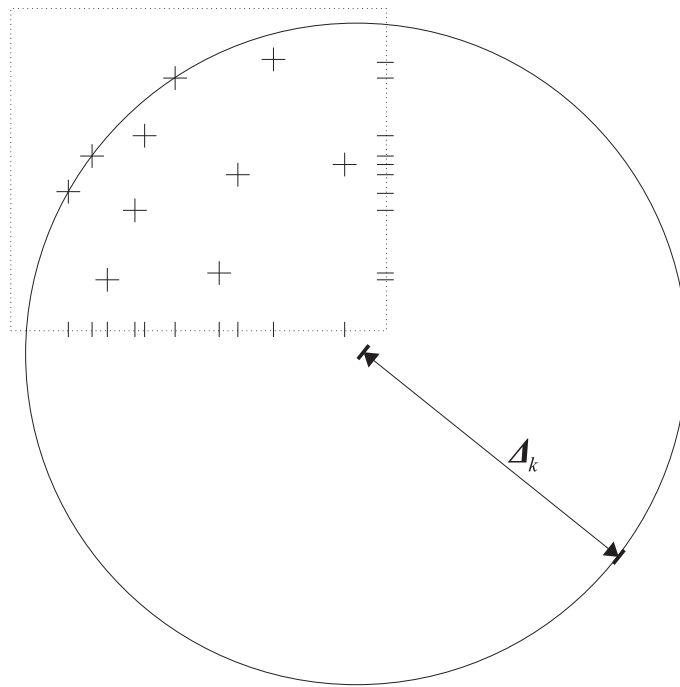


Figure 24: The trial solutions within a trust region are projected to each coordinate direction and the standard deviations of the projected values are evaluated.

which site to assign new function replications in order to satisfy the constraint (3.53) with the minimum total number of function evaluations. In solving the subproblem (3.1), we know that only g_k and G_k matter for determining the solution s^* from the definition of Q_k in (3.5). Instead of satisfying the constraint directly, we aim to control the variance of the most volatile coefficient. We minimize the corresponding ratio of the standard deviation to the expected value

$$\max_{i', j'} \left(\frac{\text{std}(g_k(i'))}{|E[g_k(i')]|}, \frac{\text{std}(G_k(i', j'))}{|E[G_k(i', j')]|} \right), \quad (3.54)$$

$$i', j' = 1, \dots, n.$$

We propose a fast but sub-optimal strategy here, one that assigns additional computational resources sequentially. We assume that when an additional batch of r replications at site y^j are newly produced, the sample mean $\bar{\mu}(y^j)$ and variance $\hat{\sigma}^2(y^j)$ remain invariant. Under this assumption, the posterior variance of $\mu(y^j)$ (see (3.50)) changes from $\hat{\sigma}^2(y^j)/r_j \rightarrow \hat{\sigma}^2(y^j)/(r_j + r)$.

First, let $\phi(\vec{r})$ (here $\vec{r} = [r_1, r_2, \dots, r_L]$) denote the largest quantity in (3.54):

$$\begin{aligned} \phi(\vec{r}) &= \max_{i', j'} \left(\frac{\text{std}(g_k(i'))}{E[g_k(i')]}, \frac{\text{std}(G_k(i', j'))}{E[G_k(i', j')]} \right) \\ &= \max_{i', j'} \left(\frac{\sqrt{\sum_{j=1}^L g_j^2(i') \hat{\sigma}^2(y^j)/r_j}}{\sum_{j=1}^L \bar{\mu}(y^j) g_j(i')}, \frac{\sqrt{\sum_{j=1}^L G_j^2(i', j') \hat{\sigma}^2(y^j)/r_j}}{\sum_{j=1}^L \bar{\mu}(y^j) G_j(i', j')} \right), \quad (3.55) \\ & \quad i', j' = 1, \dots, n. \end{aligned}$$

To achieve a good allocation scheme, we invest our new resources in the point y^j in order to most sharply decrease the quantity $\phi(\vec{r})$. After assigning

r samples to the point y^j , we obtain a new vector $\vec{r} + re_j$, where e_j is the standard unit vector in \mathbb{R}^L with 1 on the j th component. We select the site y^j with the index:

$$\arg \max_j \phi(\vec{r} + re_j). \quad (3.56)$$

The best index is determined by comparing the L different possible options.

Procedure 9. *Stabilize the quadratic model:*

Given initial sample size r_0 , batch size r , and the threshold value β .

1. *Generate r_0 function evaluations for each point for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j = 1, 2, \dots, L$.*
2. *Determine the largest quantity $\phi(\vec{r})$ which corresponds to the most volatile coefficient.*
3. *Select the site for further evaluations using (3.56).*
4. *Evaluate r function replications on the selected site and update the sample mean and sample variance.*
5. *Repeat Steps 2-4, until constraint (3.53) is satisfied.*

Selecting the Best Point

If x^+ is a new point entering the interpolation set \mathcal{I}_k , we encounter the problem of selecting the best point in \mathcal{I}_k . Since x_k is known to be the best in the previous set, the question becomes how we determine the order of x_k

and x^+ . When there is noise in the function output, we need more precise estimations of the underlying function value to make the correct decision.

In our case, since there are only two points involved, we implement a simplified variant of a ranking and selection procedure *OCBA* [17] (refer to Section 1.2.3). Suppose we have replicated r_j function values for y^j , let $\mu(y^j)$ and $\sigma^2(y^j)$ be the unknown mean and variance of the output at y^j . The point evidenced by a smaller sample mean is selected. The approximate *PSC* is a tail probability of a normal distribution:

$$PCS \sim Pr \left(N \left(\bar{\mu}(y^1) - \bar{\mu}(y^2), \frac{\hat{\sigma}^2(y^1)}{r_1} + \frac{\hat{\sigma}^2(y^2)}{r_2} \right) \leq 0 \right).$$

Sequentially allocating computational resources We consider a sequential allocation strategy to assign new computational resources to the points y^1 and y^2 , so that we can use the least number of function evaluations to satisfy the rule

$$PCS \geq 1 - \alpha. \quad (3.57)$$

The change of sample mean and sample variance follows the assumptions in Section 3.3.1.

To determine the potentially better point, we compare the derivative value:

$$\max_{j \in \{1,2\}} \frac{\partial}{\partial r_j} Pr \left(N \left(\bar{\mu}(y^1) - \bar{\mu}(y^2), \frac{\hat{\sigma}^2(y^1)}{r_1} + \frac{\hat{\sigma}^2(y^2)}{r_2} \right) \leq 0 \right).$$

It is not hard to find that, since the mean of the joint distribution $\bar{\mu}(y^1) - \bar{\mu}(y^2)$ is unchanged, we desire the largest decrease in the variance $\psi(\vec{r}) :=$

$\hat{\sigma}^2(y^1)/r_1 + \hat{\sigma}^2(y^2)/r_2$. Therefore, the problem (3.58) becomes

$$\min_{j \in \{1,2\}} \psi(\vec{r} + r e_j). \quad (3.58)$$

This determines the index of the potentially better point.

Procedure 10. *Select the best point in \mathcal{I}_k :*

Given initial sample size r_0 , batch size r , and a significance parameter α .

1. *Evaluate r_0 function evaluations for each point for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j = 1, 2$.*
2. *Select the point to carry out further replications using (3.58).*
3. *Evaluate r further function replications on the point selected and update the sample mean and sample variance.*
4. *Repeat Steps 2 and 3 until the constraint (3.57) is satisfied.*

New Termination Criterion

In UOBYQA and other model-based optimization algorithms, a test on the norm of the gradient $\|g_k\|$ or the trust region size Δ_k is typically treated as termination criteria; i.e., $\Delta_k \leq \Delta_{end} = 10^{-12}$. However, these criteria are not suitable for noisy cases. When Δ_k gets smaller, the estimated value of G_k , which is a gauge for ‘curvature’ of the quadratic model, will approach zero. This will inevitably require much more function evaluations in later

iterations to reduce the variance of the quadratic model in order to retain accuracy.

The new termination criterion is designed to relax the value of Δ_{end} , such that the algorithm terminates much earlier. We specify a parameter N_{max} that controls the maximum number of replications per site in the algorithm. N_{max} represents the amount of computing we are willing to spend at any site. In each iteration k , we will check that any point x on the edge of the subregion $\{x \mid \|x - x_k\| \leq \Delta_k\}$ is ‘separable’ from the center x_k , given the control N_{max} on the number of replications per site. Here ‘separable’ means that one point is better than the other with high accuracy ($PCS \geq 1 - \alpha$).

The difficulty occurs in estimations of sample mean $\bar{\mu}(x)$ and sample variance $\hat{\sigma}^2(x)$ of an edge point x , which demand additional function evaluations. We simplify this procedure by performing the separability test using $Q_k(\cdot)$ instead of the original $f(\cdot)$, because Q_k is a good surrogate model of the underlying mean function f when Δ_k is small. We can (a) approximate $\bar{\mu}(x)$ with $Q_k(x)$ and (b) approximate $\hat{\sigma}^2(x)$ with $\hat{\sigma}^2(x_k)$. The second approximation is valid because the variances of function output at two points are very close within a small trust region.

In fact, we enumerate $2n$ edge points (of $\{x \mid \|x - x_k\| < \Delta_k\}$) which are in standard coordinate directions from x_k , and check their separability from x_k . The $2n$ points consist of a representative set of edge points. The stopping criterion is met when a portion of the representative point set, i.e., 80% of

the $2n$ points, are separable from x_k .

In practice, what we first calculate is a least ‘separable’ distance d . By observing the posterior distribution with the approximations:

$$\begin{aligned} & \mu(x) - \mu(x_k) | X \\ & \sim N(f(x) - f(x_k), \hat{\sigma}^2(x) + \hat{\sigma}^2(x_k)/N_{max}) \\ & \approx N(Q_k(x) - Q_k(x_k), 2\hat{\sigma}^2(x_k)/N_{max}). \end{aligned}$$

We compute d , satisfying

$$Pr(N(d, 2\hat{\sigma}^2(x_k)/N_{max}) \geq 0) \geq 1 - \alpha,$$

via an inverse cdf function evaluation. Then, we say x and x_k are ‘separable’ if and only if the difference $|Q_k(x) - Q_k(x_b)| \geq d$.

3.3.2 Numerical Results

Numerical Functions

We tested the noisy UOBYQA algorithm on several numerical examples and compared it with two other noisy optimization tools, NOMADm (Nonlinear Optimization for Mixed vAriables and Derivatives in Matlab) [2] and SPSA (Simultaneous Perturbation Stochastic Approximation) [104]. NOMADm is a pattern search algorithm that implements ranking and selection; and SPSA is a line search algorithm that applies simultaneous perturbation for gradient estimation.

The test function we employed was the well-known extended Rosenbrock function:

$$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2.$$

The function has a global optima at the point $(1, 1, \dots, 1)'_n$, at which the optimal objective value attains 0. A noisy optimization problem was formulated by adding a white noise term to the objective. We also applied our algorithm to the ARGLINC function with very similar results.

We assume the random term $\xi(\omega)$ was independent of x and followed a normal distribution with mean zero and variance σ^2 , indicating the levels of noise. The starting point x_0 was $(-1.2, 1, -1.2, 1, \dots, -1.2, 1)'_n$ and the initial trust region radius Δ_0 was set to 2.

Table 8 presents the details about a single-run of the Noisy UOBYQA algorithm to the two-dimensional Rosenbrock function. The variance σ^2 was set to 0.01, which was moderately noisy from our tested noise levels. We used the following default setting of parameters: initial sampling number $r_0 = 3$, which is small but enough to generate sample mean and sample variance; the significance level $\alpha = 0.2$, the threshold value $\beta = 0.4$, the number of trail solutions $N_t = 20$, and the maximum number of evaluations per point $N_{max} = 60$. The new termination criterion we introduced terminated the algorithm earlier, at the 81st iteration. In fact, we did not observe noticeable improvement in terms of objective value (0.0017 to 0.0016) between the iteration 80 and 120, because the presence of noise deteriorated the accuracy

in these iterations. In order to keep the algorithm running and maintaining correct decisions, a large number of function evaluations were necessary. It was therefore preferable to stop the algorithm early.

In the earlier iterations, the algorithm used relatively few function replications. This implied that when noisy effects were small compared to the large function values, the basic operation of the method was unchanged.

Table 8: The performance of the Noisy UOBYQA algorithm for the Rosenbrock function, with $n = 2$ and $\sigma^2 = 0.01$.

Iteration (k)	FN	$F(x_k)$	Δ_k
1	1	404	2
20	78	3.56	9.8×10^{-1}
40	140	0.75	1.2×10^{-1}
60	580	0.10	4.5×10^{-2}
80	786	0.0017	5.2×10^{-3}
✓ Stops with the new termination criterion			
100	1254	0.0019	2.8×10^{-4}
120	2003	0.0016	1.1×10^{-4}
✓ Stops with the termination criterion $\Delta_k \leq 10^{-4}$			

In Table 10, we further compare numerical results of UOBYQA, NOMADm and SPSA. We tested the Rosenbrock function of dimension 2 and 10. Increasing the dimension significantly increased computational burden. To solve a 10-dimensional problem, we need 20000 function evaluations in total. The different scales of noise variance σ^2 were specified to be 0.001, 0.01, 0.1 and 1. We constrained the algorithm with various total number of function evaluations. The parameter setting was the same as before, except

that we used different N_{max} when the total number of function evaluations or the variance changed. A practically useful formula for N_{max} was

$$N_{max} = \frac{\text{Max FN}}{\text{Iteration \# (n)}} \cdot \delta(\sigma^2).$$

Here the iteration # represented the estimated number of iterations that the algorithm should maintain for a given dimension. This is a rough estimate and the actual number of iterations varies in different problems. $\delta(\sigma^2)$ was an adjustment value associated with the variance σ^2 . The algorithm used a relatively smaller N_{max} when the variance was small. We provide the following suggested values in Table 9.

Table 9: Suggested values to define N_{max} .

n	2	4	7	10
Iteration #	50	200	550	1000
σ^2	0.001	0.01	0.1	1
$\delta(\sigma^2)$	2.5	3	3.5	4

As shown in Table 10, for the 2 dimensional case, our algorithms performed better than the other two algorithms. The mean value was the smallest among the peer algorithms. For the 10 dimensional case, when the variance σ^2 was large, our algorithm did not perform well. We should note a fact that making the variance 10 times larger, from our previous theoretical analysis, we potentially require 10 times more function evaluations in order to achieve the same scale of accuracy. That is why the quality of the

solutions decreased sharply as σ^2 increased. Another aspect that influences the performance may be the dimension of the problem. Powell has pointed out that in higher dimensional problems, UOBYQA may not be practically useful because the number of interpolation points L is huge. We think that the Noisy UOBYQA algorithm inherits the limitation in the same way.

Simulation Problems

In this subsection, we applied the noisy UOBYQA algorithm to solve a pricing problem via simulation. The parameters we considered were the prices of M similar goods in a store, say, p_1, p_2, \dots, p_M . When a customer arrives at the store, he is sequentially exposed to the M goods by the store keeper, from the most expensive good which is of the best quality to the cheapest good. He decides to purchase the item or not after viewing it, but he will buy at most one of them. For good i , we set the probability that the customer purchases the item as

$$Prob_i = \exp(-p_i/\eta_i), \quad i = 1, 2, \dots, M.$$

The objective of the store keeper is to determine the best combination of prices $\vec{p} = [p_1, p_2, \dots, p_M]$ which yields the largest expected profit. Running the simulation for a period of time, if we denote the total number of customers as m , and the number of customers who purchase good i as m_i , we formulate

Table 10: Apply the Noisy UOBYQA algorithm to the Rosenbrock function (results are based on 10 replications of the algorithms).

n	Noise level σ^2	Max FN	Noisy UOBYQA	NOMADm	SPSA
			Mean Error	Mean Error	Mean Error
2	0.001	200	0.14	0.61	0.42
	0.01	200	0.28	0.63	0.65
	0.1	200	0.44	0.88	0.59
	1	200	0.61	1.44	0.57
	0.001	500	0.099	0.44	0.38
	0.01	500	0.18	0.57	0.35
	0.1	500	0.32	0.77	0.43
	1	500	0.47	1.26	0.49
	0.001	1000	0.024	0.44	0.30
	0.01	1000	0.18	0.46	0.32
	0.1	1000	0.20	0.77	0.34
	1	1000	0.47	1.2	0.42
	0.001	5000	0.042	0.63	2.3
	0.01	5000	0.42	0.89	2.6
	0.1	5000	0.97	1.24	2.9
	1	5000	7.7	1.78	3.3
10	0.001	10000	0.033	0.54	2.3
	0.01	10000	0.15	0.88	2.9
	0.1	10000	0.77	1.2	3.6
	1	10000	7.2	1.66	3.9
	0.001	20000	0.022	0.32	2.1
	0.01	20000	0.12	0.45	2.6
	0.1	20000	0.53	0.50	3.4
	1	20000	5.8	1.1	3.3

the stochastic problem as:

$$\max_{\vec{p}} (\text{Expected Profit}), \text{ where Profit} := \sum_{i=1}^M \frac{m_i}{m} p_i.$$

The precise solution is obtained from:

$$\begin{aligned} \max_{\vec{p}} (\text{Expected Profit}) & \tag{3.59} \\ := \max_{\vec{p}} \sum_{i=1}^M & \left[\left(\prod_{j=1}^{i-1} (1 - Prob_j) \right) Prob_i p_i \right]. \end{aligned}$$

We used Arena 9.0 from Rockwell software to model the pricing system. We considered cases $M = 2$ and $M = 10$, where M also indicated the dimension of the variable \vec{p} . In computation of the $Prob_i$, the parameter η_i can reflect the goodness of the merchandize. We set $\eta_1 = 50$ and $\eta_2 = 20$ in the two-dimensional case; set $\eta_1 = 50, \eta_2 = 48, \dots, \eta_{10} = 32$ in the ten-dimensional case.

The Noisy UOBYQA algorithm and OptQuest (the optimization add-on for Arena, <www.opttek.com>) were compared on the stochastic simulation problem. (See results in Table 11.) We implemented the same parametric setting of the algorithm as in the Section 3.3.2, except that we assigned the initial trust region as 10, and the upper bound for replication usage as 200 and 2000 for the two cases respectively. By varying the number of customers generated in each simulation run, we can have several levels of noisy “profit value”. The table shows the different levels of variances of the output. We compared with the real optimal solutions solved using the same η_i ’s via (3.59).

As we can see in the table, the Noisy UOBYQA algorithm did a uniformly better job than OptQuest, with a higher quality of solutions. For all the tested cases, the gap to the optimal solution was reduced around 50%.

Table 11: Optimization results of the pricing model (over average value of 10 runs), where the real solution for $M = 2$ is 23.23 and for $M = 10$ is 68.28.

	Model	Estimated	Max	Noisy	OptQuest
	vari-	FN		UOBYQA	
	ance				
	0.0022	200		0.10	0.29
M=2	0.014	200		0.25	0.39
	1.1	200		0.45	1.05
	0.0098	2000		0.78	1.32
M=100	0.093	2000		0.89	1.54
	1.1	2000		1.47	2.98

Chapter 4

Applications

We first discuss two real-world applications of the two-phase simulation optimization procedure: the Wisconsin Breast Cancer Epidemiology model and microwave ablation antenna design. Since these projects were completed before the full WISOPT code was finished, many of the detailed steps in WISOPT were not carried out exactly. For example, we only applied the classification-based global search in Phase I in both examples and we used the surrogate model + the Nelder-Mead method in Phase II for the Epidemiology model. We will explain the two projects in details.

In addition, we apply the WISOPT code as detailed in Appendix A to solve the ambulance base problem that was already described in Section 2.2. This is a simulation problem in a higher dimension ($n = 10$) that has been provided by [82] as an example of a typical problem of interest to the simulation community. As an alternative approach, we apply the Noisy DIRECT as the Phase I method and implement the CRN in the simulation code.

4.1 The Wisconsin Breast Cancer Epidemiology Simulation

4.1.1 Introduction

The Wisconsin Breast Cancer Epidemiology Simulation uses detailed individual-woman level discrete event simulation of four processes (breast cancer natural history, detection, treatment and non-breast cancer mortality among US women) to replicate breast cancer incidence rates according to the Surveillance, Epidemiology, and End Results (SEER) Program data from 1975 to 2000. Incidence rates are calculated for four different stages of tumor growth, namely in-situ, localized, regional and distant; these correspond to increasing size and/or progression of the disease. Each run involves the simulation of 3 million women, and takes approximately 8 minutes to execute on a 1GHz Pentium machine with 1Gb of RAM.

The four simulated processes overlap in very complex ways, and thus it is very difficult to formulate analytical models of their interactions. However, each of them can be modelled by simulation; these models need to take into account the increase in efficiency of screening processes that has occurred since 1975, the changes in non-screen detection due to increased awareness

of the disease and a variety of other changes during that time. The simulations are grounded in mathematical and statistical models that are formulated using a parametrization. For example, the natural history process in the simulation can be modelled using a Gompertzian growth model that is parameterized by a mean and variance that is typically unknown exactly, but for which a range of reasonable values can be estimated. The overall simulation facilitates interaction between the various components, but it is extremely difficult to determine values for the parameters that ensure the simulation replicates known data patterns across the time period studied. In all there are 37 of these parameters, most of which interact with each other and are constrained by linear relationships. Further details can be found in [1, 39].

A score is calculated that measures how well the simulation output replicates an estimate of the incidence curves in each of the four growth stages. Using SEER and Wisconsin Cancer Reporting System (WCRS) data, we generate an envelope that captures the variation in the data that might naturally be expected in a population of the size we simulated (see Figure 25, the y-axis indicates the incidence rate per 100,000). For the 26 years in consideration, the four growth stages give a total of 104 points, each of which is tested to see if it lies in the envelope. The number of points outside the envelope is summed to give the score (0 is ideal). While it could be argued that distance to the envelope might be a better measure, such calculations are

scale dependent and were not investigated. Unfortunately, the score function also depends on the “history” of breast cancer incidence and mortality that is generated in the simulation based on a random seed value $\xi(\omega)$. We will adopt the notation $F(x, \xi(\omega))$ where x represents the vector of parameters, and $\xi(\omega)$ indexes the replication. While we are interested in the distribution (over $\xi(\omega)$) of $F(x, \xi(\omega))$, we will focus here on the problem:

$$\min_x \max_{\xi(\omega)} F(x, \xi(\omega)),$$

or alternatively, in terms of average

$$\min_x \mathbb{E}[F(x, \xi(\omega))].$$

4.1.2 Methods and Results

The purpose of this study is to determine parameter values x that generate small values for the scoring function. Prior to the work described here, acceptance sampling had been used to fit the parameters. Essentially, the simulation was run tens of thousands of times with randomly chosen inputs to determine a set of good values. With over 450,000 simulations, only 363 were found that had a score no more than 10. That is, for a single replication ξ_1 , 363 vectors x had $F(x, \xi_1) \leq 10$.

Our first goal was to generate many more vectors x with scores no more than 10. To do this, we attempted to use the classification-based global search (Procedure 3 in Chapter 2) based on the scoring function data. The

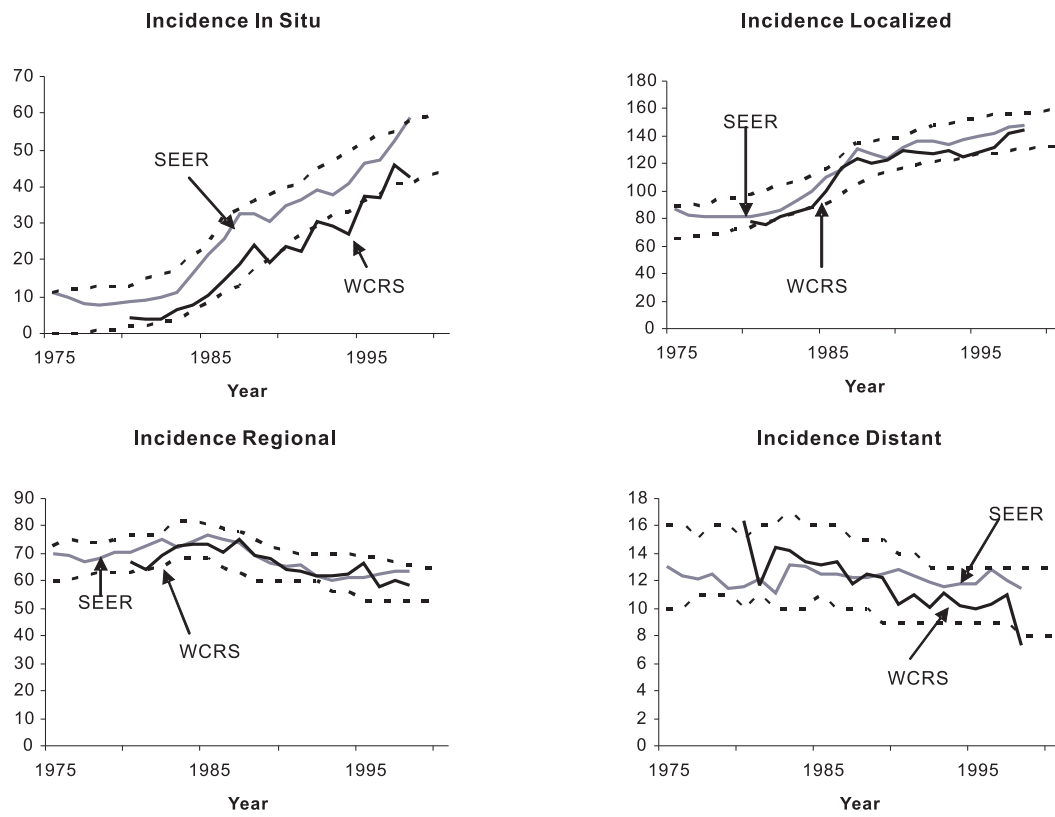


Figure 25: An envelop captures the variation of the incident curves in the four stages.

level set we considered is

$$L(c) = \{x | F(x, \xi_1) \leq c\}, \text{ for a fixed replication } \xi_1.$$

We typically use $c = 5$ to indicate good fit and $c = 10$ for acceptable parameter choices.

Since we have a vast majority of negative samples, the data-preprocessing step was applied to yield a much balanced training data set. A great portion of the negative samples were removed, resulting a training set containing all the 363 positive samples and 500 negative samples. We trained an ensemble of classifiers that predicted membership of $L(c)$. Each resulting classifier was evaluated on the testing set using the measures

$$TP = \frac{\# \text{ correctly classified positives}}{\text{total } \# \text{ of positives}}$$

and $TN = \frac{\# \text{ correctly classified negatives}}{\text{total } \# \text{ of negatives}}$

Classifiers were discarded if the value of TP was less than 0.9 (TN typically is around 0.4). The value was chosen to guarantee the probability of removing positive points in error is small. 100,000 potential values for x were uniformly generated in the feasible domain. Each of the classifiers selected was used to determine if the point x was negative (and hence removed from consideration). At that stage, there were 220 points that were hypothesized to be positive. Evaluating these points via simulation, 195 were found to be in $L(10)$. Thus, with very high success rate (89%), our classification-based global search is able to predict values of x that has a lower score $F(x, \xi_1)$.

Since the classifiers are cheap to evaluate, this process facilitates a more efficient exploration of the parameter space. Clearly, instead of using a single replication ξ_1 , we could instead replace $F(x, \xi_1)$ by $\max_{i=1}^N F(x, \xi_i)$ for some $\xi_1, \xi_2, \dots, \xi_N$ and $N > 1$. In fact this was carried out. The difficulty is that we require replication data (for our experiments we choose $N = 10$) and we update the definition of $L(c)$ appropriately. However, the process we follow is identical to that outlined here.

In our setting, x has dimension 37. Using expert advice, we only allowed 9 dimensions to change; the other 28 values were fixed to the feasible values that have highest frequency of occurrence over the positive samples. For example, if x_{37} can take possible values from $[\phi_1, \phi_2, \dots, \phi_n]$, then we set the value of x_{37} to be $\arg \min_{i=1}^n \frac{P_i}{W_i}$, where P_i and W_i are the number of appearances of ϕ_i in the positive and whole sample set. This is similar to using a naive Bayesian classifier to determine which value has the highest likelihood to be positive. Our experiments showed this choice of values outperformed even the values that experts deemed appropriate for these 28 values; a posteriori analysis confirmed their superiority.

In Phase II local search, we employed the sample-path method which fixes the number of replications $N = 10$ and derives the minimizers $x^{*,10}$. Since we carried out multiple local optimizations, the best $x^{*,10}$ is treated as an approximate solution to the underlying solution x^* . The evaluation process of a single simulation run is expensive, therefore we performed the optimization

tasks on a surrogate model (Section 1.1.1), which was constructed using a set of samples.

1558 samples were selected for further evaluations with replications. In this, we included the original 363 positive samples, another 195 positive samples selected by the classifiers and 1000 negative samples from the original data set. The 1000 negative samples were chosen such that their original scores were less than or equal to 30. In this research, we fixed the other 28 parameters using the ‘optimal setting’ we calculated. Each parameter sample was evaluated 10 times for the real function $F(x, \xi_i), i = 1, 2, \dots, 10$. Since we used a different setting for the other 28 parameters, the new results were a little different from the first replication, but the scores were generally better. We found that 310 out of the 1558 samples had maximum score less than 10.

The scores varied from replication to replication. We intended to solve the problem of minimizing $g(x)$, where $g(x)$ is some combination of the multiple score values. We either let $g(x) = \max_{i=1}^n F(x, \xi_i)$ or let $g(x) = \hat{f}^N(x) = \frac{\sum_{i=1}^n F(x, \xi_i)}{N}$ to serve this purpose. Since the lowest possible score is 0, minimizing $g(x) = \max_{i=1}^N F(x, \xi_i)$ is equivalent to reducing the upper bound. Thus, minimizing $g(x)$ also controls the distribution of function values. Our results showed that there were only small differences among the different choices of $g(x)$. In subsequent work, we primarily used the objective function $g(x) = \max_{i=1}^N F(x, \xi_i)$.

Given the 10 replications of each sample, we used the DACE toolbox [72] to fit a Kriging model $\hat{g}(x)$ to the data, which we considered as a surrogate function for our objective function $g(x)$. We used the Nelder-Mead simplex method (a derivative-free method) to optimize the surrogate function and generated several local minimizers based on different trial starting points. It is worth noting that: the first 5 parameters can be regarded as continuous variables. The following 4 parameters: ‘aggr4Node’, ‘aggr5Node’, ‘lag’, ‘LM-PRgress’ can only take on limited values. Actually, we found a total of 108 combinations for these 4 parameters. We implemented a simple branching technique when searching for the minimizers of the surrogate function. The min problem was split into 108 small sub-problems over the 5 continuous variables, where in each sub-problem the 4 parameters are fixed to one setting. The solution of the minimization problem is the best solution of the 108 sub-problems.

These local minimizers were evaluated by simulation. To improve our results further, we updated the surrogate function with the simulation results of the local minimizers and repeated the optimization. The parameter values found using this process outperform all previous values found. Our best parameter generated a score distribution with a mode of 2. Furthermore, expert analysis of various output curves generated from the simulation results with the best set of parameter values confirms the quality of this solution. All the results showed that the surrogate function using the DACE toolbox

performs well.

At the end, we summarize several evident conclusions:

- The classifier technique is cheap to use and predicts good parameter values very accurately without performing additional simulations.
- An ensemble of classifiers significantly improves classification accuracy.
- Imbalanced training data has a detrimental effect on classifier behavior. Ensuring the data is balanced in size is crucial before generating classifiers.

4.2 The Coaxial Antenna Design in Microwave Ablation

4.2.1 Introduction

Microwave ablation for the treatment of hepatic and metastatic tumors is a promising alternative when surgical resection – the gold standard – is not practical. In this procedure, a thin, coaxial antenna (probe) is inserted into the tumor (either percutaneously or during open surgery) and microwaves are radiated into the tissue. The alternating fields cause rapid rotation of the polar water molecules resulting in heating of tissue and ultimately leading to cell death. This cell death is a function of both temperature and time, where

higher temperatures lead to cell death in a shorter period of time. Since studies indicate that coagulated necrosis of tissue can be achieved within a few seconds at 60 °C, a common metric to predict cell death and ultimately lesion size is the 60 °C contour. Since this 60 °C metric is valid for both cancerous and normal tissue, design of the antenna radiation pattern is critical to achieve a heating pattern affecting only cancerous tissue. To meet these design needs, several types of coaxial antennas have been proposed and optimized for this application and are reviewed in [5].

When optimizing a design or studying performance, most studies use average dielectric properties for liver tissue that have been measured previously and are readily available in the literature [44]. However, due to the natural variation in tissue among individuals, measured dielectric properties of healthy and tumorous liver tissue may differ by as much as $\pm 10\%$ from the average value [105]. This variation means that a given patient's tissue may not have the same dielectric properties as those used in the design of an antenna, leading to suboptimal performance. Therefore it is important to ensure that antennas to be used for hepatic microwave ablation are robust, i.e., relatively insensitive to changes in physical properties of the tissue.

We present a method for optimizing a coaxial antenna for microwave ablation of hepatic and metastatic tumors that takes into account the variability in liver dielectric properties among individuals. This adds additional

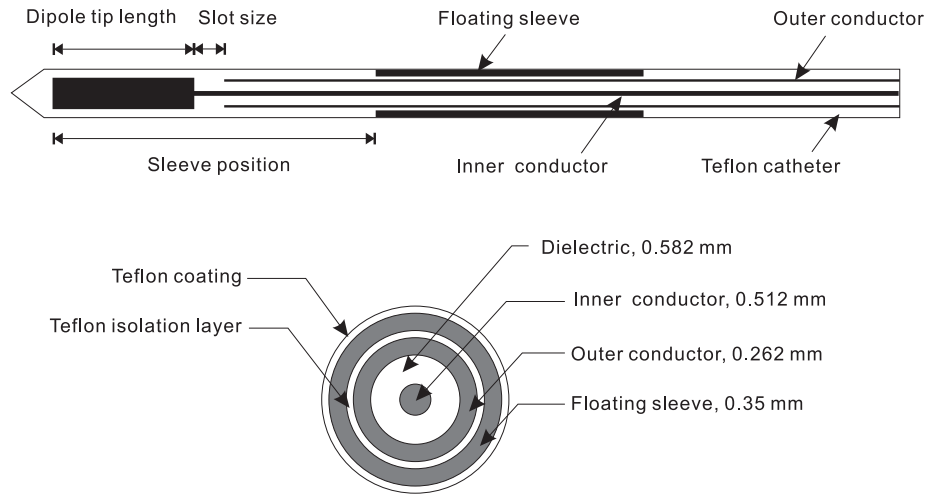


Figure 26: Structure of the floating sleeve antenna. Wall thicknesses of fixed dimensions are labeled in the figure.

complexity to the optimization problem since the performance of a particular design is now a function of its dimensions (the design variables), as well as some unknown variation in tissue properties. To design an antenna with robust performance, we apply the two-phase optimization procedure in Section 1.3. The two-phase approach addresses distinct goals: global and local, in the optimization process.

To perform this optimization we rely upon computer models. Computer models are a widely used tool in the design of antennas for microwave ablation as they provide a quick, convenient and accurate method of estimating antenna performance. Given the physical properties of liver tissue and tumor, such models can be used to predict the EM field distribution and resistive

heating in tissue due to a particular antenna design. A quantitative assessment of a particular antenna design may then be obtained by extracting appropriate metrics from solutions provided by the computer model. Suitable metrics may be the efficiency of the antenna (fraction of the power supplied that is deposited into the liver), size and shape of the predicted lesion compared to the tumor, and diameter of the antenna. An optimization problem may then be formulated where the design variables are the dimensions of the antenna, and the objective function is obtained by combining the metrics in some fashion [57]. The floating sleeve antenna presented in [112] was chosen to be optimized in this study due to its ability to create large, constrained lesions. A schematic of the antenna is shown in Figure 26. We have identified dimensions of this antenna that may be optimized to yield desirable lesion size, shape and efficiency, and we minimize the overall diameter of the probe.

4.2.2 Methods

Floating sleeve antenna

Yang et al [112] presented a floating sleeve antenna consisting of a coaxial dipole antenna with a floating sleeve used to constrain power deposition to the distal end. The structure of the antenna is shown in Figure 26. For a particular operating frequency (usually 915 MHz or 2.45 GHz), dimensions of the antenna that affect the radiation pattern and efficiency of the antenna

Table 12: Design metrics in the sleeve antenna.

Parameter	Range of values
Length of dipole tip	1 – 60 mm
Slot size	1 – 50 mm
Sleeve position	1 – 60 mm
Thickness of Teflon coating	0.1 – 1 mm
Thickness of Teflon isolation layer	0.1 – 1 mm
Length of sleeve	5 – 50 mm

are: (a) length of the dipole tip, (b) slot size, (c) sleeve position, (d) thickness of Teflon isolation layer, (e) thickness of Teflon coating and (f) sleeve length. Throughout this section, an individual design, $x \in \mathbb{R}^6$, may be expressed as the vector of the design variables (a)–(f), in mm. As explained in [112], the floating sleeve is effective in constraining the lesion longitudinally when it is approximately a half wavelength long. Note that this is not half the wavelength of a plane wave propagating through either liver or the Teflon of the catheter. Rather, this is half a wavelength in the layered Teflon/liver medium (outside the metal sleeve) whose effective properties are somewhere between those of liver and Teflon and is a function of the Teflon thickness. Therefore since we are varying the thickness of the Teflon layers we expect the optimal sleeve length to change as well. Table 12 shows the range over which of the dimensions (a)–(f) were varied.

Finite element model of coaxial sleeve antenna

For this study we used the commercial *finite element* (FE) package, COMSOL Multiphysics v3.2 (COMSOL Inc. Burlington, MA) to simulate antenna performance and determine the objective function for a given antenna design. This software allows us to specify the geometry of an antenna design and then solves Maxwell's equations in the surrounding tissue. We coupled this software with MATLAB, to perform the optimization of the antenna.

The model involves the antenna inserted into an infinitely large piece of liver. Dimensions of the antenna as well as design variables are illustrated in Figure 26. Input power was set to 120 W at an operating frequency of 2.45 GHz. Due to the cylindrical symmetry of the geometry, we were able to reduce computational burden by implementing an axially symmetric model. A steady state nonlinear solver was used to compute the resistive heating ($Q(\mathbf{r})$) which is proportional to the square of the local electric field. Computation time for each simulation was approximately 11 s on a computer with a 3 GHz Intel P4 processor and 1 GB memory.

Typically, constant values of dielectric properties are assumed when designing an antenna. Often these properties are obtained from average values reported in the literature for healthy human liver [29, 44, 105]. However there is a natural variation in tissue among individuals. [105] have measured dielectric properties of healthy and tumorous ex-vivo human liver at room temperature in the 0.3–3 GHz range. Their results show standard deviations

of $\pm 10\%$ in samples taken from different individuals. Also, it is expected that the dielectric properties of this tissue will vary during the course of ablation as tissue water content and temperature change from their steady state values. As such, it is important to ensure that antennas used for microwave ablation are robust, i.e., relatively insensitive to changes in the physical properties of tissue. In this study, we used average values of dielectric constant (43.03) and conductivity (1.69), as in [112] and assumed these dielectric properties vary randomly as a Gaussian distribution with standard deviation $\pm 10\%$ about this mean. Thus, the dielectric constant and conductivity may be expressed as $\mathcal{N}(43.03, 4.303^2)$ and $\mathcal{N}(1.69, 0.169^2)$. While we do not account for changes in dielectric properties during the course of ablation, because the antenna is more robust to variations in tissue properties, it should lead to a better performance with respect to dielectric property changes during the course of ablation. Moreover, it may be possible to utilize this noisy optimization algorithm to take into account these changes in future studies.

The resistive heating ($Q(\mathbf{r})$) profiles calculated by the FE model are used as input to a thermal model which predicts temperature profiles from which an estimate of lesion size is obtained. For computational efficiency, we have chosen to use a simple thermal model

$$Q(\mathbf{r}) = \rho c \frac{dT(\mathbf{r})}{dt}, \quad (4.1)$$

where c [J/(kg·K)] and ρ [kg/m³] are the specific heat capacity and density

of liver tissue, respectively, and dT/dt [K/s] is the change in local temperature with time. Here, for computational efficiency, we have ignored thermal conductivity and the (cooling) effects of blood perfusion. The lesion size and shape metrics are calculated using the 60 °C contour after 180 s with an input power of 120 W.

Objective metrics for assessing antenna performance

In this study we are optimizing for lesion size and shape, antenna efficiency and antenna size. In practice, design variables may be selected to fit the heating pattern of an antenna to the tumor at hand. Since most tumors are spherical in shape [112], our goal in this study was to optimize an antenna to yield the lesion with largest radius and having a shape that is as close to a sphere as possible. We identified two metrics to assess the size and shape of the lesion: lesion radius and axial ratio. These metrics are illustrated for an example $Q(\mathbf{r})$ profile in Figure 27. Note that an axial ratio (as annotated in Figure 27) of 0.5 would yield a spherical lesion shape. The efficiency of an antenna may be measured by computing the reflection coefficient ($S_{11}dB$) – the ratio of power reflected to power input. The more negative the S_{11} , the more power is coupled into the liver. Reflected power was calculated from the FE model by sampling the net time-averaged power flow at the antenna feedpoint and subtracting from the input power (120 W). Finally, the antenna being optimized may be used in a minimally invasive procedure

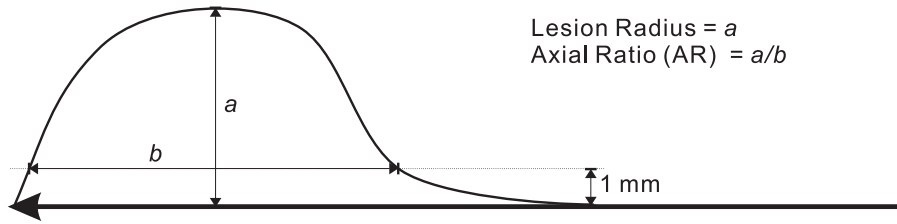


Figure 27: Objective metrics for assessing size and shape of a $Q(\mathbf{r})$ profile

Table 13: Objective metrics in the sleeve antenna design.

Metric	Measure of	Goal
Lesion radius	Size of lesion in radial direction	Maximize
Axial ratio	Proximity of lesion shape to a sphere	Fit to 0.5 (see figure 27)
S_{11}	Efficiency of antenna	Minimize (or maximize $ S_{11} $)
Probe radius	Radial size	Minimize

(i.e., percutaneously); thus, it is desirable to yield a design with the smallest radius. These objective metrics are summarized in Table 13.

We employ an algorithm that only handles a single objective function and so the above four objectives need to be combined. A simple way to do this is to assign weights to each metric, based on their relative importance, and then sum up the weighted objectives. Since the range over which these objectives vary is not the same, we normalized each objective by their largest possible value so that the weighted sum is not skewed by the scale of each individual metric. Each metric was deemed to be equally important and so identical weights of 0.25 were assigned to each normalized objective. The

optimization problem thus formulated is written as:

$$\begin{aligned}
& \min_{x \in \mathbb{R}^6} f(x) \\
& := \mathbb{E} [F(x, \xi(\omega))] \\
& = \mathbb{E} \left[-p_1 \frac{\text{Lesion}(\xi(\omega))}{H_1} + p_2 \frac{|\text{AR}(\xi(\omega)) - 0.5|}{H_2} + \right. \\
& \quad \left. p_3 \frac{S_{11}(\xi(\omega))}{H_3} + p_4 \frac{\text{Probe}(\xi(\omega))}{H_4} \right], \quad (4.2)
\end{aligned}$$

where the weights $p_1 = p_2 = p_3 = p_4 = 0.25$ and H_i represent normalization values. The function $F(x, \xi(\omega))$ is often called the sample response function. The optimization formulation (4.2) aims to maximize the expected performance among different individuals, whose specific physical parameters $\xi(\omega)$ are extracted from predefined distributions. (In our case $\xi(\omega)$ indicates the dielectric properties.) Besides (4.2), other robust formulations such as $\min_{x \in \mathbb{R}^n} \max_{i=1}^N F(x, \xi_i)$ and $\min_{x \in \mathbb{R}^n} \min_{i=1}^N F(x, \xi_i)$ are possible, but these are not discussed here.

4.2.3 Results

Calibrating the antenna design metrics

We present a step by step explanation of the application of the two-phase framework (Section 1.3) in antenna design, aiming to minimize the performance measure f in (4.2). In Phase I, to prepare the training set for classifiers, we used the uniform LHS to generate 2,000 design samples and later

evaluated them with one replication of the FE model. At this stage, the combined objective values f were found to have a skewed distribution in the range $[-0.3705, 3597]$. Roughly around 20% samples yielded objective values in $[0, 3597]$, while the remaining 80% of the objective values covered the range $[-0.3705, 0]$. For a clear view, we only plotted the histogram of the objective values over the interval $[-0.3705, 0]$ (see Figure 28(a)). Note that there were no points in the range $[-0.1, 0]$, which showed that the 20% bad set may be outliers of the simulation. But we still include them in the training set as negative samples.

We set the level set parameter c to be -0.2765 , which was the 10% quantile of the objective values. The level set

$$L(c) = \{F(x, \xi_1) \leq -0.2765\}$$

thus defined contained 199 positive samples and the remaining 1801 samples ($\notin L(c)$) were labelled as negative. This is an ill-balanced training set. Thereby, we applied the one-sided under-sampling method (Procedure 2 in Section 2.1) to reduce the number of negative samples to 388. The method first applies the 1-NN rule to obtain a consistent training set and then removes of Tomek links. By duplicating the positive samples, we ended up with a much more balanced set with 398 positive vs. 388 negative samples. The ratio of positive to negative was mitigated greatly from 1:9 to 1:1.

Before assembling classifiers, the 6 candidate classifiers were tested on the training set. We employed the criterion g-mean $g \geq 0.5$ to examine the

accuracy of each classifier. After training and testing the classifiers on two separate subsets respectively (Procedure 1 in Section 2.1), the observed g-means were: SVM linear kernel 0.8149, SVM quadratic kernel 0.4852, SVM cubic kernel 0.7661, SVM Gaussian kernel 0.8306, C4.5 0.6532 and 5-NN 0.7549. Only the SVM with quadratic kernel did not meet our criterion and was therefore dropped from our ensemble. The other 5 classifiers were included to our ensemble and trained with the full data set.

For vast alternative designs, we again used the uniform LHS to generate a number of much more refined data: 15,000 designs. We felt the distances among the designs were already sufficiently small and suitable for the local search in Phase II. After the prediction of the ensemble of classifiers, only 522 out of the 15,000 designs were classified as positive. These newly discovered positive samples were further evaluated with our FE model. This time, 74% designs were found to be correctly predicted and located in the level set $L(c)$. The best design we obtained yielded objective value -0.3850. As a comparison, the histogram of the second stage data over the range $[-0.3850, 0]$ was plotted in Figure 28(b). We can clearly see that our newly identified designs were much better than the previous training set, implying that we indeed sampled over a superior region. The mode of the distribution improved from -0.17 to -0.31.

The multistart optimization using the phase transition procedure generated the set \mathcal{I} of 10 initial points, with corresponding initial trust region

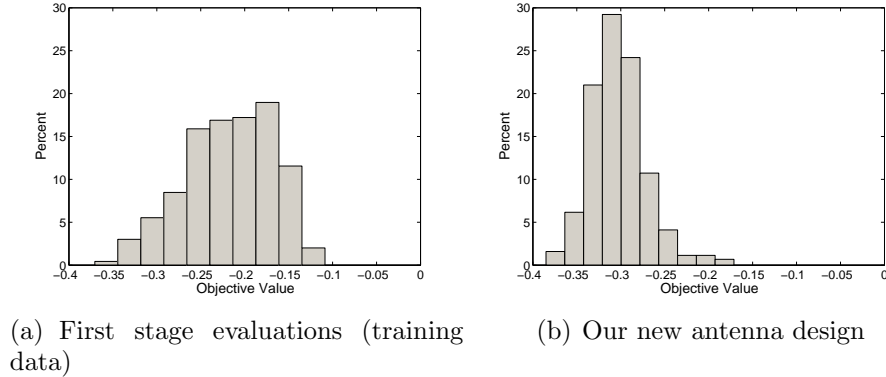


Figure 28: Histograms of the objective values

radii.

In Phase II local optimizations, we employed the VN-SP-UOBYQA algorithm because CRN can be implemented in the FE model. As a general setting for the algorithm, we set the initial number of samples $N_0 = 3$, which was used to estimate the initial sample mean and sample covariance matrix, and set a predefined sequence:

$$\alpha_k = 0.1 \times (0.98)^k. \quad (4.3)$$

This sequence satisfies the assumptions required in the convergence theory [27]. Other choices (instead of 0.1 and 0.98) are clearly possible, but we found these values to work well in this application setting. Future work will determine an automatic scheme to set these values. We limited the maximum number of function evaluations to 2000, therefore, it took roughly

6 h for the entire optimization process. We chose the initial trust region radius Δ_0 to be 1, which corresponded to a 1 mm local search region centered around the initial design x_0 . The normalization values in (4.2) were $H_1 = 3$, $H_2 = 0.5$, $H_3 = 30$, and $H_4 = 0.5$, which correspond to maximum values expected for each of the individual metrics. The VN-SP-UOBYQA algorithm was applied 10 times using each starting point in \mathcal{I} . At the end of Phase II, the best design out of the 10 local optimums is (9.67 1.62 19.53 0.42 0.12 16.11).

Discussion

While comparison of the compound objective function shows that the optimization procedure helped yield an improved design $x^* = (9.67 \ 1.62 \ 19.53 \ 0.42 \ 0.12 \ 16.11)$, it is important to confirm that improved performance was achieved in terms of the individual metrics. The goal of the optimization process was twofold: (a) improve robustness of the design so that each individual metric is less sensitive to variations in tissue dielectric parameters among individuals and (b) to improve the values of each of the individual metrics. Figure 29 shows the distribution for each of the individual metrics of the optimal design and the design in [112] for a common random sample of 100 different values for the dielectric properties within the $\pm 10\%$ specified range. Also included are the distributions of design presented in [112].

The antenna presented by [112] has the following objective metrics: lesion radius = 1.92 cm, axial ratio = 0.42, S_{11} = -15.9 dB and probe radius = 0.17 cm. In comparison, the optimal design presented in this work has objective metrics: lesion radius = 2.03 cm, axial ratio = 0.48, S_{11} = -17.8 dB and probe radius = 0.20 cm. Table 15 provides the mean and standard deviation for each of the individual metrics.

These results indicate the two-phase procedure yields a design which has improved values for the compound objective and each of the individual objective metrics when compared to the original design of the sleeve antenna, except for probe radius. In particular, the variation in the axial ratio and reflection coefficient of the optimal design are 24% and 90%, respectively, lower than the original design. Although the average value of the lesion radius of the optimal design is larger than that of the original design, there is a 15% increase in standard deviation. If this is a potential issue, we could augment our objective function to incorporate this feature; for example, by introducing an additional term that limits the standard deviation. The probe radius is a function of the physical dimensions of the antenna and is thus independent of any variations in tissue properties, which explains its variance of 0 in Table 15.

Figure 30 shows $Q(\mathbf{r})$ heating profiles of Yang et al's design compared to the optimal design presented here when simulating using the average values for dielectric parameters. The optimal design not only creates a larger

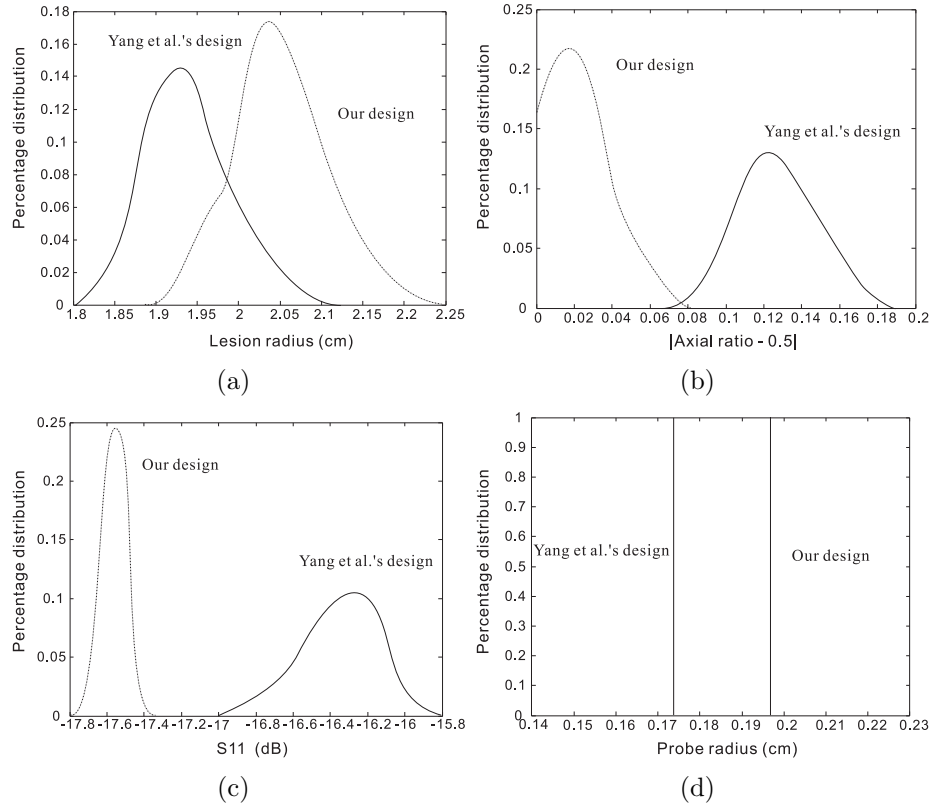


Figure 29: Variations of individual objective metrics (frequency plot). (a) Lesion radius, (b) $|\text{Axial ratio} - 0.5|$, (c) S_{11} , (d) Probe radius

Table 14: Comparison of dimensions of the original and optimized antennas.

Parameter	Design by Yang et al.	Our optimal design
Length of dipole tip	9.00 mm	9.67 mm
Slot size	2.00 mm	1.62 mm
Sleeve position	20.00 mm	19.53 mm
Thickness of Teflon coating	0.15 mm	0.42 mm
Thickness of Teflon isolation layer	0.15 mm	0.12 mm
Length of sleeve	16.00 mm	16.11 mm

Table 15: Comparison of the objective metrics of the original and optimized antennas.

	Design by Yang et al		Our optimal design	
	Mean	Std	Mean	Std
Lesion radius	1.9190	0.0628	2.0313	0.0727
Axial ratio - 0.5	0.0785	0.0311	0.0239	0.0237
S_{11}	-15.89	0.2751	-17.75	0.0268
Probe radius	0.1730	0	0.1970	0

lesion, but also does better in constraining the lesion to the distal end of the antenna. The antenna designed in [112] has a $Q(\mathbf{r})$ profile with a ‘tail’ along the axis of the antenna, as is clear from Figure 30(a). Note that for both antennas, the actual longitudinal extent of the lesion after taking into account thermal conductivity is likely to be even larger, thereby potentially destroying large amounts of healthy tissue. While it is plausible that the optimal design may also exhibit a ‘tail’ in the lesion after considering effects of thermal conductivity, it is likely to be much smaller since the amount of electromagnetic power deposited along the axis of the coaxial cable is much smaller.

The probe radius of the optimal design is slightly larger than that in [112]. However, both designs are well within the range of probes typically used in percutaneous applications. If a smaller probe radius is desirable, the weight associated with this metric (w_4) may be emphasized. There is also a slight improvement in reflection coefficient, although the difference in the actual

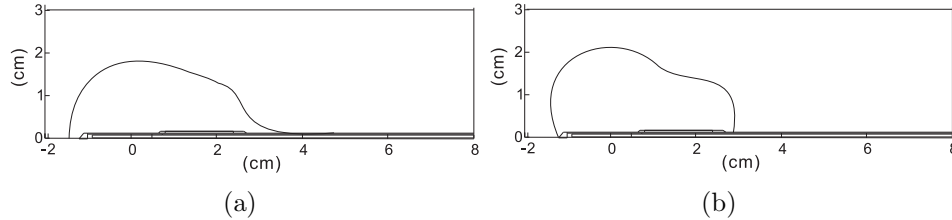


Figure 30: Comparing simulated $Q(\mathbf{r})$ profiles using the average dielectric properties (43.03,1.69). (a) antenna design presented by Yang et al (b) optimal design derived by our algorithm

amount of power reflected is negligible.

The optimized antenna has a $Q(\mathbf{r})$ profile with axial ratio closer to 0.5 (which is the axial ratio a perfectly spherical lesion would yield) than Yang et al’s design. Notice that the shape of the $Q(\mathbf{r})$ profile is not perfectly spherical. This is because the metric employed only considers the extents of the lesion in the longitudinal and radial directions. An improved metric that analyzes the shape of the lesion along its entire boundary may help yield even better designs.

4.2.4 Conclusions

We have optimized the design of a floating sleeve antenna for microwave ablation of hepatic tumors using the two-phase optimization framework. This was done by identifying desirable features for a coaxial antenna for this application and formulating a mathematical optimization problem to select the dimensions of the floating sleeve antenna that optimize these features.

We accounted for the natural variation in physical properties of liver tumors/tissue among individuals by incorporating a stochastic component into the optimization problem.

We achieved a substantial improvement in the variation in two of the three objective metrics (axial ratio and S_{11}) as a function of dielectric properties of the tissue, although variation in lesion radius slightly increased. Mean values of all the objective metrics of the optimum design except the probe radius were superior to the original design of the sleeve antenna.

While our method yields an improved design, several aspects of the procedure may be enhanced to yield even better performance. The metric we have used for assessing lesion shape only utilizes knowledge of the maximal extents in the radial and longitudinal directions. An improved metric would analyze features along the entire lesion boundary. In order to ease computational burden, we have neglected effects of thermal conductivity. These effects may be included to improve the accuracy of the finite element model. Lastly, availability of more measurements of dielectric properties of human liver tumors would allow for better modeling in the variation of these properties.

4.3 Ambulance Base Problem

The ambulance base problem aims to determine the optimal locations of ambulance bases such that the averaged response time of an ambulance to an emergency call is minimized. A detailed description of the problem can be found in Section 2.2. As a specific example, we consider 5 ambulance bases, whose locations are $p(i), i = 1, 2, \dots, 5$, in the region $[0, 1]^2$ (therefore, the problem is a 10 dimensional optimization problem). We assume that the emergency calls arrive following a Poisson process and the positions of the calls follow a joint triangle distribution $g(x_1, x_2) = \hat{g}(x_1)\hat{g}(x_2)$, where $\hat{g}(x)$ is a one-dimensional triangle distribution $\hat{g}_{0,1,0.8}(x)$. As an illustration, we plot in Figure 33 a distribution of simulated call locations, with a mode at $[0.8, 0.8]$. The other simulation parameters used were the same as that presented in the former example; for example, we used a simulation time of 500 hours, arrival rate $\lambda_a = 0.5$ and the speed of the ambulance $v = 0.5$.

Instead of using 20,000 simulation runs to derive an exact solution using Noisy DIRECT, we took the two-phase approach and scheduled 10,000 simulation evaluations in Phase I and another 10,000 simulation evaluations in Phase II local optimizations. Other options of the Noisy DIRECT were the default values (listed in the appendix). Table 16 shows the iterations of the Noisy DIRECT algorithm.

At the end of phase I, we observe that a total of 431 possible points were

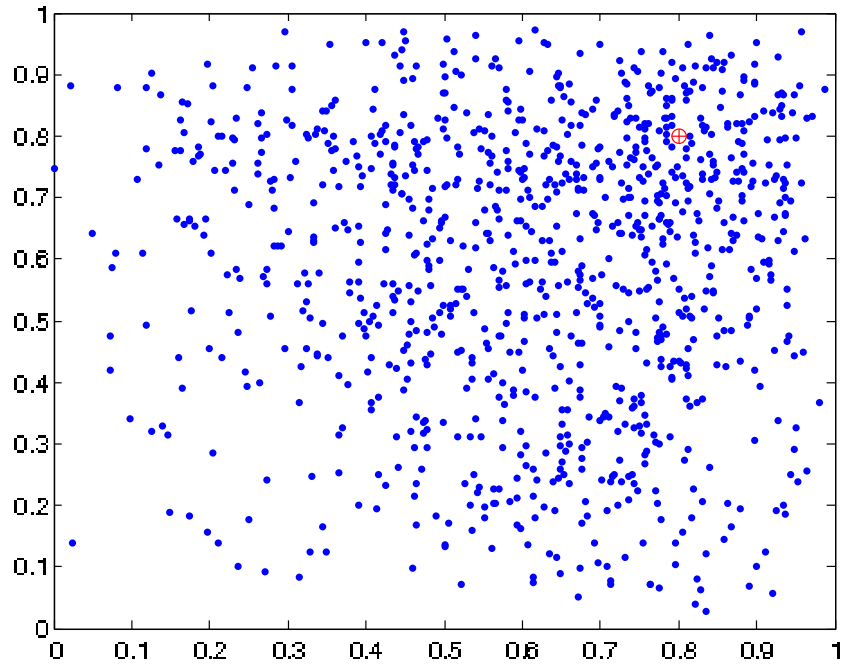


Figure 31: A distribution of emergency calls

Table 16: The output of Noisy DIRECT in the ambulance base problem

Iteration (k)	$\bar{F}(x_k)$	#Point	# Function Evals
1	0.5305	21	63
2	0.4490	39	117
3	0.4254	73	219
4	0.4208	103	1496
5	0.3715	145	2756
6	0.3715	185	3588
7	0.3715	223	4296
8	0.3640	319	6794
9	0.3505	431	10345

considered. Counting replications at the points, the algorithm used 10345 total function evaluations. The best solution in Phase I yields an objective value 0.3505. The 431 points were passed to the phase transition module, and we obtained 10 starting points for Phase II local optimizations. Since the 5 positions of ambulance bases were arranged in the variable x , switching the sequence of points is considered to be different local solutions. Indeed, when we plot the positions of the 10 starting points $x_0^i, i = 1, 2, \dots, 10$, we notice that it only has two types of patterns involved (Figure 32). Both patterns are roughly symmetric about the diagonal but the orientations are different. Future work will investigate how to deal with symmetric solutions of this nature.

Since we were able to set the random seed in each simulation run, the VN-SP-UOBYQA was applied in Phase II for the CRN case. For each run of the algorithm, we constrained a maximum 1,000 function evaluations. The best solution is plotted in Figure 32 and it yielded an objective value 0.3234. As we have explained, the output is a sample average of the replicated outputs and is biased by noise. To test the quality of this solution, we plugged it into the same refined model (in Section 2.2) which uses a much longer simulation time. The result turned out to be 0.3301, slightly better than the result computed in Section 2.2 and closer to the computed value of the global minimum.

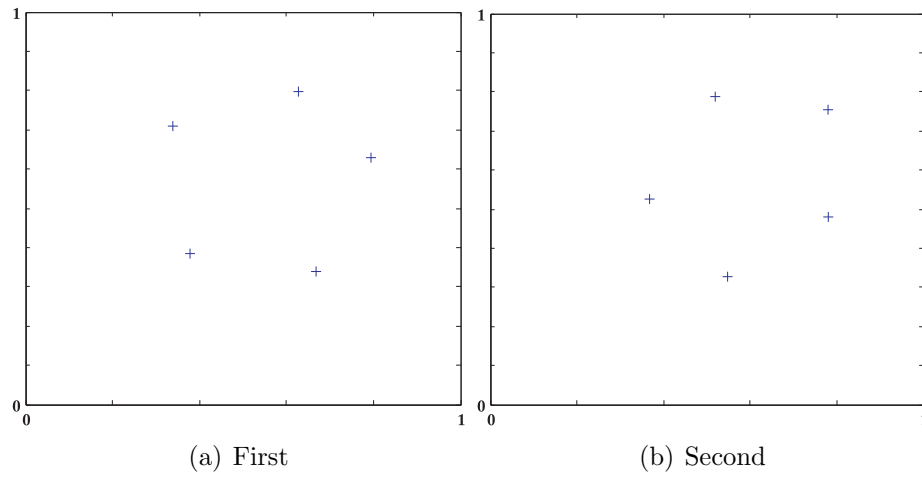


Figure 32: Two possible patterns of ambulance bases

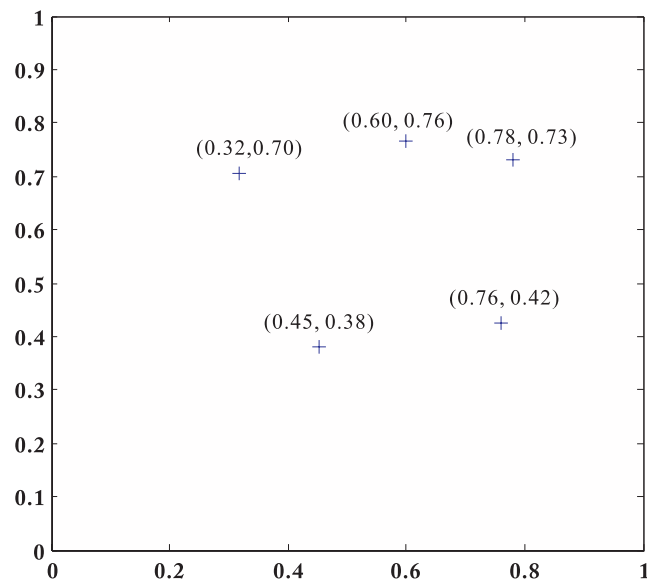


Figure 33: Positions of the ambulance bases.

Chapter 5

Monte Carlo Simulation

Efficiency in Neuro-Dynamic

Programming

Monte Carlo simulation plays an important role in *neuro-dynamic programming* (NDP)/*reinforcement learning* [9]. NDP is a class of reinforcement learning methods that approximate the optimal cost-to-go function. The idea of an approximate cost function helps NDP avoid the curse of dimensionality and distinguishes the NDP methods from earlier approximation versions of dynamic programming (DP) methods. Sub-optimal DP solutions are obtained at significantly smaller computational cost.

Rollout algorithms in NDP may approximate the cost-to-go function by simulation using a heuristic policy. At each stage, a policy improvement is calculated using the simulation approximation to estimate future costs. In this chapter, we discuss how to improve Monte Carlo simulation efficiency in deriving rollout policies/controls for stochastic optimal control problems.

‘Efficiency’ explicitly refers to consuming the least simulation effort (time) while obtaining accurate rollout policies. We perform Bayesian analysis for Monte Carlo simulation output, and accordingly we are able to 1) evaluate the accuracy of selected policies and 2) optimally allocate additional simulation evaluations to improve the accuracy. A small scale fractionated radiotherapy problem is presented to illustrate the advantages of our simulation allocation procedures.

5.1 Introduction

Every year, nearly 500,000 patients in the US are treated with external beam radiation, the most common form of radiation therapy. Before receiving irradiation, the patient is imaged using computed tomography (CT) or magnetic resonance imaging (MRI). The physician contours the tumor and surrounding critical structures on these images and prescribes a dose of radiation to be delivered to the tumor. *Intensity-Modulated Radiotherapy* (IMRT) is one of the most powerful tools to deliver conformal dose to a tumor target [11, 111, 93]. The treatment process involves optimization over specific parameters, such as angle selection and (pencil) beam weights [33, 34, 79, 102]. The organs near the tumor will inevitably receive radiation as well; the physician places constraints on how much radiation each organ should receive. The dose is then delivered by radiotherapy devices, typically in a fractionated regime

consisting of five doses per week for a period of 4–9 weeks [36].

Generally, the use of fractionation is known to increase the probability of controlling the tumor and to decrease damage to normal tissue surrounding the tumor. However, the motion of the patient or the internal organs between treatment sessions can result in failure to deliver adequate radiation to the tumor [68, 110]. We classify the delivery error in the following types:

1. *Registration Error* (see Fig. 34 (a)). Registration error is due to the incorrect positioning of the patient in day-to-day treatment. This is the *interfraction error* we primarily consider in this paper. Accuracy in patient positioning during treatment set-up is a requirement for precise delivery. Traditional positioning techniques include laser alignment to skin markers. Such methods are highly prone to error and in general show a displacement variation of 4–7mm depending on the site treated. Other advanced devices, such as electronic portal imaging systems, can reduce the registration error by comparing real-time digital images to facilitate a time-efficient patient repositioning [93].
2. *Internal Organ Motion Error*, (Fig. 34 (b)). The error is caused by internal motion of organs and tissues of a human body. For example, intracranial tissue shifts up to 1.5 mm when patients change position from prone to supine. The use of implanted radio-opaque markers allow physicians to verify the displacement of organs.

3. *Tumor Shrinkage Error*, (Fig. 34 (c)). This error is due to tumor area shrinkage as the treatment progresses. The originally prescribed dose delivered to target tissue does not reflect the change in tumor area. For example, the tumor can shrink up to 30% in volume within 3 treatments.
4. *Non-rigid Transformation Error*, (Fig. 34 (d)). This type of intrafraction motion error is internally induced by non-rigid deformation of organs, including for example, lung and cardiac motion in normal breathing conditions.

In our model formulation, we consider only the registration error between fractions and neglect the other three types of error. Internal organ motion error occurs during delivery and is therefore categorized as an *intrafraction error*. Our methods are not real-time solution techniques at this stage and hence are not applicable to this setting. Tumor shrinkage error and non-rigid transformation error mainly occur between treatment sessions and are therefore called interfraction errors. However, the changes in the tumor in these cases are not volume preserving and incorporating such effects remains a topic of future research. The principal computational difficulty arises in that setting from the mapping of voxels between two stages.

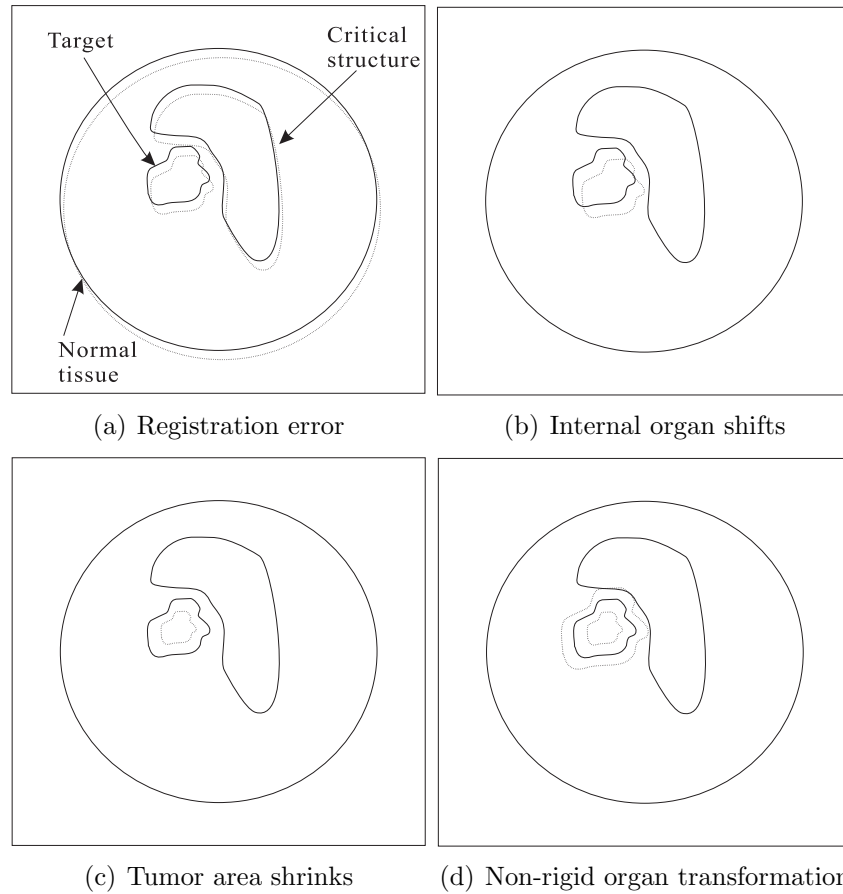


Figure 34: Four types of delivery error in hypo-fraction treatment

Off-line planning is currently widespread. It only involves a single planning step and delivers the same amount of dose at each stage. It was suggested in [10, 71, 108] that an optimal inverse plan should incorporate an estimated probability distribution of the patient motion during the treatment. Such distribution of patient geometry can be estimated [22, 54], for

example using a few pre-scanned images, by techniques such as Bayesian inference [109]. The probability distributions vary among organs and patients.

An alternative delivery scheme is so called on-line planning, which includes multiple planning steps during the treatment. Each planning step uses feedback from images generated during treatment, for example by CT scans. On-line replanning accurately captures the changing requirements for radiation dose at each stage, but it inevitably consumes much more time at every replanning procedure.

We formulate a *dynamic programming* (DP) framework that solves the day-to-day on-line planning problem. The optimal policy is selected from several candidate deliverable dose profiles, compensating over time for movement of the patient. The techniques are based on neuro-dynamic programming (NDP) ideas [9], which compute sub-optimal policies. Besides the NDP approaches, Sir et al [103] has applied other approximation techniques to solve the DP problem; for example, certainty equivalent control and open-loop feedback control.

In *neuro-dynamic programming/reinforcement learning* [6, 9, 47, 106], Monte Carlo simulation has been extensively used to evaluate cost structure when explicit form of the cost structure is not available or hard to obtain. Simulated cost values are incorporated in optimization procedures to select optimal policies, moreover, these values are employed in training for approximate cost-to-go functions. When incorporated within an optimization, Monte

Carlo simulation may involve substantial computation and dominate the solution time. Computational efficiency of simulation arises as a challenging issue in algorithmic design.

We focus on analyzing the rollout algorithm [6, 7, 8, 9, 36], one of *approximate dynamic programming* (ADP) techniques, for stochastic optimal control problems. The rollout algorithm performs a one-time policy improvement over an existing heuristic policy (or possible multiple heuristic policies). To illustrate, consider a discrete-time finite horizon control problem with N stages/periods. We assume the existence of perfect state information $x_k, k = 0, 1, \dots, N$, and suppose the state evolves according to a transition function f_k :

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \dots, N - 1. \quad (5.1)$$

Here u_k is the policy function applied to x_k at stage k and is drawn from a finite collection of policies U_k . $\mathbf{w} = \{w_0, w_1, \dots, w_{N-1}\}$ is a random vector in the state transition. Each component w_k follows a pre-defined probability distribution. The system yields an instantaneous cost/error $g_k(x_k, u_k(x_k), w_k)$ at stage k and a final terminal cost $g_N(x_N)$ at the last stage. Given an initial state x_0 , we aim to find a policy vector $\mathbf{u} = \{u_0, u_1, \dots, u_{N-1}\}$ that minimizes

an expected total cost-to-go starting from stage 0:

$$J_0^*(x_0) = \min_{\substack{\mathbf{u}: u_j \in U_j, \\ j=0,1,\dots,N-1}} \mathbb{E}_{\mathbf{w}} \left[\sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) + g_N(x_N) \middle| \begin{array}{l} x_{k+1} = f_k(x_k, u_k, w_k), \\ k = 0, 1, \dots, N-1 \end{array} \right]. \quad (5.2)$$

If we denote a cost-to-go function associated with the policy vector \mathbf{u} and starting from stage k by

$$J_k^{\mathbf{u}}(x_k) = \mathbb{E}_{\mathbf{w}} \left[\sum_{j=k}^{N-1} g_j(x_j, u_j, w_j) + g_N(x_N) \middle| \begin{array}{l} x_{j+1} = f_j(x_j, u_j, w_j), \\ j = k, k+1, \dots, N-1 \end{array} \right],$$

then the optimal cost-to-go function is simply

$$J_k^*(x_k) = \min_{\substack{\mathbf{u}: u_j \in U_j, \\ j=0,1,\dots,N-1}} \mathbb{E}_{\mathbf{w}} J_k^{\mathbf{u}}(x_k).$$

In particular, we have the recursive form of optimal cost-to-go functions

$$J_k^*(x_k) = \min_{u_k \in U_k} \mathbb{E}_{w_k} [g_k(x_k, u_k, w_k) + J_{k+1}^*(f_k(x_k, u_k, w_k))]. \quad (5.3)$$

The key idea of ADP centers around approximating the optimal cost-to-go function J_{k+1}^* with a simple-form function. Specifically, the rollout algorithm approximates the function J_{k+1}^* by a heuristic cost-to-go function $J_{k+1}^{\boldsymbol{\mu}}$, where $J_{k+1}^{\boldsymbol{\mu}}$ is obtained by applying the heuristic policy $\boldsymbol{\mu}$ starting from stage k and in all future stages. The heuristic policy $\boldsymbol{\mu}$ is called the *base*

policy. Define the Q -factor of (x_k, u_k) based on the heuristic policy $\boldsymbol{\mu}$ as

$$\begin{aligned} Q_k^\boldsymbol{\mu}(x_k, u_k) &= \mathbb{E}_{w_k} [g_k(x_k, u_k, w_k) + J_{k+1}^\boldsymbol{\mu}(f_k(x_k, u_k, w_k))] \\ &= \sum_{w_k} p(x_k, u_k, w_k) (g_k(x_k, u_k, w_k) + J_{k+1}^\boldsymbol{\mu}(f_k(x_k, u_k, w_k))), \end{aligned} \quad (5.4)$$

where state x_k evolves to the next stage with a transition probability $p(x_k, u_k, w_k)$.

This method yields a rollout policy

$$\bar{\mu}_k(x_k) = \arg \min_{u_k \in U_k} Q_k^\boldsymbol{\mu}(x_k, u_k). \quad (5.5)$$

The above equation corresponds to a one-step lookahead policy update. If the heuristic policy μ_k is one of the policies in U_k , it is shown in [6] that the rollout policy outperforms the heuristic policy

$$J_k^{\bar{\boldsymbol{\mu}}}(x_k) \leq J_k^\boldsymbol{\mu}(x_k).$$

The heuristic cost $J_{k+1}^\boldsymbol{\mu}(x_{k+1}) = J_{k+1}^\boldsymbol{\mu}(f_k(x_k, u_k, w_k))$ may be evaluated by Monte Carlo simulation when no closed form of $J_{k+1}^\boldsymbol{\mu}$ is available. Simulation generates a collection of sample trajectories from x_{k+1} : $\{\tilde{x}_{k+1,i} = x_{k+1}, \tilde{x}_{k+2,i}, \dots, \tilde{x}_{N,i}\}$, $i = 1, 2, \dots, M$. Each of them uses one set of realized random values $\{\tilde{w}_{k+1,i}, \tilde{w}_{k+2,i}, \dots, \tilde{w}_{N-1,i}\}$. The corresponding sample cost-to-go equals the sum of cumulative instantaneous costs plus the final cost

$$c_i^\boldsymbol{\mu}(x_{k+1}) = \sum_{j=k+1}^{N-1} g_j(\tilde{x}_{j,i}, \mu_j, \tilde{w}_{j,i}) + g_N(\tilde{x}_{N,i}).$$

The heuristic cost is estimated by

$$J_{k+1}^\boldsymbol{\mu}(x_{k+1}) \approx \tilde{J}_{k+1}^\boldsymbol{\mu}(x_{k+1}) = \frac{1}{M} \sum_{i=1}^M c_i^\boldsymbol{\mu}(x_{k+1}). \quad (5.6)$$

Such a simulation process can be time-consuming, depending on the complexity of the dynamic system and the number of simulation runs/replications M . We expect that a larger M improves the Monte Carlo estimation accuracy in (5.6), thus generating correct rollout policies, c.f., (5.5). Simulation error results in suboptimal policies, and possibly yields a lower-performance rollout policy than the base policy. Traditionally, the number M is set at a ‘large value’, and is uniform for all states at all stages, but the result of the setting is ambiguous. (Note that there are two levels of approximations involved

$$J_k^* \approx J_k^\mu \text{ and } J_{k+1}^\mu \approx \tilde{J}_{k+1}^\mu.$$

The discussion on simulation accuracy is in relation to the second level approximation, which is affected by the sample average approximation error in (5.6).)

In this chapter, we discuss an improvement to the computational efficiency of the Monte Carlo estimation. We deal with the tradeoff between rollout algorithm accuracy and Monte Carlo simulation efficiency. On one hand, we improve the precision of selecting rollout policies by increasing the number of simulation runs. On the other hand, we control the total number of simulation runs in order to save computation. In practical implementation, variable sample sizes M may be allowed for different states x_{k+1} and a mechanism to determine appropriate sizes $M_{x_{k+1}}$ is needed. The ideal pattern would be: promising states are assigned more simulation replications

in order to get accurate cost estimates; while for inferior states, a minimum number of replications should be used.

Our approach constructs a Bayesian posterior estimation for simulation outputs, and incorporates the posterior distribution information in the rollout algorithm. In Section 5.2, we first analyze the probability of correct selection (*PCS*) of rollout policies, given a fixed number of simulation replications. Secondly, in Section 5.3 we design a computation resource allocation scheme that can appropriately schedule additional simulation replications to obtain certain policy accuracy. Section 5.4 details numerical results on a fractionated radiotherapy example.

5.2 Evaluating Rollout Policy Accuracy

We outline some further details of the stochastic dynamic programming model. More specifically, we assume the dynamic model has these properties:

- a finite horizon with N stages.

We do not consider infinite horizon problems, because in such problems, the lengths of simulated sample trajectories may vary. Typically, a simulated trajectory is terminated early before reaching the final stage, when there is an evidence that a good sample cost is obtained. This complicates the problem and its analysis by introducing early termination error in addition to the sample average error in (5.6).

- a finite policy set $U_k = U = \{\hat{u}_l, l = 1, 2, \dots, L\}$.

Note that $U_k \equiv U, \forall k$, but this can be generalized easily at the expense of extra notations. If the policy set U consists of continuous or infinite many candidate policies, we can select a representative finite policy subset out of the original set. A typical example is the radiotherapy problem in Section 5.4.

- the transition random variable w_k has a finite number (D) of realizations.
- a finite or infinite state space \mathcal{S} for x_k .

In Section 5.1, we have mentioned that Monte Carlo simulation is employed to calculate the approximate heuristic costs $\tilde{J}_{k+1}^\mu(f_k(x_k, u_k, w_k))$, and therefore yields approximate Q -factors

$$\tilde{Q}_k^\mu(x_k, u_k) = \mathbb{E}_{w_k} \left[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}^\mu(f_k(x_k, u_k, w_k)) \right].$$

Note that $\tilde{Q}_k^\mu(x_k, u_k)$ is a random variable depending on the realizations of $\{w_{k+1}, w_{k+2}, \dots, w_{N-1}\}$. The rollout policy $\bar{\mu}_k$ may be imprecise due to the sample average estimation error. The probability of correct selection (*PCS*) is a measure to quantify the accuracy of the selected policy $\bar{\mu}_k$

$$PCS = Pr \left(\tilde{Q}_k^\mu(x_k, \bar{\mu}_k) + \delta \leq \tilde{Q}_k^\mu(x_k, \hat{u}_l), \forall \hat{u}_l \in U / \bar{\mu}_k \right), \quad (5.7)$$

where δ is an indifference-zone parameter. An indifference-zone formulation (5.7) [62] is introduced instead of direct comparisons $\tilde{Q}_k^\mu(x_k, \bar{\mu}_k) \leq$

$\tilde{Q}_k^\mu(x_k, \hat{u}_l), \forall \hat{u}_l \in U/\bar{\mu}_k$. The approach essentially ignores the Q -factor difference within the indifference-zone parameter δ thus relaxes the difficulty of distinguishing the difference between two Q -factors. For certain situations, the formulation is necessary; for example when different policies \hat{u}_l for x_k yield identical Q -factor values.

In Section 5.2.1, we describe how to construct Bayesian posterior estimation for the heuristic cost $J_{k+1}^\mu(x_{k+1})$. Both mean and variance information is modeled by posterior distributions. In Section 5.2.2, we describe how to employ the Bayesian information to compute the *PCS* of the selected policy.

5.2.1 Bayesian Posterior Estimation

The heuristic cost $J_{k+1}^\mu(x_{k+1})$ is estimated by the average of Monte Carlo sample costs $c_i^\mu(x_{k+1}), i = 1, 2, \dots, M$, c.f., (5.6). We assume that the underlying distribution of sample costs follows a normal distribution with mean $\theta(x_{k+1})$ and variance $\sigma^2(x_{k+1})$

$$c_i^\mu(x_{k+1}) \sim N(\theta(x_{k+1}), \sigma^2(x_{k+1})).$$

Based on the observed sample costs $c_i^\mu(x_{k+1})$, we can derive the Bayesian posterior distributions for the unknown parameters θ and σ^2 . (For convenience, we ignore the state x_{k+1} and use the notations θ and σ^2 in this subsection.) The mean parameter θ corresponds to the heuristic cost $J_{k+1}^\mu(x_{k+1})$ and is of more interest to us.

In the Bayesian framework, the mean θ and variance σ^2 are considered as random variables, whose distributions are inferred by Bayes' rule. Following the procedures in Section 1.2, we can asymptotically derive the posterior distribution of $\theta|X$ as

$$\theta|X \sim N(\bar{\theta}, \hat{\sigma}^2/M). \quad (5.8)$$

Here $\bar{\theta}$ and $\hat{\sigma}^2/M$ are the sample mean and sample variance, respectively. ' $|X$ ' stands for the posterior distribution given evaluated dataset.

5.2.2 Computing the PCS

Variable sample sizes M are admissible and can potentially save a significant amount of computational effort. Suppose a total of $M_{x_{k+1}}$ simulated trajectories are generated to evaluate $M_{x_{k+1}}$ sample costs $c_i^\mu(x_{k+1})$; see Figure 35 for an illustration. According to the Bayesian analysis in Section 5.2.1, we have the posterior estimation for the cost-to-go

$$J_{k+1}^\mu(x_{k+1})|X = \theta(x_{k+1})|X \sim N\left(\bar{\theta}(x_{k+1}), \frac{\hat{\sigma}^2(x_{k+1})}{M_{x_{k+1}}}\right). \quad (5.9)$$

By plugging (5.9) into (5.5), we see that

$$Q^\mu(x_k, u_k)|X = \sum_{w_k} p(x_k, u_k, w_k) \left(g_k(x_k, u_k, w_k) + J_{k+1}^\mu(f_k(x_k, u_k, w_k))|X \right) \quad (5.10)$$

also follows a normal distribution, because a linear combination of normal random variables is a normal variable. According to the definition in (5.7),

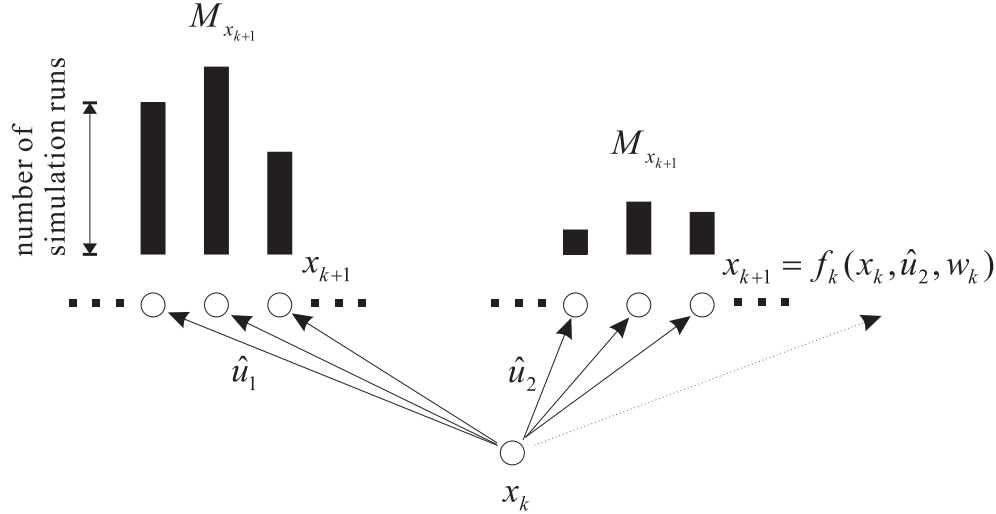


Figure 35: $M_{x_{k+1}}$ sample trajectories are simulated to evaluate the sample costs starting from the state x_{k+1} .

the *PCS* of the selected rollout policy can therefore be estimated via the posterior probability

$$\begin{aligned}
 PCS &= Pr \left(\tilde{Q}_k^\mu(x_k, \bar{\mu}_k) + \delta \leq \tilde{Q}_k^\mu(x_k, \hat{u}_l), \forall \hat{u}_l \in U / \bar{\mu}_k \right) \\
 &\approx Pr (Q_k^\mu(x_k, \bar{\mu}_k) | X + \delta \leq Q_k^\mu(x_k, \hat{u}_l) | X, \forall \hat{u}_l \in U / \bar{\mu}_k) \quad (5.11) \\
 &\geq 1 - \sum_{\substack{l=1,2,\dots,L \\ \hat{u}_l \neq \bar{\mu}_k}} Pr (Q_k^\mu(x_k, \bar{\mu}_k) | X + \delta > Q_k^\mu(x_k, \hat{u}_l) | X) \\
 &= 1 - \sum_{\substack{l=1,2,\dots,L \\ \hat{u}_l \neq \bar{\mu}_k}} Pr (Q_k^\mu(x_k, \hat{u}_l) | X - Q_k^\mu(x_k, \bar{\mu}_k) | X < \delta). \quad (5.12)
 \end{aligned}$$

In the above derivations, the first approximation is a substitution of Bayesian posterior distributions. The probability value can be computed based on the distributions of normal posterior variables $Q_k^\mu(x_k, u_k) | X$. The inequality is by Bonferroni's inequality [30]. Applying the inequality changes the joint

probability evaluation (5.11), which is difficult to calculate, to simple-form pairwise computations. Note that the pairwise subtraction $Q_k^\mu(x_k, \hat{u}_l)|X - Q_k^\mu(x_k, \bar{\mu}_k)|X$ is a normally distributed variable

$$Q_k^\mu(x_k, \hat{u}_l)|X - Q_k^\mu(x_k, \bar{\mu}_k)|X \sim N(a, b). \quad (5.13)$$

If we denote

$$\bar{Q}_k^\mu(x_k, u_k) = \sum_{w_k} p(x_k, u_k, w_k) (g_k(x_k, u_k, w_k) + \bar{\theta}(f_k(x_k, u_k, w_k))),$$

then the parameters a and b are

$$\begin{aligned} a &= \bar{Q}_k^\mu(x_k, \hat{u}_l) - \bar{Q}_k^\mu(x_k, \bar{\mu}_k), \\ b &= \sum_{w_k} \left(p^2(x_k, \hat{u}_l, w_k) \frac{\hat{\sigma}^2(f_k(x_k, \hat{u}_l, w_k))}{M_{f_k(x_k, \hat{u}_l, w_k)}} + p^2(x_k, \bar{\mu}_k, w_k) \frac{\hat{\sigma}^2(f_k(x_k, \bar{\mu}_k, w_k))}{M_{f_k(x_k, \bar{\mu}_k, w_k)}} \right). \end{aligned}$$

A lower bound estimate for the *PCS* is provided in (5.12), which is relatively easy to evaluate. Each probability component is a simple cdf calculation.

Given simulated sample costs, we analyze the accuracy of the selected rollout policies. Since we implement the Bayesian probability calculation, the *PCS* is the so called *Bayesian PCS*, which is a subjective probability value based on available sample costs. Besides the *PCS* evaluation, our analysis also provides an overall picture of the distributions of Q -factors; Figure 36 shows a plot of mean and variance of each Q -factor. Increasing the number of simulation replications can potentially decrease the variance of posterior Q -factors (making the shape of the curve narrower), and can therefore assist the selection process.

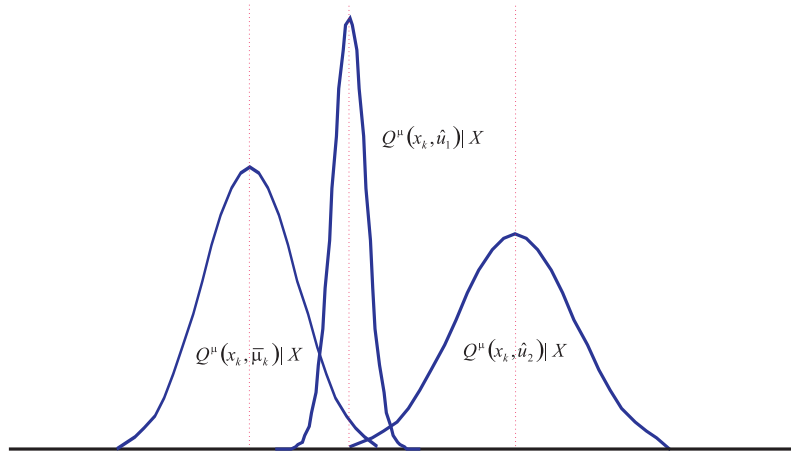


Figure 36: The posterior distribution plot of Q -factors. The rollout policy corresponds to the best Q -factor.

5.3 Allocating Simulation Resource

5.3.1 The Resource Allocation Scheme

At each stage, our objective is to obtain a certain accuracy on the probability of correct selection

$$PCS \geq \alpha, \quad (5.14)$$

where α is a threshold value. Typically, α is a value between 90% and 99%. A new resource allocation question arises about how to optimally assign the simulation replications, i.e., $M_{x_{k+1}}$, in order to satisfy the accuracy requirement. Alternatively, we may wish to optimally allocate a limited number of simulations to maximize the PCS of the rollout policy.

We propose a fast but suboptimal strategy here, one that assigns simulation replications sequentially. Let $PCS(\vec{M})$, where

$$\vec{M} = \{M_{x_{k+1}=f_k(x_k, u_k, w_k)} | \forall u_k, w_k\},$$

be the PCS using the sample size vector \vec{M} (as calculated using (5.9), (5.10), and (5.12)). Note that the length of \vec{M} depends on the cardinality L of U and the size D of possible realizations of w_k . We first evaluate $M_{x_{k+1}} = M_0$ replicated sample costs for states $x_{k+1} = f_k(x_k, u_k, w_k), \forall u_k, w_k$, where M_0 is an initial replication number. These initial simulated costs provide an estimate of the sample mean and sample variance. If the accuracy constraint (5.14) is not satisfied, additional simulation runs are carried out sequentially.

We assume that when an additional batch of \vec{M} replications are assigned to evaluate the cost-to-go $J_{k+1}^\mu(x_{k+1})$ of the state x_{k+1} , the sample mean $\bar{\theta}(x_{k+1})$ and sample variance $\hat{\sigma}^2(x_{k+1})$ remain invariant. Under this assumption, the posterior distribution of $J_{k+1}^\mu(x_{k+1})|X$ changes to

$$J_{k+1}^\mu(x_{k+1})|X \sim N\left(\bar{\theta}(x_{k+1}), \frac{\hat{\sigma}^2(x_{k+1})}{M_{x_{k+1}} + \vec{M}}\right).$$

To achieve a good allocation scheme, we invest our new resources in order to most sharply increase the potential PCS . Select the combination:

$$(\dot{u}_k, \dot{w}_k) = \arg \max_{u_k, w_k} PCS(\vec{M} + \vec{M}e_{u_k, w_k}), \quad (5.15)$$

where e_{u_k, w_k} is the standard unit vector with 1 on the component corresponding to (u_k, w_k) . When the size of the combinations $L \times D$ is moderate, the

computation is fast, since it only includes a limited number of cdf calculations.

Procedures to sequentially allocate simulation resources Given an initial sample size \bar{M}_0 , a batch size \bar{M} , a maximum sample size \bar{M}_{max} , and a threshold value α .

1. Simulate \bar{M}_0 sample costs for every possible state x_{k+1} . Set $M_{x_{k+1}} \leftarrow \bar{M}_0$.
2. Compute the sample mean $\bar{\theta}(x_{k+1})$ and sample variance $\hat{\sigma}^2(x_{k+1})$.
3. Determine the best combination index (\hat{u}_k, \hat{w}_k) via (5.15).
4. Simulate \bar{M} sample costs for the state $x_{k+1} = f_k(x_k, \hat{u}_k, \hat{w}_k)$ corresponding to the selected policy index.
5. Repeat Steps 2-4, until constraint (5.14) is satisfied or one of the sample sizes $M_{x_{k+1}}$ exceeds the maximum sample size \bar{M}_{max} .

5.3.2 Special Case – Finite State Space

We consider a special case when the state space \mathcal{S} is finite. In particular, we have a discrete form

- state space $\mathcal{S} = \{s_j, j = 1, 2, \dots, P\}$.

Since any state x_{k+1} is an element of \mathcal{S} , states $x_{k+1} = f_k(x_k, u_k, w_k)$, resulting from different combinations of (u_k, w_k) , may collapse to the same state $s_j \in \mathcal{S}$. Therefore, it is only necessary to consider variable sample sizes M_{s_j} based on different states s_j . See Figure 37 for an illustration. For promising states s_j , we expect the numbers M_{s_j} are large in order to accurately estimate the cost-to-go starting from s_j , while for other states, the numbers should be limited.

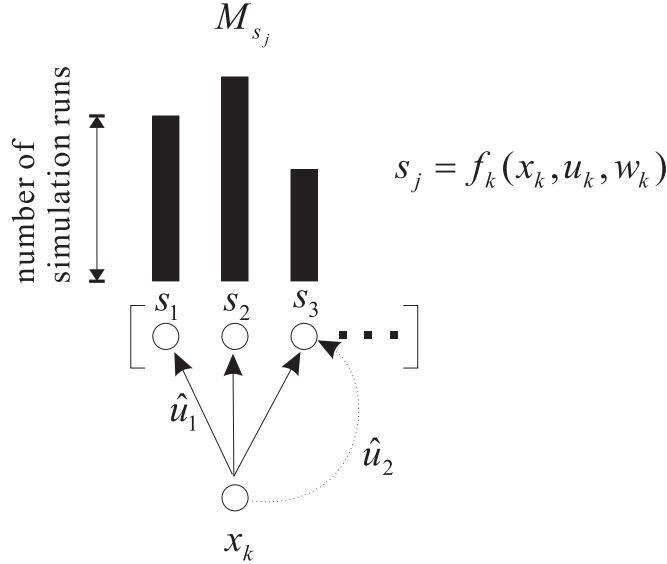


Figure 37: Finite state space. M_{s_j} sample trajectories are simulated to evaluate the cost-to-go starting from state s_j . State s_j at stage $k + 1$, derived from any policy u_k , use the same replication number.

Similar to our previous analysis, we have the posterior distribution of the cost-to-go

$$J_{k+1}^\mu(s_j)|X = \theta(s_j)|X \sim N\left(\bar{\theta}(s_j), \frac{\hat{\sigma}^2(s_j)}{M_{s_j}}\right). \quad (5.16)$$

The computation of PCS follows the same way.

The same sequential allocation scheme is implemented as above. The only difference is that now the vector $\vec{M} = \{M_{s_1}, M_{s_2}, \dots, M_{s_P}\}$ is of length P , thus in Step 3 of the allocation scheme we select the best index \hat{j} as

$$\hat{j} = \arg \max_{j=1,2,\dots,P} PCS(\vec{M} + \vec{M}e_j), \quad (5.17)$$

where $PCS(\vec{M} + \vec{M}e_j)$ is the potential PCS when \vec{M} simulation runs are added to M_{s_j} . Evaluation of $PCS(\vec{M})$ now refers to (5.16) and (5.12). There are a total of P potential PCS to evaluate.

5.3.3 An Extension – Group Similar M by Policy

The procedure in Section 5.3.1 assigns variable sample sizes $M_{x_{k+1}}$ for different states x_{k+1} . The length of the vector \vec{M} to determine is $L \times D$. In order to simplify the allocation procedure, we may group similar $M_{x_{k+1}}$ – use the same sample size $M_{\hat{u}_l}$ for all states $x_{k+1}^{\hat{u}_l} = f_k(x_k, \hat{u}_l, w_k)$, that are derived by the policy \hat{u}_l . A superscript \hat{u}_l is added to indicate the state $x_{k+1}^{\hat{u}_l}$ is derived by the policy \hat{u}_l . The numbers $M_{\hat{u}_l}$ now are uniform within each policy, and differ among policies. For promising policies, we expect the number $M_{\hat{u}_l}$ should be large to provide an good estimate, while for others, the number should be kept small. This manipulation only requires to compute a vector $\vec{M} = \{M_{\hat{u}_1}, M_{\hat{u}_2}, \dots, M_{\hat{u}_L}\}$ of length L .

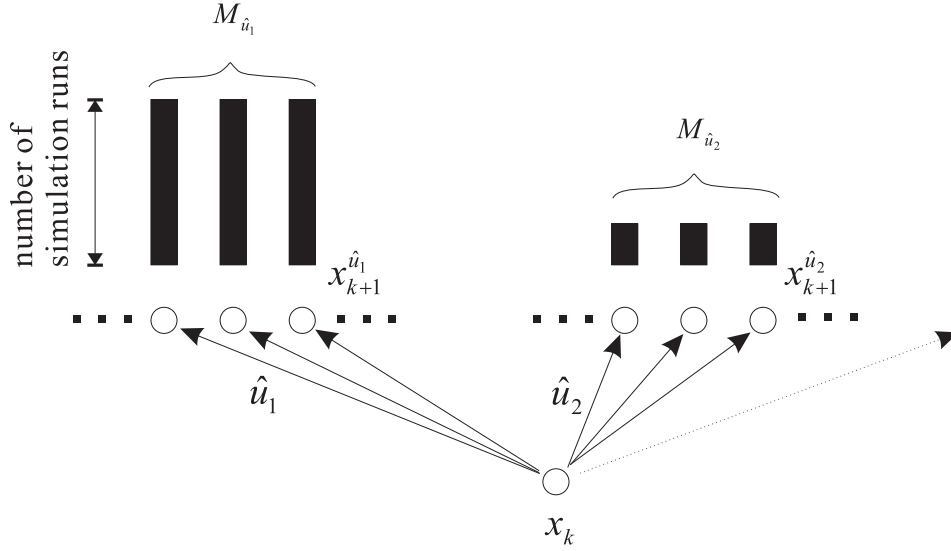


Figure 38: $M_{\hat{u}_l}$ sample trajectories are simulated to evaluate the sample costs starting from the state $x_{k+1}^{\hat{u}_l}$ which is derived from the policy \hat{u}_l . All the states x_{k+1} derived from the policy \hat{u}_l share the same replication number.

The posterior of the cost-to-go becomes

$$J_{k+1}^{\mu}(x_{k+1}^{\hat{u}_l})|X = \theta(x_{k+1}^{\hat{u}_l})|X \sim N\left(\bar{\theta}(x_{k+1}^{\hat{u}_l}), \frac{\hat{\sigma}^2(x_{k+1}^{\hat{u}_l})}{M_{\hat{u}_l}}\right), \quad (5.18)$$

which is plugged in (5.12) to compute PCS . Each posterior Q -factor $Q_k^{\mu}(x_k, \hat{u}_l)|X$ is obtained with same sample size $M_{\hat{u}_l}$. As a replacement of Step 3 in the resource allocation scheme, we determine the best index \hat{l} via

$$\hat{l} = \arg \max_{l=1,2,\dots,L} PCS(\vec{M} + \bar{M}e_l).$$

The operation is much simplified: at each iteration of the scheme, we only need to evaluate L potential PCS . We also expect the number of iterations of the scheme is reduced, too.

5.4 A Fractionated Radiotherapy Problem

The first example is adapted from a dynamic programming model in radiotherapy treatment [26, 36]. We demonstrate the application of the simulation resource allocation scheme with grouped $M_{\hat{u}_l}$ by policy.

In a radiotherapy treatment, radiation dose is delivered to the cancer tissue in a total of N fractionated treatments. The ideal plan is to deliver the prescribed dose to the target (cancer) region, while avoiding the surrounding healthy tissue (critical organs and normal tissue). Since patient motion is inevitable during the treatments, our objective is to minimize the total miss-delivered error, including both overdose and underdose errors, throughout the treatment process.

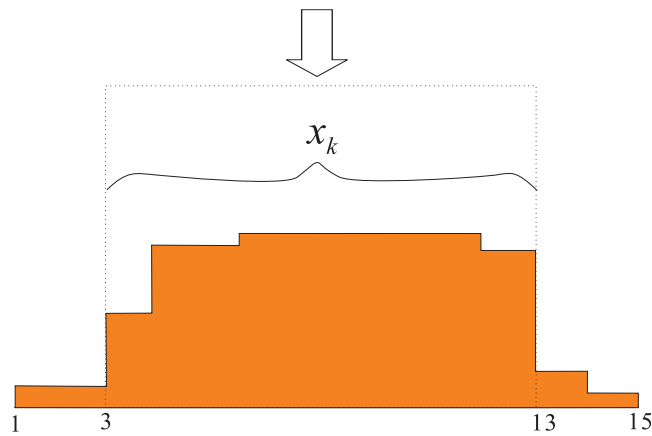


Figure 39: A simple one-dimension problem. x_k is the dose distribution over voxels in the target: voxels 3, 4, \dots , 13.

A simple one-dimensional dynamic programming model is established. See Figure 39 for an illustration. The setting consists of 15 voxels $\{1, 2, \dots, 15\}$,

where the target voxel set $\mathcal{T} = \{3, 4, \dots, 13\}$ is located in the center, and the remaining voxels represent the healthy tissue. The whole process includes $N = 10$ fractionated treatments. The state $x_k(i), k = 0, 1, \dots, N, i \in \mathcal{T}$, contains the actual dose delivered to target voxels after k stages/treatments, with the initial state $x_0(i) = 0$. $u_k(x_k)$ is the dose applied at the k th stage and is subject to an uncertain patient positioning error w_k . The state transition is in the form

$$x_{k+1}^{u_k}(i) = f_k(x_k(i), u_k(i), w_k) = x_k(i) + u_k(i + w_k), \forall i \in \mathcal{T}. \quad (5.19)$$

w_k is regarded as a discretely distributed random variable, with $D = 5$ possible realizations. We assume $\{w_k\}$ are i.i.d. and follow the distribution

$$w_k = \begin{cases} -2, & \text{with probability } 0.02 \\ -1, & \text{with probability } 0.08 \\ 0, & \text{with probability } 0.8 \\ 1, & \text{with probability } 0.08 \\ 2, & \text{with probability } 0.02, \end{cases}$$

for every state k .

If we denote the ideal prescribed dose on the target by T , where $T(i) = 1, i \in \mathcal{T}$ and $T(i) = 0$ elsewhere, the final cost is a linear combination of the

absolute differences between the final dose and the ideal dose

$$J_N(x_N) = \sum_{i \in \mathcal{T}} \phi(i) |x_N(i) - T(i)|.$$

Here the vector ϕ weights the dose delivery error in different regions. Typically, we set $\phi(i) = 10$, for $i \in \mathcal{T}$ and $\phi(i) = 1$ elsewhere to emphasize the importance of the target tissue. The immediate cost g_k at each stage corresponds to the amount of dose delivered outside of the target,

$$g_k(x_k, u_k, w_k) = \sum_{i+w_k \notin \mathcal{T}} \phi(i+w_k) u_k(i+w_k).$$

The policy set U contains $L = 5$ different delivery policies

$\{u_c, u_r, u_{mr1}, u_{mr2}, u_{mr3}\}$:

- Constant policy u_c : at each stage, deliver $u_c = T/N$ and ignores the patient motion error.
- Reactive policy u_r : at each stage, deliver the residual required dose divided by the remaining time stages $u_r = \max(0, T - x_k)/(N - k)$.
- Modified reactive policies u_{mr} : an amplifying parameter β is multiplied to the reactive policy $u_{mr} = \beta \cdot \max(0, T - x_k)/(N - k)$. Three modified reactive policies $\{u_{mr1}, u_{mr2}, u_{mr3}\}$ corresponding to $\beta = \{1.5, 2, 2.5\}$ are considered.

In the implementation of the rollout algorithm, as u_{mr3} yielded the best performance among all the policies in the set U , it was used as the base

heuristic policy $\mu_k = u_{mr3}$. We implemented the version of the allocation scheme with grouped sample sizes $M_{\hat{u}_l}, \hat{u}_l \in U = \{u_c, u_r, u_{mr1}, u_{mr2}, u_{mr3}\}$ (Section 5.3.1 and Section 5.3.3). $M_{\hat{u}_l}$ sample trajectories were simulated starting at $x_{k+1}^{\hat{u}_l}$ and applying the policy μ to approximate the cost-to-go $J_{k+1}^\mu(x_{k+1}^{\hat{u}_l})$.

To demonstrate our simulation resource allocation scheme, we considered the following setting. The initial sample size $\bar{M}_0 = 10$. 10 sample costs were not a big quantity, but provided good estimates of the sample mean and sample variance. The maximum sample size \bar{M}_{max} was set to 50,000. To sharply increase the sample sizes $M_{\hat{u}_l}$, every time we tried a batch of $\bar{M} = M_{\hat{u}_l}$ additional simulations, which is equivalent to reducing the posterior variance $\hat{\sigma}^2(x_{k+1}^{\hat{u}_l})/M_{\hat{u}_l}$ to a half. For all the stages, we required the accuracy of the selected rollout policy to be greater than $\alpha = 90\%$. Despite the situation when the maximum number of sample size 50,000 was reached, the accuracy was always obtained.

In one trajectory executing the rollout policy, Table 17 shows the sample sizes $M_{\hat{u}_l}$ used at the stages 0-9. We observe that $M_{\hat{u}_l}$ are actually varied from policy to policy. Another interesting observation is that the sum $\sum_{\hat{u}_l} M_{\hat{u}_l}$ decreases sharply as the stage k increases. This shows that our variable sample size approach is advantageous over the fixed sample size approach, which uses a uniform sample size for all states $x_{k+1}^{\hat{u}_l}$ and for all stages. To explain the decrease of sample sizes along the stage, we may refer to Table 18,

which plots the sample-mean Q -factors $\bar{Q}_k^\mu(x_k, \hat{u}_l)$ in the same trajectory. At early stages, the Q -factors are very close. For example, we recorded that $\bar{Q}_0^\mu(x_0, \hat{u}_l)$ at stage 0 were $\{0.3305, 0.3293, 0.3264, 0.3236, 0.3216\}$. The reason for close Q -factors (cost-to-go applying different policies) is that even if a poor policy is applied at stage 0, there are still many chances to apply good policies in later stages to compensate for the mistake. However, as stage k increases, there is no room for such compensation. Thus we observe that the range of Q -factors increases as k increases. At early stages, since the Q -factors are close, it necessitates a large sample size to distinguish the policies correctly. On the other hand, in late stages, since the Q -factor differences are large, only the least number of simulation runs were used; for example, for stage 8 and 9, $M_{\hat{u}_l} = \bar{M}_0 = 10, \forall \hat{u}_l \in U$.

The indifference-zone parameter δ was set as 0.01 in this example, because we found that the expected initial cost-to-go $J_0(x_0)$ was close to 0.3. The value implies that we would not distinguish two policies, if their associated Q -factors were within 0.01. The introduction of the indifference-zone parameter is important. As we see in Table 18, at the last stage, the Q -factors for policies $\{u_r, u_{mr1}, u_{mr2}, u_{mr3}\}$ are at the same value, because these reactive policies yield exactly the same amount of dose (by their definition). If there is more than one ‘best’ policy, the PCS cannot be larger than 0.5. The indifference-zone formulation is introduced to tackle this problem. The indifference-zone parameter should be set carefully, since a biased value could either lose the

Table 17: Variable sample sizes $M_{\hat{u}_i}$ for all the stages.

Stage (k)	M_{u_c}	M_{u_r}	$M_{u_{mr1}}$	$M_{u_{mr2}}$	$M_{u_{mr3}}$
0	10000	10000	10000	20000	40000
1	5000	5000	5000	10000	20000
2	10000	10000	5000	10000	20000
3	2500	5000	2500	2500	10000
4	2000	600	2000	2000	5000
5	20	640	640	20	320
6	20	80	20	20	80
7	10	10	10	10	10
8	10	10	10	10	10
9	10	10	10	10	10

selection accuracy (draw a selection decision at an improper time) or make it difficult to identify the best policy.

The rollout policy corresponds to the policy with the best Q -factors. In Table 18, we find that the rollout policy $\bar{\mu}_k$ matches u_{mr3} in many stages (because u_{mr3} is overall the best policy), and also matches u_{mr1} or u_{mr2} a couple of times. Since the constant policy u_c is an offline policy, which does not use the current delivery status x_k , it performed the worst. At stage 0, the realized displacement of w_0 was 0, thus the optimal cost-to-go $J_1(x_1)$ become smaller than $J_0(x_0)$. Nevertheless, at stage 1, the realized displacement was -1, resulting in a miss-delivered dose. The optimal cost-to-go $J_2(x_2)$ increased.

Since we grouped $M_{\hat{u}_i}$ by policy, the promising policies in the set U end up with larger sample sizes. As shown in Table 17, $M_{u_{mr3}}$ is likely to be the largest number, because the policy u_{mr3} is likely to be the best policy. The fact is also implied from the posterior PCS computation in (5.12). We found that if we invest simulation resources to the states $x_{k+1}^{\bar{\mu}_k}$ derived from the best policy $\bar{\mu}_k$, the potential variance reduction can benefit each pairwise probability component, thus induces a quick PCS increment. Promising policies yield Q -factors that are close to the best Q -factor $\bar{Q}_k^\mu(x_k, \bar{\mu}_k)$ (referring to Figure 36); therefore, they need more simulation replications in order to differentiate from the best policy.

Table 18: Sample-mean Q -factors for all the stages.

Stage (k)	$\bar{Q}_k^\mu(x_k, u_c)$	$\bar{Q}_k^\mu(x_k, u_r)$	$\bar{Q}_k^\mu(x_k, u_{mr1})$	$\bar{Q}_k^\mu(x_k, u_{mr2})$	$\bar{Q}_k^\mu(x_k, u_{mr3})$	Policy (\hat{u}_l)
0	0.3305	0.3293	0.3264	0.3236	<u>0.3216</u>	u_{mr3}
1	0.2405	0.2394	0.2388	0.2378	<u>0.2359</u>	u_{mr3}
2	0.4368	0.4357	0.4327	0.4279	<u>0.4276</u>	u_{mr3}
3	0.3751	0.3757	0.3729	<u>0.3687</u>	0.3689	u_{mr2}
4	0.6582	0.6585	0.6525	0.6437	<u>0.6397</u>	u_{mr3}
5	0.6044	0.6085	0.6009	0.5937	<u>0.5867</u>	u_{mr3}
6	0.8099	0.7233	<u>0.6788</u>	0.6981	0.7023	u_{mr1}
7	1.1238	0.7294	0.7001	0.6711	<u>0.6387</u>	u_{mr3}
8	1.5821	1.0648	0.9379	<u>0.8112</u>	0.8537	u_{mr2}
9	1.5356	0.6556	<u>0.6556</u>	<u>0.6556</u>	<u>0.6556</u>	$u_{mr1,2,3}$

5.5 Conclusions

Monte Carlo simulation is an important component in neuro-dynamic programming (for stochastic problems). Although simulation error is inevitably involved, we aim to minimize the amount the simulation effort while obtaining a desired algorithmic accuracy. A variable sample size structure for simulation is implemented. Appropriate sample sizes in the structure are determined with the Bayesian posterior information. We have shown that our approach significantly saves simulation effort and is more effective when the simulation runs are expensive to carry out.

Chapter 6

Conclusions

In a simulation-based optimization problem, we seek the optimal setting of input parameters to maximize the simulation performance or to consume the least simulation cost. The objective function corresponds to a performance (or cost) measure and has several properties that distinguish it from deterministic objectives: 1) There is typically no closed form of the objective function. 2) The objective function is normally computationally expensive. 3) The objective function is affected by uncertainties in simulation. Throughout the thesis, we have considered minimizing the expected form of the stochastic objective function.

The two-phase optimization framework (WISOPT) has been successfully applied to simulation-based optimization problems; more generally, the framework is applicable to handle noisy functions. Phase I involves global exploration of the entire domain space to identify promising local subregions. The approaches implemented in Phase I are the classification-based global search and the Noisy DIRECT algorithm. A phase transition module is applied to derive the locations of a collection of promising subregions.

Phase II applies local optimization techniques to determine optimal solutions in each local subregion. According to the implementation of Common Random Numbers (CRN), we consider two extensions of the UOBYQA algorithm: the VNSP-UOBYQA and Noisy UOBYQA algorithms. A Matlab implementation of the WISOPT two-phase framework optimization is available for download, and further details are available in Appendix A.

In Phase I, both the classification-based global search and the Noisy DIRECT algorithm return a collection of samples (over the domain), which are densely distributed in promising regions. The classification-based global search simplifies the objective function as a 0-1 indicator function, which is approximated by an ensemble of classifiers. The Noisy DIRECT method adopts the idea of partitioning domains. The promising regions are subjected to higher levels of partitioning into smaller hyperrectangles. At the end of Phase I, the phase transition module applies a non-parametric local regression process to identify the region centers and sizes.

In Phase II, the modified UOBYQA algorithms search the local optimums starting from each region center. The VNSP-UOBYQA algorithm is designed for the CRN case and the Noisy UOBYQA algorithm is designed for the white noise case. Both algorithms apply a similar Bayesian sampling scheme to determine the replication sizes, thus making the algorithms smoothly proceed as in the deterministic case. The best local optimum is considered as the final output of the algorithms.

Bayesian analysis has played an important role in our algorithmic design. Most of our algorithms replicate multiple function evaluations to reduce the output uncertainty. Since Bayesian analysis constructs succinct forms of posterior distributions to model the simulation output, i.e., using normal distributions or t-distributions, they are easily incorporated into existing optimization algorithms. Most commonly, we establish specific variance controlling criteria for individual algorithms, i.e., controlling volatility of promising hyperrectangle set in DIRECT and controlling volatility of the subproblem solutions in UOBYQA. Such criteria are tested using a Monte Carlo validation process, which is based on the derived Bayesian posterior distributions. When the desired accuracy (or the criterion) is not satisfied, we need to increase the replication number to reduce the uncertainty even more.

To illustrate the effectiveness of WISOPT, we have applied it to a variety of applications. In the Wisconsin Epidemiology Simulation Model, we aimed to determine the best parameter setting which drives the simulation model to accurately replicate real patient incidences. In the microwave antenna project, we aimed to find the best antenna design that yields a desired performance, under some realistic assumptions regarding properties of individual's livers. In the ambulance base simulation problem, we determined the locations of multiple ambulance bases such that the expected response time of an ambulance to an emergency call was minimized.

While these examples demonstrated the effectiveness of our code in solving simulation optimization problems, we believe its applicability is even more widespread. For example, with the tools available, we may solve other types of problems, such as calibration, SVM parameter tuning, inverse problems, and two-phase stochastic integer programming problems with recourse, each of which is considered to be a difficult problem to solve. The WISOPT code couples many contemporary statistical tools (Bayesian statistics and nonparametric statistics) and optimization techniques, and shows effectiveness in processing noisy function optimizations. Moreover, the WISOPT is applicable to deterministic global optimization problem as well. In fact, for deterministic problems, we have designed a different approach for the phase transition module, because there is no random error involved in the regressions.

There are still several challenges facing our two-phase strategy. For example, we are currently working on unconstrained stochastic problems, but how do we extend our work to handle problems with simple bounds? For the UOBYQA algorithm, there are difficulties in extending the analysis to problems with simple bounds. Another question would be how to design a more robust phase transition procedure that generates different radii for different subregions. Moreover, we need to consider adding more optimization ‘modules’ besides current ones in our two-phase framework and extend the framework to handle mixed-variable problems. More general termination

criteria are also needed.

At the end of the thesis, we presented a special application of Bayesian analysis in Neuro-Dynamic Programming (NDP). The type of dynamic programming problem has time stages and internal state transitions, which are different from the main theme of the thesis. Control at each time stage is computed in a sequential manner. NDP provides approximation methods to solve the dynamic programming problem efficiently, but only derive sub-optimal controls. We modify the rollout algorithm from NDP, that has shown advantages in deriving accurate stochastic controls, while saving Monte Carlo simulation resources. A test example in fractionated radiotherapy treatment is solved using the new algorithm. Besides the application in NDP, we believe the Bayesian simulation analysis is extendable to much more optimization techniques in various contexts.

Appendix A

User Guide to WISOPT

WISOPT is an optimization package designed for simulation-based optimization problems in the form (1.1). The flow chart for the two-phase framework is given in Figure 3. In this appendix, we give a short guide to the WISOPT optimization package. Most of the routines in the WISOPT package are implemented in Matlab. Although the original versions of the VNSP-UOBYQA and the Noisy UOBYQA are coded in C, mex interfaces [76] are employed to call them directly within Matlab.

The primary routine in the WISOPT is called as follows:

```
[globX, globY] = WISOPT(problem, opt)
```

In the above expression, the input parameters

- `problem` is a structure defining the objective function
- `opt` is a structure defining the algorithm options, including options for individual algorithms

and output parameters

- `globX` is the global solution x^* .
- `globY` is the global objective value

A routine `setup.m` is designed to prepare the values for the structures `problem` and `opt`. More specifically, the `problem` structure contains the following fields:

- `problem.fun` is a function handle to the objective function.
- `problem.lowBnd` is the lower bound of the input parameters.
- `problem.upBnd` is the upper bound of the input parameters.
- `problem.CRN` is an indicator to the Common Random Number (CRN) implementation. 1 = CRN case, 0 = white noise case.
- `problem.n` is the dimension of the problem. The value is computed based on the length of the lower bound vector.

Since WISOPT is an optimization framework hooking up various algorithms, the structure `opt` contains the options for running WISOPT and options for each component algorithm in the two-phase framework.

- `opt.phaseIMethod` a string field indicating the Phase I method to use.

Available choices currently include

- ‘CBS’ - the classification-based global search

- ‘NoisyDIRECT’ - the Noisy DIRECT algorithm.
- `opt.phaseTransMethod` is a string field indicating the phase transition algorithm to use.
 - ‘localReg’ - default algorithm for general noisy cases
 - ‘RSquaure’ - algorithm to apply only when the objective function is deterministic.
- `opt.cbs` is a structure of the classification-based global search options.
- `opt.direct` is a structure of the Noisy DIRECT options
- `opt.nuobyqa` is a structure of the Noisy UOBYQA options.
- `opt.vnsp` is a structure of the VNSP-UOBYQA options

Detailed options for each algorithm are listed below. For the classification-based global search, the following options should be supplied (the default value for an option is printed in the parenthesis)

- `opt.cbs.samplingMethodStage1` and `opt.cbs.samplingMethodStage2` - handles of stage 1 and stage 2 sampling functions. Currently available choices of sampling functions include `gridMesh` for grid sampling and `LHS` for the Latin Hypercube Sampling. The sampling method routines are located in the subfolder `\sampling_methods`. The standard calling format of a sampling function is


```
X = sampling_fun(lowBnd, upBnd, nSample)
```

where `X` is the output sample set, `lowBnd` and `upBnd` define the domain and `nSample` indicates the number of samples to be generated. Users can supply customized sampling functions following the above format.

- `opt.cbs.numSampleStage1` (500), `opt.cbs.numSampleStage2` (5000) - numbers of samples evaluated in both stages.
- `opt.cbs.classifiers` - contains the structure defining the classifiers to use. Five classifiers are provided: `SVM_linear`, `SVM_quadratic`, `SVM_cubic`, `SVM_gaussian` are support vector machine classifiers with different kernel functions; `kNearestNeighbor` is the k-Nearest Neighbor classifier. All the classifiers functions are located in the `\classifiers` subfolder and should have the following calling sequence:

```
testY = classifier_fun(trainX, trainY, testX)
```

where `trainX` and `trainY` are the input training data for the classifier and `testX` is the input test data. `testY` is the classifier prediction based on `testX`.

- `opt.cbs.ratio` (0.1) - the cutoff ratio for level set.

For the Noisy DIRECT method, the following options are supplied:

- `opt.direct.maxevals` (2000) - maximum number of function evaluations.
- `opt.direct.showits` (0) - level of display information.
- `opt.direct.iniRepNum` (3) - initial number of replications per point.
- `opt.direct.alpha` (1.3) - replication number increment ratio.
- `opt.direct.maxRepNum` (50) - maximum number of replications.
- `opt.direct.mcNum` (100) - Monte Carlo experiment number.
- `opt.direct.rateThresh` (0.9) - potential set identification accuracy.

The Phase II methods, the VN_{SP}-UOBYQA and the Noisy UOBYQA share similarities in parameters, which are:

- `opt.vnsp.maxFun` (2000) - maximum number of function evaluations.
- `opt.vnsp.iPrint` (0) - level of display information.
- `opt.vnsp.rhoBeg` (1) and `opt.vnsp.rhoEnd` (0.0001) - starting and ending values of the radius. The value of `opt.vnsp.rhoBeg` is adjusted according to the phase transition module.
- `opt.vnsp.iniRepNum` (3) - initial number of replications per point.
- `opt.vnsp.maxRepNum` (100) - maximum number of replications.

The objective function, varied in the CRN case and the white noise case, is defined as follows:

the CRN case: $y = \text{fun}(x, i)$

the white noise case: $y = \text{fun}(x)$

In the CRN case, the index i should be specified and the objective function returns the value $F(x, \xi_i)$ according to the value of i . Examples of test functions can be found in the `\test_functions` directory.

Here is an example of generating a noisy Rosenbrock function, which automatically decides the functional form by detecting whether the second input i is supplied. In the white noise case, an additive noise term with mean 0 and variance 0.1^2 is appended to the function output; in the CRN case, the first component x_1 of x is modified by noise, which is the same as the example given in Section 3.2.

```
function yRand = Rosen(x, varargin)
% in the white noise case, the objective function takes a single
% input x
if length(varargin) < 1
    sigma = 0.1; mu = 0;
    yRand = 100*(x(1)^2-x(2))^2 + (1-x(1))^2 + normrnd(mu,sigma);
else
% in the CRN case, a sequence of omega should be predefined
    global omega;
```

```

startNum = varargin{1};
xhat = x(1)*omega(startNum);
yRand = 100*(xhat^2-x(2))^2 + (1-xhat)^2;
end

```

As a test example, we show how to set up and solve the Rosenbrock function in the CRN case. Given the sequence of $\omega \sim N(1, 0.1^2)$ as in Section 3.2, the global minimizer of the function is $[0.4162, 0.1750]$, at which the optimal objective value is 0.4632.

```

>> problem.fun = @Rosen;
>> problem.lowBnd = [-2 -2];
>> problem.upBnd = [2 2];
>> problem.CRN = 1;

```

Set up the global random vector ω ,

```

>> global omega
>> omega = normrnd(1,0.1,1000,1);

```

Set the Phase I method to be the classifier method,

```

>> opt.phaseIMethod = 'CBS';

```

Execute the WISOPT routine,

```

>> [globX globY] = WISOPT(problem, opt)

```

```

Starting Phase I optimization
Applying the Classification-Based Global Search
Applying the phase transition module
Starting Phase II optimization
Common Random Number case
Applying the VNSP_UOBYQA algorithm

```

```

globX =
    0.4206    0.1812
globY =
    0.4551

```

At the return of WISOPT, the global solution is [0.4206, 0.1812] and the global objective value is 0.4551. As we notice, the objective value is a sample mean of the replicated outputs and is biased by noise.

We give a second example of optimizing the Goldstein Price function. This time we minimize the function in the white noise case and use the Noisy DIRECT method as the Phase I method. The source code for the noisy Goldstein Price function is printed below.

```

function yRand = gp(x, varargin)
x1 = x(1); x2 = x(2);
value =(1+(x1+x2+1).^2.*(19-14.*x1+3.*x1.^2-14.*x2...
    +6.*x1.*x2+3.*x2.^2)).*(30+(2.*x1-3.*x2).^2.*...

```

```

        (18-32.*x1+12.*x1.^2+48.*x2-36.*x1.*x2+27.*x2.^2));
% the white noise case
if length(varargin) < 1
    sigma = 0.1; mu = 0;
    yRand = value + normrnd(mu,sigma);
else
% the CRN case
    global omega;
    startNum = varargin{1};
    yRand = value + omega(startNum);
end

```

An additive noise term $N(0, 0.1^2)$ is added to the function output in the white noise case. The Goldstein Price function has a global solution at $[0, -1]$ and a global objective value at 3. Let us set up the optimization problem and the Phase I method,

```

>> problem.fun = @gp;
>> problem.lowBnd = [-2 -2];
>> problem.upBnd = [2 2];
>> problem.CRN = 1;
>> opt.phaseIMethod = 'NoisyDIRECT';

```

Execute the optimization command,

```
>> [globX globY] = WISOPT(problem, opt)
```

```
Starting Phase I optimization
```

```
Applying the Noisy DIRECT algorithm
```

```
Applying the phase transition module
```

```
Starting Phase II optimization
```

```
White Noise Case
```

```
Applying the Noisy UOBYQA algorithm
```

```
globX =
```

```
    -0.0002    -0.9986
```

```
globY =
```

```
    2.9677
```

Again, we obtain a good solution $[-0.0002, -0.9986]$, which is close to the true solution.

At the end of the thesis, we provide a table summarizing the options in WISOPT.

Table 19: Options with default values

Options	Default values
<code>opt.phaseIMethod</code>	'NoisyDIRECT'
<code>opt.phaseTransMethod</code>	'localReg'
Classification-based global search	
<code>opt.cbs.samplingMethodStage1</code>	@LHS
<code>opt.cbs.samplingMethodStage2</code>	@LHS
<code>opt.cbs.numSampleStage1</code>	500
<code>opt.cbs.numSampleStage2</code>	5000
<code>opt.cbs.ratio</code>	0.1
Noisy DIRECT	
<code>opt.direct.maxevals</code>	2000
<code>opt.direct.showits</code>	0
<code>opt.direct.iniRepNum</code>	3
<code>opt.direct.alpha</code>	1.3
<code>opt.direct.maxRepNum</code>	50
<code>opt.direct.mcNum</code>	100
<code>opt.direct.rateThresh</code>	0.9
VNsp-UOBYQA or Noisy UOBYQA (replace <code>vnsp</code> by <code>nuobyqa</code>)	
<code>opt.vnsp.maxFun</code>	2000
<code>opt.vnsp.iPrint</code>	0
<code>opt.vnsp.rhoBeg</code>	1
<code>opt.vnsp.rhoEnd</code>	0.0001
<code>opt.vnsp.iniRepNum</code>	3
<code>opt.vnsp.maxRepNum</code>	100

Bibliography

- [1] CISNET, <http://cisnet.cancer.gov/profiles/>.
- [2] M. Abramson. NOMADm: Mash adaptive direct search Matlab toolbox, 2006. <<http://www.afit.edu/en/ENC/Faculty/MAbramson/NOMADm.html>>.
- [3] S. Andradóttir. Simulation optimization. In J. Banks, editor, *Handbooks of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter 9. John Wiley & Sons, New York, 1998.
- [4] A. Ben-Tal and A. Nemirovski. Robust solutions to uncertain programming. *Operations Research Letters*, 23:769–805, 1999.
- [5] J. M. Bertram, D. Yang, M. C. Converse, J. G. Webster, and D.M. Mahvi. A review of coaxial-based interstitial antennas for hepatic microwave ablation. *Critical Reviews in Biomedical Engineering*, 34:187–213, 2006.
- [6] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, third edition, 2005.
- [7] D. P. Bertsekas. Dynamic programming and suboptimal control: A

survey from AD to MPC. 2005. To appear in the European Journal on Control and 2005 CDC Proceedings.

- [8] D. P. Bertsekas and D. A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristic*, 5(1), 1999.
- [9] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
- [10] M. Birkner, D. Yan, M. Alber, J. Liang, and F. Nusslin. Adapting inverse planning to patient and organ geometrical variation: Algorithm and implementation. *Medical Physics*, 30(10):2822–31, 2003.
- [11] Th. Bortfeld. Current status of IMRT: physical and technological aspects. *Radiotherapy and Oncology*, 61(2):291–304, 2001.
- [12] E. P. G. Box and N. R. Draper. *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, New York, 1986.
- [13] M. Cao and M. C. Ferris. Genetic algorithms in optimization. *Journal of Undergraduate Mathematics and its Applications*, 12:81–90, 1991.
- [14] Ph. Chan, W. Fan, A. Prodromidis, and S. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, 14(6):67–74, 1999.

- [15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. Ph. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [16] C.-H. Chen, K. Donohue, E. Yucesan, and J. Liu. Optimal computing budget allocation for monte Carlo simulation with application to product design. *Simulation Modelling Practice and Theory*, 11:57–74, 2003.
- [17] H.-C. Chen, C.-H. Chen, and E. Yucesan. An asymptotic allocation for simultaneous simulation experiments. In *Proceedings of the 1999 Winter Simulation Conference*, pages 359–366, 1999.
- [18] S. Chick. Personal communication.
- [19] S. E. Chick and K. Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49:1609–1624, 2003.
- [20] A. R. Conn, K. Scheinberg, and P. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. *Approximation Theory and Optimization, Tributes to M. J. D. Powell*, edited by M. D. Buhmann and A. Iserles, pages 83–108, 1996.
- [21] A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and

- F. Giannessi, editors, *Nonlinear Optimization and Applications*, pages 27–47. Plenum Press, New York, 1996.
- [22] C. L. Creutzberg, G. V. Althof, M. de Hooh, A. G. Visser, H. Huizenga, A. Wijnmaalen, and P. C. Levendag. A quality control study of the accuracy of patient positioning in irradiation of pelvic fields. *International Journal of Radiation Oncology, Biology and Physics*, 34:697–708, 1996.
- [23] A. Das and B. K. Chakrabarti, editors. *Quantum Annealing and Related Optimization*. Springer-Verlag, Heidelberg, 2005.
- [24] A. P. Dawid. Some matrix-variate distribution theory: Notational considerations and a bayesian application. *Biometrika*, 68(1):265–74, 1981.
- [25] M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, Inc, New York, 1970.
- [26] G. Deng and M. C. Ferris. Neuro-dynamic programming for fractionated radiotherapy planning. To appear in *Optimization in Medicine*.
- [27] G. Deng and M. C. Ferris. Variable-number sample-path optimization. 2007. To appear in *Mathematical Programming, Series B*.
- [28] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

- [29] F. A. Duck. *Physical Properties of Tissue: A Comprehensive Reference Book*. Academic Press: Harcourt Brace Jovanovich, 1990.
- [30] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, Pacific Grove, California, 3rd edition, 2004.
- [31] R. W. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operations Research*, 46:271–281, 1990.
- [32] M. Ferris, G. Deng, D. G. Fryback, and V. Kuruchittham. Breast cancer epidemiology: Calibrating simulations via optimization. *Oberwolfach Reports*, 2:9023–9027, 2005.
- [33] M. C. Ferris, J.-H. Lim, and D. M. Shepard. Optimization approaches for treatment planning on a Gamma Knife. *SIAM Journal on Optimization*, 13:921–937, 2003.
- [34] M. C. Ferris, J.-H. Lim, and D. M. Shepard. Radiosurgery treatment planning via nonlinear programming. *Annals of Operations Research*, 119:247–260, 2003.
- [35] M. C. Ferris, T. S. Munson, and K. Sinapiromsaran. A practical approach to sample-path simulation optimization. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 795–804, Orlando, Florida, 2000. Omnipress.

- [36] M. C. Ferris and M. M. Voelker. Fractionation in radiation treatment planning. *Mathematical Programming B*, 102:387–413, 2004.
- [37] D. E. Finkel. DIRECT optimization algorithm user guide. Technical Report CRSC-TR03-11, Center for Research and Scientific Computation, North Carolina State University, Raleigh, NC, 2003.
- [38] D. E. Finkel. *Global Optimization with the DIRECT Algorithm*. PhD thesis, North Carolina State University, 2005.
- [39] D. G. Fryback, N. K. Stout, M. A. Rosenberg, A. Trentham-Dietz, V. Kuruchittham, and P. L. Remington. The Wisconsin breast cancer epidemiology simulation model. *Journal of the National Cancer Institute Monographs*, 36:37–47, 2006.
- [40] M. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 53:199–248, 1994.
- [41] M. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002.
- [42] M. C. Fu and J. Hu. Sensitivity analysis for Monte Carlo simulation of option pricing. *Probability in the Engineering and Information Sciences*, 9(3):417–446, 1995.
- [43] M. C. Fu and J. Q. Hu. *Conditional Monte Carlo: Gradient Estimation*

and Optimization Applications. Kluwer Academic Publishers, Boston, MA, 1997.

- [44] S. Gabriel, R. W. Lau, and C. Gabriel. The dielectric properties of biological tissues: III. parametric models for the dielectric spectrum of tissues. *Phys. Med. Biol.*, 41:2271–2293, 1996.
- [45] F. Glover. Tabu search – part 1. *ORSA Journal on Computing*, 1:190–206, 1989.
- [46] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.
- [47] A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.
- [48] G. Gürkan, A. Yonca Özge, and S. M. Robinson. Sample-path optimization in simulation. In J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, editors, *Proceedings of the 1994 Winter Simulation Conference*, 1994.
- [49] G. Gürkan, A. Yonca Özge, and S. M. Robinson. Sample-path solution of stochastic variational inequalities, with applications to option pricing. In J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain,

- editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 337–344, 1996.
- [50] G. Gürkan, A. Yonca Özge, and S. M. Robinson. Solving stochastic optimization problems with stochastic constraints: An application in network design. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 471–478, 1999.
- [51] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Mateo, CA, 2000.
- [52] T. Homem-de-Mello. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13(2):108–133, 2003.
- [53] T. Homem-de-Mello. On rates of convergence for stochastic optimization problems under non-I.I.D. sampling. *submitted for publication*, 2006.
- [54] M. A. Hunt, T. E. Schultheiss, G. E. Desobry, M. Hakki, and G. E. Hanks. An evaluation of setup uncertainties for patients treated to pelvic fields. *International Journal of Radiation Oncology, Biology and Physics*, 32:227–33, 1995.

- [55] W. Huyer and A. Neumaier. Snobfit - stable noisy optimization by branch and fit. 2006. submitted.
- [56] K. Inoue, S. E. Chick, and C.-H. Chen. An empirical evaluation of several methods to select the best system. *ACM Transactions on Modeling and Computer Simulation*, 9(4):381–407, 1999.
- [57] M. F. Iskander and A. M. Tumeh. Design optimization of interstitial antennas. *IEEE Transactions on Biomedical Engineering*, 36(2):238–246, 1989.
- [58] D. R. Jones. The DIRECT global optimization algorithm. *Encyclopedia of Optimization*, 2001.
- [59] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, 1993.
- [60] P. Kim and Y. Ding. Optimal engineering system design guided by data-mining methods. *Technometrics*, 47(3):336–348, 2005.
- [61] S.-H. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM TOMACS*, 11:251–273, 2001.
- [62] S.-H. Kim and B. L. Nelson. Selecting the best system: Theory and

- methods. In *Proceedings of the 2003 Winter Simulation Conference*, pages 101–112, 2003.
- [63] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.
- [64] S. B. Kotsiantis and P. E. Pintelas. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):46–55, 2003.
- [65] M. Kubat, R. Holte, and S. Matwin. Detection of oil spills in radar images of sea surface. *Machine Learning*, 30:195–215, 1998.
- [66] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning, ICML'97*, pages 179–186, 1997.
- [67] H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, New York, 1978.
- [68] K. M. Langen and T. L. Jones. Organ motion and its management. *International Journal of Radiation Oncology, Biology and Physics*, 50:265–278, 2001.

- [69] A. Law and W. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, third edition, 2000.
- [70] R. M. Lewis and V. Torczon. Pattern search algorithm for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.
- [71] J. G. Li and L. Xing. Inverse planning incorporating organ motion. *Medical Physics*, 27(7):1573–1578, July 2000.
- [72] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Dace - a Matlab kriging toolbox, informatics and mathematical modelling, 2002.
- [73] O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT Press, Cambridge, Massachusetts, 2000.
- [74] O. L. Mangasarian. Support vector machine classification via parameterless robust linear programming. *Optimization Methods and Software*, 20:115–125, 2005.
- [75] M. Marazzi and J. Nocedal. Wedge trust region methods for derivative free optimization. *Mathematical Programming*, 91:289–305, 2002.
- [76] MATLAB. *User's Guide*. The MathWorks, Inc., 2007.

- [77] T. Mitchell. *Machine Learning*. McGraw–Hill, New York, 1999.
- [78] J. J. Moré. Recent developments in algorithms and software for trust region methods. In *Mathematical Programming: The State of the Art*, pages 258–287. Springer Verlag, 1983.
- [79] A. Niemierko. Optimization of 3D radiation therapy with both physical and biological end points and constraints. *International Journal of Radiation Oncology, Biology and Physics*, 23:99–108, 1992.
- [80] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, second edition, 2006.
- [81] D. Optiz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
- [82] R. Pasupathy and S. G. Henderson. A testbed of simulation-optimization problems. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 255–263, 2006.
- [83] E. L. Plambeck, B. R. Fu, S. M. Robinson, and R. Suri. Throughput optimization in tandem production lines via nonsmooth programming. In J. Schoen, editor, *Proceedings of the 1993 Summer Computer Simulation Conference*, pages 70–75, San Diego, California, 1993. Society for Computer Simulation.

- [84] E. L. Plambeck, B. R. Fu, S. M. Robinson, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75:137–176, 1996.
- [85] M. J. D. Powell. UOBYQA: Unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92:555–582, 2002.
- [86] M. J. D. Powell. The NEWUOA software for unconstrained optimization with derivatives. *DAMTP Report 2004/NA05, University of Cambridge*, 2004.
- [87] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [88] Y. Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics*, A7:799–811, 1978.
- [89] S. M. Robinson. Analysis of sample-path optimization. *Mathematics of Operations Research*, 21:513–528, 1996.
- [90] R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.
- [91] A. Ruszczyński and A. Shapiro. Stochastic programming models. In *Stochastic Programming*, pages 1–63. Elsevier, Amsterdam, Netherlands, 2003.

- [92] Th. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer. McGraw–Hill Book Company, New York, 2003.
- [93] W. Schlegel and A. Mahr, editors. *3D Conformal Radiation Therapy - A Multimedia Introduction to Methods and Techniques*. Springer-Verlag, Berlin, 2001.
- [94] L. M. Schmitt. Theory of genetic algorithms. *TCS: Theoretical Computer Science*, 259, 2001.
- [95] L. M. Schmitt. Theory of genetic algorithms II: Models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *TCS: Theoretical Computer Science*, 310, 2004.
- [96] M. Semini and H. Fauske. Applications of discrete-event simulation to support manufacture logistics decision-making: A survey. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 1946–1953, 2006.
- [97] A. Shapiro. Asymptotic behavior of optimal solutions in stochastic programming. *Mathematics of Operations Research*, 18:829–845, 1993.
- [98] A. Shapiro. Statistical inference of stochastic optimization problems. In

- S. P. Uryasev, editor, *Probabilistic Constrained Optimization: Methodology and Applications*, pages 91–116. Kluwer Academic Publishers, 2000.
- [99] A. Shapiro. Monte Carlo sampling approach to stochastic programming. In J. P. Penot, editor, *ESAIM: Proceedings*, volume 13, pages 65–73, 2003.
- [100] A. Shapiro. Monte Carlo sampling methods. In *Stochastic Programming*, pages 1–63. Elsevier, Amsterdam, Netherlands, 2003.
- [101] A. Shapiro and T. Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming: Series A*, 81(3):301–325, 1998.
- [102] D. M. Shepard, M. C. Ferris, G. Olivera, and T. R. Mackie. Optimizing the delivery of radiation to cancer patients. *SIAM Review*, 41:721–744, 1999.
- [103] M. Y. Sir, M. A. Epelman, and title = S. M. Pollock.
- [104] James C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, New York, 2003.
- [105] P. R. Stauffer, F. Rossetto, M. Prakash, D. G. Neuman, and T. Lee. Phantom and animal tissues for modeling the electrical properties of human liver. *Int. J. Hyperthermia*, 19(1), 2003.

- [106] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.
- [107] I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man and Communications, SMC-6*, pages 769–772, 1976.
- [108] J. Unkelback and U. Oelfke. Inclusion of organ movements in IMRT treatment planning via inverse planning based on probability distributions. *Institute of Physics Publishing, Physics in Medicine and Biology*, 49:4005–4029, 2004.
- [109] J. Unkelback and U. Oelfke. Incorporating organ movements in inverse planning: Assessing dose uncertainties by Bayesian inference. *Institute of Physics Publishing, Physics in Medicine and Biology*, 50:121–139, 2005.
- [110] L. J. Verhey. Immobilizing and positioning patients for radiotherapy. *Seminars in Radiation Oncology*, 5(2):100–113, 1995.
- [111] S. Webb. *The Physics of Conformal Radiotherapy: Advances in Technology*. Institute of Physics Publishing Ltd., 1997.
- [112] D. Yang, J. M. Bertram, M. C. Converse, A. P. O’Rourke, J. G. Webster, S. C. Hagness, J. A. Will, and D. M. Mahvi. A floating sleeve antenna yields localized hepatic microwave ablation. *IEEE Transactions on Biomedical Engineering*, 53(3):533–537, 2006.