Continuous optimization via simulation using Golden Region search

by

Alireza Kabirian

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Olafsson, Major Professor
Douglas D Gemmill
John K Jackman
Max D Morris
Jennifer Jane Blackhurst
David M Frankel

Iowa State University

Ames, Iowa

2009

To:

Cyrus, the Great

and

My Parents: Jamileh and Fathollah

Table of Contents

Acknowledgement

I would like to take this opportunity to express my thanks to those who helped me with conducting research and the writing of this thesis. First and foremost, I thank Dr. Sigurdur Olafsson for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Douglas D Gemmill, Dr. John K Jackman, Dr. Max D Morris, Dr. Jennifer Jane Blackhurst and Dr. David M Frankel.

Abstract

Simulation Optimization (SO) is the use of mathematical optimization techniques in which the objective function (and/or constraints) could only be numerically evaluated through simulation. Many of the proposed SO methods in the literature are rooted in or originally developed for deterministic optimization problems with available objective function. We argue that since evaluating the objective function in SO requires a simulation run which is more computationally costly than evaluating an available closed form function, SO methods should be more conservative and careful in proposing new candidate solutions for objective function evaluation. Based on this principle, a new SO approach called Golden Region (GR) search is developed for continuous problems. GR divides the feasible region into a number of (sub) regions and selects one region in each iteration for further search based on the quality and distribution of simulated points in the feasible region and the result of scanning the response surface through a metamodel. The experiments show the GR method is efficient compared to three well-established approaches in the literature. We also prove the convergence in probability to global optimum for a large class of random search methods in general and GR in particular.

Chapter 1: Introduction

In this chapter, we introduce concepts of Simulation Optimization (SO) problems/methods. SO methodologies have wide applications, going well beyond the wedding between (discrete event) simulation and optimization which is the focus of this research. Therefore, we begin with introducing and discussing the kind of problems that are similar to the major problem considered in this research. Then the concepts of discrete event simulation are presented and properties of SO problems are explained. Although chapter 2 will review the literature and problem background extensively, we briefly go over the literature and mention the shortcomings of the available methods in this chapter. At the end, the hypothesis of this dissertation is discussed and research goals are set out.

1. What is Optimization?

In simple word, optimization is a mathematical decision making process for finding the best option (decision) among a set of possible options. The best is defined as the option which satisfies an objective the most. Numerous types of optimization problems and methods exist in the literature. We very briefly classify these problems/methods in this section. Before that, we need to define components of an optimization problem.

1.1. Components of Optimization Problems

Optimization problems are made up of three basic ingredients:

1. An objective function which we want to minimize or maximize. For instance, in a manufacturing process, we might want to maximize the profit or minimize the cost. In fitting experimental data to a user-defined model, we might minimize the total deviation of observed data from predictions based on the model. In designing an automobile panel, we might want to maximize the strength.

2. A set (vector) of unknowns or variables which affect the value of the objective function. In the manufacturing problem, the variables might include the production volumes of each product in the manufacturing process. In the data-fitting problem, the unknowns are the parameters that define the model. In the panel design problem, the variables used define the shape and dimensions of the panel.
3. A set of constraints that allow the unknowns to take on certain values but exclude others. For the manufacturing problem, it does not make sense to produce a negative volume for a product, so we constrain all the production-volume variables to be non-negative. In the panel design problem, we would probably want to limit the weight of the product and to constrain its shape.

Now, the optimization problem is then: Find values of the variables that minimize or maximize the objective function while satisfying the constraints.

When the vector of decision variables is assigned some values, it is synonymously called a design, a design point, a solution or a point. If a solution satisfies all constraints, it is called feasible, otherwise it is infeasible. The set of all feasible points is called feasible region.

1.2. Classification of Optimization Problems

Optimization problems could be classified based on the type of their basic components, the type of functions and feasible region.

1.2.1 Objective Function

Almost all optimization problems have a single objective function. When they don't they can often be reformulated so that they do. The two interesting exceptions are:

1. No objective function. In some cases (for example, design of integrated circuit layouts), the goal is to find a set of variables that satisfies the constraints of the model. The user does not particularly want to optimize anything so there is no reason to define an objective function. This type of problems is usually called a feasibility problem.
2. Multiple objective functions. Often, the user would actually like to optimize a number of different objectives at once. For instance, in the panel design problem, it would be nice to minimize weight and maximize strength simultaneously. Usually, the different objectives are not compatible; the variables that optimize one objective may be far from optimal for the others. In practice, problems with multiple objectives are reformulated as single-objective

problems by either forming a weighted combination of the different objectives (e.g. utility method, goal programming,…) or else replacing some of the objectives by constraints.

There are two types of optimization:

1. Minimization of the objective function
2. Maximization of the objective function

However, Maximization and minimization problems are two sides of the same coin; by multiplying -1 to the objective function of a minimization (maximization) problem, the problem is converted into a maximization (minimization) problem, though the optimum value remains the same.

1.2.2 Decision Variables

Based on the type of each variable, optimization problems are of one of the following three classes:

1. Continuous: all decision variables are continuous.
2. Discrete: all decision variables take discrete values. Discrete variables could be classified into two groups: binary variables, integer variables. Based on another criterion, discrete variables are either quantitative or qualitative variables. A qualitative variable is a code which its magnitude is meaningless. E.g. a decision variable in a manufacturing system might be the selection of the type of transportation in the assembly line. There are three options available: conveyer, crane and cart. The transportation-type decision variable is a qualitative one and the modeler may arbitrarily assign code one, two and three to conveyer, crane and cart respectively.
3. Mixed (Integer) Variables: a combination of continuous and discrete variables.

Based on the type of the vector of decision variables, problems could be classified into the following groups:

1. With a single vector of decision variables
2. With “nested” vectors of decision variables (design configuration)

In design configuration problems (Pierreval and Paris 2003), there is a “parent” vector of decision variables; when certain decision variables take certain values, then a “child” vector of decision variables is defined under the value of that element. Many such “nested” decision variables could be defined.

1.2.3 Constraints

Based on the existence of constraints, there are two classes of problems:

1. Unconstrained: there are no constraints except the domain of decision variables.
2. Constrained: there is at least one constrain other than the domain of variables

Based on the type of constraints, there are two types of problems:

1. With equality constraints
2. Without equality constraints

1.2.4 Functions

Based on linearity of functions, there are two classes of problems:

1. Linear: the objective function and the functions in the constraints are all linear
2. Nonlinear: at least there is a nonlinear function in the objective or a constraint

Functions are either:

1. Convex or
2. Concave or
3. Non-convex and non-concave

Functions are either:

1. Continuous and twice differentiable or
2. Continuous and non-twice differentiable or
3. Non-continuous

Functions are either:

1. Deterministic: knowing the values of the decision variables, the function has a fixed value
2. Stochastic: there are uncontrollable random variables that affect the values of function. In other words, fixing the values of decision variables, the function is a random variable and a function of uncontrollable random variables.

Consequently, an optimization problem is called deterministic, if its objective function and the functions in constraints are all deterministic; otherwise it is stochastic optimization problem.

The closed form of each function based on decision variables and certain uncontrollable random variables is either:

1. Known: there exist a closed-form formula for the function or
2. Unavailable: there is no formula for the function, but fixing the decision variables, it could be numerically evaluated

1.2.5 Feasible region

There are two types of problems based on the type of feasible region:

1. With convex feasible region
2. With non-convex feasible region

1.3. Optimization Methods

There are two major groups of optimization methods in the literature for single objective deterministic optimization problems:

1. Mathematical programming
2. Heuristic Methods

Many methods are classified into mathematical programming group:

1. Linear programming methods: continuous variables with linear known and closed-form functions
2. Linear integer programming methods: discrete variables with linear, known and closed-form functions
3. Linear mixed integer programming methods: mixed integer variables and linear functions
4. Nonlinear programming: continuous variables and nonlinear functions
5. Dynamic programming: discrete variables

Heuristic Optimization methods are classified as:

1. Metaheuristic Methods
2. Heuristic methods with limited applicability

Metaheuristic methods don't need the closed-form formula of the objective function and the functions in the constraints. These methods are quite general and could be applied in almost all types of optimization problems. However metaheuristics are primarily used for combinatorial optimization problems which are discrete-variable optimization problems with compact but very large feasible

regions. Examples are Genetic Algorithm, Simulated Annealing, Tabu Search, Scatter Search, etc. We will review this class in chapter 2 more extensively.

On the other hand, there are certain classes of optimization problems which could be solved with specific heuristic methods. Often times, these special-purpose heuristic methods are more efficient than metaheuristics because they employ special properties of the problem structure

For stochastic optimization problems with known functions, Stochastic programming methods or heuristic methods such as Sample Path Optimization are usually used.

When one or more functions in a Stochastic Optimization problem are unknown, deterministic heuristic methods or numerical optimization methods could be employed.

2. Numerical Optimization of Expensive and Noisy Functions

Now we turn our attention to a subclass of optimization problems. In this section, we look into numerical optimization problems with expensive and noisy functions. These problems have wide applications. Hence, to illustrate where such concepts and methods apply, we start out this section by presenting a handful of examples.

2.1. Examples- Objective Functions

Consider a chemical reaction in which a certain amount of materials are mixed together to produce a final product. The best final product would be as much acidic (measured by PH) as possible. A chemist may want to conduct a number of lab experiments with different combinations of materials in order to find the combination resulting in the highest PH. Assume the test is costly and destructive, that is, the expensive consumed materials in an experiment could not be retrieved. From an Operations Research (OR) point of view, this is a numerical optimization problem with expensive Objective Function (OF). It is "numerical" because there is no a priory formula for the OF, but the OF could be evaluated point wise. It is "expensive" because the experiment is costly. Indeed, whenever the chemist wants to know the value of PH (OF) for a given combination of materials (decision variables, DVs), he/she must conduct an expensive lab experiment with the combination and find the numerical output value of the PH.

Kabirian and Hemmati (2006) address a conceptually similar problem. This time the noisy expensive experiment is a fluid dynamic submodel. Again there is an optimization model that this time has to find

the best development plans for an existing natural gas transmission pipeline network. The OF is the net present worth of the operating and capital costs of the development plans. The operating costs are functions of the pressure and mass flow rate of natural gas in the pipeline network. But the difficulty is that given the physical structure of a network, the flow of natural gas through the pipelines must be analyzed by a fluid dynamic simulation to know the values of pressure and mass flow rate affecting the OF. The computations required for a fluid mechanics simulation is huge and therefore the OF is deemed computationally expensive.

Consider another case from a news channel which says North Korea is testing a new long-range missile. The accuracy of a missile is the most important objective of testing a missile. More precisely, North Korea wants to minimize the distance between the designated target and the hit point of the missile. This distance is of course a function of launch conditions and control systems (DVs) and also weather situations (uncontrollable variables). The production and launch costs of a missile in this case may not be as expensive as political costs of testing threatening advanced military weapons. Politically, the costs of these tests include international sanctions or isolation. Indeed, North Korean scientists may not be able to test the missile very many times in order to find the best settings for the missile. In this case, the OF is deemed unknown again and is politically expensive. Since there are uncontrollable variables in the test, the OF is noisy too.

In another example, suppose a veterinarian is studying the effectiveness of a new medicine for a certain bovine disease. The medicine could be made with 20 different combinations of material. When an ill animal takes any of the possible medicines under study, it either dies or is cured. The goal is to find the combination (DV) resulting in maximum cure rate (OF). To test a combination, an ill buffalo is needed. Therefore there are two barriers in testing a medicine. First, a bad medicine may kill the animal and this is morally expensive. Second, the number of available ill buffalos may be limited, that is, the budget for evaluating the objective function is confined. In addition, the OF is noisy because some of the body conditions of buffalos are uncontrollable variables of these tests.

Similar concepts exist when the effect of pricing policies are tested in the real market. Suppose Wal-Mart is interested in determining the price (DV) of a product which would result in the highest profit (OF). The profit is defined as the total demand in a period of time times the fixed price of the item. Demand of this product is a complex function of the price and uncontrollable market conditions. The marketing department is unable to find a reliable formula for the demand and therefore a closed-form formula for the objective function does not exist. However, fixing the price in a certain Wal-Mart store

and selling the product for a period of time, an estimate of the demand could be found for that fixed price. In fact numerical values of the OF are available by testing a given price in a small real market. This OF is of course noisy and expensive. Noisy because of the uncontrollable market conditions and expensive because of the less profit made due to suboptimum values of price tested in a Wal-Mart store.

Sometimes, knowing the value of the objective function given a solution requires solving another optimization problem. In other words, there is a master optimization problem with an OF dependent on the optimal values of some slave optimization problem(s). Here is an example from supply chain management. In Figure 1.1, company A has two sources to obtain a raw material. Company B1 is one supplier of this material and is itself dependent on company B2 for its own raw material. Company C is also able to sell the raw materials needed by company A. Each company in the chain has a production line, a raw material inventory (I) and a stock of finished goods. Now, the master problem is that Company A has to find the best ordering policy from suppliers B1 and C (DVs) such that the risk of stock out at I_A is minimized (OF). The risk at inventory I_A is a function of optimal production schedules of B1, B2 and C. The optimization models of production scheduling in the aforementioned companies (slave problems) are dependent on the ordering policy of company A. Therefore, fixing the ordering policy of Company A, the risk is determined by solving slave optimization problems. Solving slave problems are computationally intensive and hence, the objective function of the master problem is expensive.

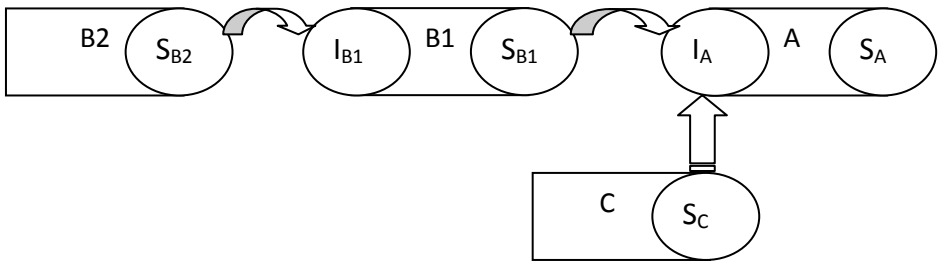


Figure 1.1: An example of supply chain

2.2. Examples- Functions in Constraints

From constraints perspective, the application is even further. There are some design optimization cases where knowing the feasibility of a design requires running a simulation model, an experiment, etc which are expensive and sometimes noisy.

One example of this situation is discussed in Kabirian and Hemmati (2007). They develop an optimization model for the design of natural gas transmission pipelines network in which a fluid dynamics sub-model must be run every time the optimization algorithm needs to know if a proposed design is feasible.

As mentioned in section one, an approach for transforming a multi-objective optimization problem into a single objective problem is to optimize one of the objectives while constraining the values of the others. If one or more of the objectives to be constrained are expensive, then knowing the feasibility of a solution in the resulting single objective problem is expensive. Recall the chemical experiment example in section 2.1. Suppose this time the chemist wants to minimize the viscosity of the product while again maximizing PH. In the transformed single objective optimization problem corresponding to this situation, the combination of materials resulting in minimum viscosity should be found such that the PH of the product is at least 12. Knowing if the PH of a combination is over 12 requires running an expensive experiment. In this case, both OF and constrains have expensive functions. Not surprisingly, there are also cases where although there exist a cheap-to-be-evaluated closed-form formula for the OF, one or more constraints are expensive.

Often times, the expensive functions in the constraints are noisy too. This means that given a solution, we are never sure if the solution is feasible unless we take infinite samples of the noisy function(s) in the constraint(s) which is practically impossible. This is a major issue in these kinds of problems because the optimum introduced by the optimization process might actually be infeasible!

However, since the constraints of any optimization algorithm could be incorporated into the objective function using penalty function methods, these cases might also be solved using expensive-function algorithms.

2.3. Properties of the problem

The examples provided in previous sections showcased the basic properties of the problem of our interest. The first difference between these problems and what we call ordinary optimization problems (like those problems solved by mathematical programming techniques) is that the closed form of the OF is not available. However, there exist a “routine” by which an (estimated) value of the OF for a known solution could be obtained. This routine could be a:

- black-box submodel such as:

    - a numerical analysis like a computer simulation or a numerical solver for a complex set of equations
    - an optimization submodel
    - a prediction model like an artificial neural network or an expert system
- real-system sampling such as
    - a pilot test or lab experiment
    - a full-scale real implementation
    - surveying

The **fundamental assumption** of the problem of our interest is that using the routine to evaluate the OF is monetarily expensive, time consuming, computationally intensive, practically dangerous or limited, or ethically unjustifiable.

In the main problem considered in this research, the OF is a function of two sets of variables:

- a vector of controllable variables, namely decision variables
- a vector of uncontrollable variables

When we don't have control on some variables, this means the objective function values obtained by fixing the decision variables would be noisy or stochastic. This noise is the second property which challenges the optimization algorithms.

3. What is Discrete-Event Simulation?

3.1. Principles and Definitions

Simulation is the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system that is represented (Banks 1998).

Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behavior of a system, ask "what if" questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modeled with simulation.

There are several concepts underlying simulation. These include system and model, events, system state variables, entities and attributes, list processing, activities and delays, and finally the definition of discrete-event simulation. Additional information on these topics is available from Banks, et al. (2000) and Law and Kelton (2000).

A model is a representation of an actual system. The model should be complex enough to answer the questions raised, but not too complex (Carson 2004).

Consider an event as an occurrence that changes the state of the system. In an example about a queue of customers in a bank (Figure 1.2), events include the arrival of a customer for service at the bank, the beginning of service for a customer, and the completion of a service.

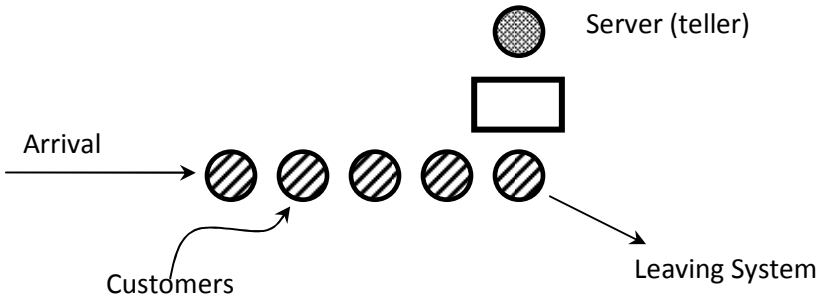


Figure 1.2: A bank system

Discrete-event models are dynamic, i.e., the passage of time plays a crucial role. Most mathematical and statistical models are static in that they represent a system at a fixed point in time. Consider the annual budget of a firm. This budget resides in a spreadsheet. Changes can be made in the budget and the spreadsheet can be recalculated, but the passage of time is usually not a critical issue.

The system state variables are the collection of all information needed to define what is happening within the system to a sufficient level (i.e., to attain the desired output) at a given point in time. The determination of system state variables is a function of the purposes of the investigation, so what may be the system state variables in one case may not be the same in another case even though the physical system is the same. Determining the system state variables is as much an art as a science. However, during the modeling process, any omissions will readily come to light. (And, on the other hand, unnecessary state variables may be eliminated.) In the bank example, the number of customers in the queue might be a state in a simulation study.

Having defined system state variables, a contrast can be made between discrete-event models and continuous models based on the variables needed to track the system state. The system state variables in a discrete-event model remain constant over intervals of time and change value only at certain well-defined points called event times. Continuous models have system state variables defined by differential or difference equations giving rise to variables that may change continuously over time.

An entity represents an object that requires explicit definition. An entity can be dynamic in that it "moves" through the system, or it can be static in that it serves other entities. In the example, the customer is a dynamic entity, whereas the bank teller is a static entity.

An entity may have attributes that pertain to that entity alone. Thus, attributes should be considered as local values. In the example, an attribute of the entity could be the time of arrival.

A resource is an entity that provides service to dynamic entities. The resource can serve one or more than one dynamic entity at the same time, i.e., operates as a parallel server. A dynamic entity can request one or more units of a resource. If denied, the requesting entity joins a queue, or takes some other action (i.e., diverted to another resource, ejected from the system). If permitted to capture the resource, the entity remains for a time, and then releases the resource. There are many possible states of a resource. Minimally, these states are idle and busy. But other possibilities exist including failed, blocked, or starved.

Entities are managed by allocating them to resources that provide service, by attaching them to event notices thereby suspending their activity into the future, or by placing them into an ordered list. Lists are used to represent queues. Lists are often processed according to FIFO (first-in first- out), but there are many other possibilities. For example, the list could be processed by LIFO (last-in first-out), according to the value of an attribute, or randomly, to mention a few. An example where the value of an attribute may be important is in SPT (shortest process time) scheduling. In this case, the processing time may be stored as an attribute of each entity. The entities are ordered according to the value of that attribute with the lowest value at the head or front of the queue.

An activity is a duration of time whose duration is known prior to commencement of the activity. Thus, when the duration begins, its end can be scheduled. The duration can be a constant, a random value from a statistical distribution, the result of an equation, input from a file, or computed based on the event state. For example, a service time may be a constant 10 minutes for each entity; it may be a random value from an exponential distribution with a mean of 10 minutes, etc.

A delay is an indefinite duration that is caused by some combination of system conditions. When an entity joins a queue for a resource, the time that it will remain in the queue may be unknown initially since that time may depend on other events that may occur.

Discrete-event simulations contain activities that cause time to advance. Most discrete-event simulations also contain delays as entities wait. The beginning and ending of an activity or delay is an event.

A discrete-event simulation model can now be described as one in which the state variables change only at those discrete points in time at which events occur. Events occur as a consequence of activity times and delays. Entities may compete for system resources, possibly joining queues while waiting for an available resource. Activity and delay times may "hold" entities for durations of time. A discrete-event simulation model is conducted over time ("run") by a mechanism that moves simulated time forward. The system state is updated at each event along with capturing and freeing of resources that may occur at that time.

3.2. Advantages and Disadvantages

Compared to analytical methods, simulation methods have some advantages and disadvantages. Some advantages are as follows:

1. Simulation lets the user test every aspect of a proposed change or addition without committing resources to their acquisition.
2. By compressing or expanding time simulation allows the modeler to speed up or slow down phenomena so that you can thoroughly investigate them.
3. Simulation studies aid in providing understanding about how a system really operates rather than indicating an individual's predictions about how a system will operate.
4. Visualizing the system, many simulation packages allow you to see your facility or organization actually running.
5. When analytical methods are unable to model stochastic systems due to complexity, simulation is probably the best and only remaining alternative.
6. Compared to analytical methods, when using simulation, there is less need to accept unrealistic assumptions in modeling.

However, simulation has some disadvantages too:

1. Model building requires special training.
2. Simulation results may be difficult to interpret.
3. Simulation modeling and analysis can be time consuming and expensive.

4. Simulation based Optimization

In this report, by simulation, we mean discrete-event simulation unless otherwise stated. Simulation based optimization has a number of synonyms like “simulation optimization” or “optimization via simulation”.

Simulation optimization is a special case of numerical optimization for noisy and expensive functions in which, fixing a design point, discrete event simulation is used for estimating the objective function and/or functions in constraints. Simulation is computationally expensive and the output of simulation which estimates the objective function of the simulated designed point is stochastic (noisy).

The advantages of simulation optimization over other optimization models roots in the advantages of discrete-event simulation analysis which were stated in section 3.2. Application of simulation based optimization is as wide as applicability of discrete event simulation analysis. In the bank example of Figure 1.2, the bank manager might wish to know how fast the service (DV) should be such that the average waiting time of customers in line (OF) is minimized.

To explain the power of simulation optimization, we explain a recent research work by Ghosh et. al. (2009).The research is about a simple model of a closed loop supply chain in which a producer manufactures new products to order (Figures 1.3 and 1.4). Some new products are returned by the customers after evaluation. In this model, any new product can get returned with probability β. These returned products can no longer be sold as new. Instead, they are inspected, refurbished and placed into inventory to be resold as “refurbished” products. The producer is a price-taker in the market for new products, but it can set the relative price, ρ , of refurbished products as a fraction of the price of new products. In this model, the demand for new and refurbished products follow to Poisson processes with the rates $\lambda_N(\rho)$ and $\lambda_R(\rho)$ which are known functions of ρ . The time required to produce a new product is exponentially distributed with rate μ and the manufacturing server is not allowed to idle unless the queue is empty. When a demand for a refurbished item arrives, if such a product is available in inventory then the demand is satisfied; otherwise, the customer is lost. The goal in this research is to carry out a heavy traffic analysis of the system, and find asymptotically optimal service rate under

optimal prices which minimizes a suitable cost function. This goal is achieved after a lengthy statistical, mathematical and optimization analysis.

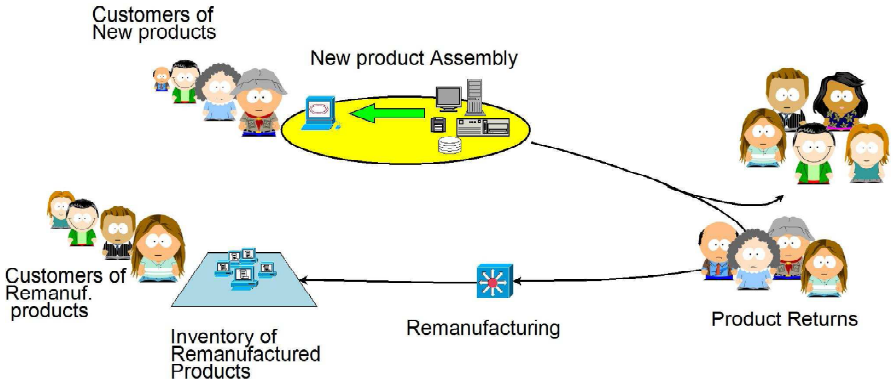


Figure 1.3: A schematic view of the manufacturing and remanufacturing markets for a product

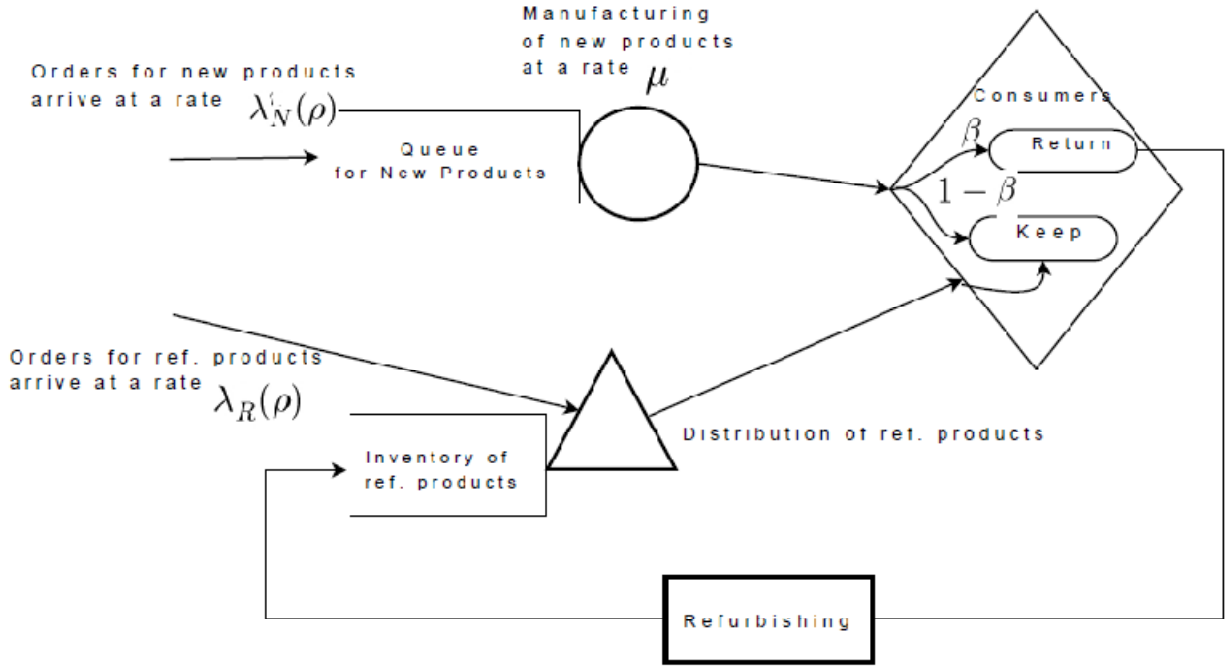


Figure 1.4: A flow diagram of the manufacturing and remanufacturing markets for a product

While the analysis of the research is very impressive, there are a number of strong assumptions in modeling. Most importantly, we have no example to believe that the production time in a real manufacturing process is exponentially distributed. For example, exponential service in a computer making company like Dell says that the conditional probability that we need to wait, say, more than another 10 hours before the manufacturing process of a computer is completed, given that the process has not yet finished after 30 hours, is no different from the initial probability that we need to wait more

than 10 hours for the process to end. In other words, when 30 hours of work has been done on a computer but the production is not yet complete, these 30 hours of work have been useless and production has not progressed at all! This can't be realistic. Another strong assumption is that unmet demand is lost. In reality, a fraction of the unmet demands of new (refurbished) computers could be satisfied by refurbished (new) product. The researcher(s) also assume that remanufacturing is done instantly which is another restricting assumption.

There are many more features of the real markets that have also been ignored in this model for simplicity. The manufacturing company might have different products whose demands are correlated. The activity of competitors also affects the price of the product in the market and the level of demand. To summarize, real-world situations are a lot more complex than simplifying and sometimes unrealistic assumptions used in these kinds of analytical modeling.

Simulation optimization along with modern computing power is an answer to modeling complex world and addressing aforementioned criticisms. When simulating a system, there is no restriction on the distributions, relationship of the components of the system, relations between the system and beyond the boundaries of the system. Simulation can model a system with as much details, realities and complexities as the modeler wants and is satisfied with; hence, with fast computational resources, simulation optimization could solve ***any*** real stochastic complex optimization problem. This is the power of simulation optimization.

In the literature, many optimization methods have been developed since the early days of the emergence of modern operations research after the Second World War. However, most of the attention of the researchers in optimization community has been focused on the problems and methods with available or cheap objective functions. Among these methods, linear, integer and nonlinear programming methods are the early contributions. By late 1980s and early 1990s, metaheuristic methods such as genetic algorithm, simulated annealing, tabu search and scatter search were proposed to address complex objective functions with combinatorial feasible regions. Since these methods are "black-box" procedures, they only use numerical values of the complex objective function without caring about its form or derivatives. Therefore the metaheuristics are considered plausible SO methods. Also, a large number of heuristic methods have been proposed that are black box (or numerical) in the way they treat the objective function. However, the key issue is that the complexity of a function which is one of the underlying motivations of the development of many of the heuristic methods does not necessarily imply the expensiveness of the function. Nevertheless, the metaheuristic methods are still

among popular options for SO problems. Ordinarily, a “good” optimization algorithm is the one that requires less computation time for optimization. But in simulation optimization, a “good” optimization algorithm is probably the one that requires less objective function evaluations to find the optimum.

Looking more carefully to the literature, most of the proposed practical SO methods have a core iterative search based strategy which evaluates available information of past searches (if any) in each iteration, propose new candidate solution(s), and simulate these candidate(s). Therefore, the total computational time in each iteration can be divided into two parts:

1. The time required to propose new candidate solution(s) , and

2. The simulation run-time for the proposed candidates.

In the first part, the methods use some intelligent strategy to propose new candidate solution(s), and in the second part, the objective function(s) of these candidate solution(s) are evaluated. In the case of deterministic optimization problems where a closed-form objective function is available or whenever the objective function is cheap, the time required for the second part (objective function evaluation) is usually negligible as opposed to the time of the first part of proposing new solution(s). In contrast, when a simulation run is used to estimate the objective function, the time of the first part could be much less than the second part because running a simulation model for real systems is usually computationally time-consuming. It is therefore reasonable to measure the efficiency of an SO algorithm in terms of the number of simulation runs required and design the algorithm such that as few as possible simulation runs are needed.

Many current methods for SO have originally been developed for problems with available closed-form objective function. Therefore, since in the deterministic context the time of evaluating the closed-form objective function could be considered negligible, these methods may not carefully allocate the simulation runs when applied directly to simulation optimization problems. In other words, they are not efficient in proposing new solutions, that is, allocating simulation runs, and hence require a large amount of time to simulate each of these solutions. This fact which is the missing point of the literature motivates the development of methods that make better use of the past information of simulated points in a search based strategy. In this research, we will try to bridge this quite wide **research gap** by proposing a new expensive-function method that is more specifically designed for SO problems.

To improve the efficiency in the allocation of simulation runs, we believe a "good" SO optimization algorithm may use the following simple principles to reduce the number of objective-function evaluations:

1. It should be reluctant to evaluate the objective function of a single solution point more than once.
2. It should be reluctant to evaluate the objective function of a solution point that is so close to some other previously simulated points that there is no significant difference between the designs corresponding to these points. This is primarily germane to continuous problems.
3. It should control the amount of search in the neighborhood of already found good solution points and avoid excessively searching the neighborhood of local optima.
4. When searching new local optima in unvisited areas of the feasible region, the search strategy should directly take into account the size of those unvisited areas.

Accounting for these characteristics requires more computational memory to store past information and control over past searches. This translates into more computational burden for what we referred to above as the first part of search based methods. On other hand, we content that this increased "thinking" and higher computational overhead could be justified by significantly less cost in the second part if more appropriate candidate solutions are suggested in the first part and hence fewer simulation runs are required. This fact usually justifies more complex methods which are more conservative in proposing new candidates. If the simulation run time is large enough, we believe that the best methods are those that try to extract and use the valuable information obtained right after each simulation run before proposing a new candidate or at least their population of proposed candidates in each iteration is as little as possible (with a conservative perspective). It is noteworthy that these four desired characteristics or principles are not the only desired characteristics of SO algorithms, and others may be more appropriate in certain applications. However, as we will see in the algorithm proposed in this research, following these principles appears to result in an effective search algorithm.

5. Hypothesis of This Research

In this dissertation, we introduce Golden Region search as a new SO method to bridge the research gap explained in section 4. The hypothesis that is studied in this work is the following:

In comparison to other methods available in the literature, the Golden Region Search method, for some classes of simulation optimization problems, needs fewer number of simulation runs to reach a given neighborhood of the objective function of the global optimum.

We first define a large subclass of Random Search methods called Probabilistic Search. Then we prove that when Probabilistic Search methods are merged with Ranking and Selection methods, the convergence of the Hybrid method is guaranteed. Golden Region method is a special case of Probabilistic Search methods. Therefore, this method converges too. We design an experiment and compare the efficiency of Golden Region method with other methods in the literature to evaluate the hypothesis of this research.

6. Organization of This Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews the types of SO problems and provides a brief description of the available methods in the literature. Chapter 3 discusses the so called Probabilistic Search methods as a sub class of Random Search Methods. We prove the convergence of these methods when they are merged with Ranking and Selection procedures. In chapter 4, the new method of this research called Golden Region Search is introduced. We prove that Golden Region search is classified into the group of Probabilistic Search methods and consequently, the convergence results proven in chapter 3 is applicable for Golden Region method as well. Numerical results of chapter 5 provide further support for the applicability of Golden Region method and help find the kind of SO problems for which Golden Region works well. In chapter 6, we discuss the contribution of this report, its shortcomings and research lines for future work.

Chapter 2: Problem Background

In this chapter, a review of the literature is presented. First, SO problems are classified. There are many papers in the literature which classify SO methods. We provide a short review of these survey articles. Then, a classification of the methods of SO is presented and the main ideas and procedures of each method are briefly explained. At the end, we review commercial SO software packages available in the market.

1. Simulation Optimization Problems

In chapter 1, a general classification of optimization methods was presented. SO problems are classified almost in the same way used for general optimization problems. Again, the problems are classified as either single-objective or multi-criteria. The problems are either with or without stochastic inequality constraints. The DVs are either qualitative, quantitative or a mixture of the two. Or, DVs are either discrete, continuous or mixed.

An optimization problem can be modeled in different ways which would result in same optimal values. The major difference between different models is the time required for finding the optimum (efficiency). Defining decision variables in an optimization problem is one step of modeling which can have a huge impact on efficiency. The problems in SO can be classified into the following three groups based on modeling of decision variables:

1. With a single binary decision variable for each design point (selection of the best)
2. With a single vector of decision variables
3. With “nested” vectors of decision variables (design configuration)

In selection of the best (SB) problems, there are a number of designs where a binary decision variable is attached to each design, that is the design is either selected or not.

In problems with a single vector of decision variables, similar decision variables characterize the designs. Assigning feasible values to decision variables in the vector means creating a design. Each decision variable could be either continuous or discrete. If the designs could be enumerated, a binary decision variable could then be attached to each design, either selection or not, and consequently the problem would be solvable with SB methods, though sometimes this approach is inefficient.

In design configuration problems (Pierreval and Paris 2003), there is a "parent" vector of decision variables; when certain decision variables take certain values, then a "child" vector of decision variables is defined under the value of that element. Many such "nested" decision variables could be defined. Of course, the decision variables could be "pooled" in these cases and the binary SB variables are attached, but this again may be inefficient.

2. Methods of Simulation Optimization

The literature of SO methods dates back to early years of the second half of 20th century. In two seminal papers, Robbins and Moreno (1951) and Kiefer and Wolfowitz (1952) introduced Stochastic Approximation as an early class of SO methods. Since then, many researchers have proposed methods for dealing with the unique challenges in SO problems. When computing power increased dramatically in the last decades of 20th century, discrete-event simulation became a more plausible option for analyzing stochastic systems. Consequently, applicability of simulation triggered the need for optimization methods which could find the best when simulation evaluates the quality of the best. In the last 20 years, the literature of SO has grown very fast. Many new methods have recently been introduced-some of which efficient, some mathematically elegant, some good for commercial softwares and still some just for scientific publication.

2.1. Survey Articles in the Literature

Many people have attempted to classify methods of SO and provide a survey of the literature. Authors of survey papers have more or less similar classifications of the methods, but with emphasis on certain method of their interest.

Carson and Maria (1997), classify the methods into the following 6 groups:

1. Gradient-based Methods
2. Stochastic Optimization

3. Response Surface Methodology
4. Heuristic Methods
5. Combined Methods (A-teams)
6. Statistical methods

For gradient based group, methods of estimating gradient have been mentioned in the article without actually surveying methods which use these gradients. Among heuristic methods, the authors consider Genetic Algorithm, Evolutionary Strategies, Simulated Annealing, Tabu Search and Nelder-Mead Search as the most important.

In 1998, three chapters of Handbook of Simulation edited by Banks (1998) discuss SO methods. Kleijnen (1998) introduces experimental designs for sensitivity analysis, optimization and validation of simulation models; Goldsman and Nelson (1998a) review Ranking and Selection methods and Multiple Comparison techniques. Andradottir (1998a and 1998b) classifies the rest of SO methods into two groups, one for discrete problems and another for continuous decision variables. In continuous part, she mentions Sample Path Optimization and Stochastic Approximation along with gradient estimation routines such as Finite Difference, Perturbation Analysis and Likelihood Ratio Method. In discrete part, methods such as Random Search, Stochastic Ruler Algorithm and Simulated Annealing are reviewed. In Goldsman and Nelson (1998b) and Andradottir (1998b), a summary of the chapters in this handbook are published at Winter Simulation Conference (WSC) 98.

In the following WSC, Azadivar (1999) looks at the literature of SO methods with a broader view. He classifies the problems into single-objective and multi-criteria. He also distinguishes between the problems with quantitative and qualitative variables. For single-objective methods, Azadivar (1999) uses a classification similar to Carson and Maria (1997). Swisher and Jacobsen (1999) completes Azadivar's work by reviewing statistical methods. They classify these methods this way:

1. Ranking and Selection
    a. Indifference Zone
    b. Subset Selection
2. Multiple Comparisons
    a. Paired-t, Bonferroni, all-pairwise comparisons
    b. All-pairwise multiple comparisons
    c. Multiple Comparison with a Control

   d. Multiple comparisons with the best
3. Unified Methods

Swisher et. al. (2000) reviews SO methods in WSC 00 and classifies the methods into continuous (gradient and non-gradient) and discrete settings. Stochastic Approximation is mentioned as a gradient method. For non-gradient continuous methods, the authors review Nelder-Mead and Hooke and Jeeves algorithms. In discrete part, statistical methods and heuristic approaches like Ordinal Optimization, Simulated Annealing, Genetic Algorithm, Stochastic Search and Nested Partitions are mentioned.

In 2001, Fu (2001) provides a comprehensive review of SO methods. Four Classes of methods introduced in this paper are Statistical Methods, Metaheuristics, Stochastic Optimization and Other Methods. Goldsman and Nelson (2001) update review of statistical methods of SO in their paper. Sequential Indifference-Zone Selection approaches are also reviewed in Pichitlamken and Nelson. (2001). Fu (2002) has a similar classification of SO methods as Fu (2001).

In WSC 02, two papers review available methods. Olafsson and Kim (2002) classifies the methods into continuous and discrete problems. In continuous part, methods such as Stochastic Approximation, Sample Path Optimization and Response Surface Methodology are mentioned. Methods in discrete part are classified into three groups: Statistical Methods, Stochastic Search and Metaheuristics. Magoulas et. al. (2002) has a short review of global search methods such as Simulated Annealing, Genetic Algorithm and Particle Swarm.

Gosavi (2003) is a comprehensive book which reviews SO methods. In this book, methods are classified into two groups, Parametric Optimization and Control Optimization. In Control Optimization, the book focuses on Dynamic Programming and Reinforcement Learning. In Parametric Optimization, discrete methods such as Statistical Procedures and Metaheuristics and continuous methods including gradient and non-gradient approaches are reviewed. In WSC 01, 03, 04 and 06, April at. al. (2001, 2003, 2004, 2006) provide a short review of classical SO methods directed toward advertising new capabilities of OptQuest software. In WSC 05, Fu et. al. (2005) presents a similar review paper.

Five chapters of Elsevier Handbooks in Operations Research and Management Science-Simulation have been assigned to review SO methods (Henderson and Nelson 2006). Kim and Nelson (2006) reviews Statistical Methods similar to many other review papers of these methods. Barton and Meckesheimer (2006) recognizes Metamodel methods as a promising class of global SO approaches. Metamodels are classified in this paper into these categories:

- Response Surface Metamodels
- Regression Spline Metamodels
- Spatial Correlation (Kriging) Metamodels
- Radial Basis Function Metamodels
- Neural Network Metamodels

Fu (2006) reviews gradient estimation methods that are useful when ideas of mathematical programming methods are employed in SO. Andradottir (2006) review Random Search methods of SO and Olafsson (2006) puts more emphasis on Metaheuristics and their role in SO. Kim (2006) also reviews Gradient methods of SO in WSC 06. Kim and Nelson (2007) also updates the list of Statistical Methods of SO with new advances in these methods.

One of the most recent review papers of SO methods is Fu et. al. (2008) where methods are categorized into the following groups:

- Sample Path Optimization
- Sequential Response Surface Methodology
- Stochastic Approximation
- Deterministic metaheuristics

2.2. Available Methods

We don't intend to review all the methods of SO in this section, but we try to mention main ideas of some. The classification we use for this purpose is the following:

1. Single-Objective Methods with Quantitative Variables
    a. Gradient Methods
    b. Statistical Methods
    c. Heuristic Methods
    d. Other Methods
    e. Mixed Methods
2. Single-Objective Methods with Qualitative Variables
3. Multiresponse Methods

2.2.1 Single-Objective Quantitative-Variable Methods

2.2.1.1 Gradient based Methods

In mathematical programming, there are many methods which use derivatives of the objective function to reach a (local) optimum. Hill Climbing methods (Luenberger 1984) are probably among the most well known approaches in which a search path starts from somewhere in the feasible region and moves in direction of the derivative of objective function. Under some regularity conditions for the objective function, these methods are guaranteed to reach a local optimum.

In SO, the ideas of using derivatives in mathematical programming have been the basis of gradient based methods. The challenge in SO is that the objective function is not known and hence a closed-form formula for derivative of the objective function does not exist. The remedy is that the derivatives could still be estimated for each point. Then these estimated derivatives could be used in math programming methods.

There are many methods in the literature which approximate derivatives of the objective function using simulation. Some of these methods are "Black-box" in nature which means they can estimate derivatives without looking into how simulation transforms the inputs to the outputs. Some others, however, are not as general as former methods and applicable to certain classes of SO problems. We first review gradient estimation methods and then explain some gradient based SO procedures.

- **Gradient Estimation Methods**
    1. **Finite Difference**

This method is the simplest way to estimate gradient. Let $f(\theta)$ be a deterministic objective function and $\theta = (\theta_1, \theta_2, \dots, \theta_d)$ be the vector of decision variables. Then the derivative of this function with respect to θ_i is defined as $f_i'(\theta) = \lim_{c\to 0} \frac{f(\theta+ce_i)-f(\theta)}{c} \quad \forall\, i = 1,2, \dots, d$ where e_i is ith unit vector and c is the step size. Now let $\hat{f}$ be the output of simulation estimating the objective function. The gradient could now be estimated using forward or central finite difference. Using forward method, the estimated derivative is defined as $\hat{f}_i'(\theta) = \frac{\hat{f}(\theta+ce_i)-\hat{f}(\theta)}{c} \quad \forall\, i = 1,2, \dots, d$. Indeed, forward method uses $d+1$ simulation runs to estimate the objective function at a given point. Using central method, the estimated derivative is $\hat{f}_i'(\theta) = \frac{\hat{f}(\theta+(c/2)e_i)-\hat{f}(\theta-(c/2)e_i)}{c} \quad \forall\, i = 1,2, \dots, d$ where $2d$ simulation runs are consumed.

Although forward method seems more economical in terms of using simulation runs in higher dimensions, central method has better statistical properties (Andradottir 1998a).

2. Simultaneous Perturbations

The perturbation term in the name of this method is because the objective function is somehow "perturbed". Spall (1992) proposed a version of this method which consumes less simulation runs than finite difference (Gosavi 2003).

The gradient in a fixed point $\theta = (\theta_1, \theta_2, \dots, \theta_d)$ is computed as follows:

$$F^+ = \hat{f}\left(\theta_1 + \frac{cB_1}{2}, \theta_2 + \frac{cB_2}{2}, \dots, \theta_d + \frac{cB_d}{2}\right)$$

$$F^- = \hat{f}\left(\theta_1 - \frac{cB_1}{2}, \theta_2 - \frac{cB_2}{2}, \dots, \theta_d - \frac{cB_d}{2}\right)$$

$$\hat{f}_i'(\theta) = \frac{F^+ - F^-}{cB_i} \quad \forall\, i = 1,2, \dots, d$$

where B_i is a Bernoulli random variable which takes 1 and -1 equally likely. In fact Simultaneous Perturbations method requires only two simulation run to estimate the gradient in a given point.

3. Frequency Domain Analysis

In this method, the value of decision variable (or decision vector in general) is oscillated during simulation run to evaluate the sensitivity of the objective function to the oscillations and get an estimate of the gradient (Schruben and Cogliano 1991). In other words, to approximate the gradient at $\theta = (\theta_1, \theta_2, \dots, \theta_d)$, the value of the ith decision variable at time t of the simulation clock is set to $\theta_i(t) = \theta_i + \alpha_i \sin(\omega_i t)$ where $\omega = (\omega_1, \omega_2, \dots, \omega_d)$ and $\alpha = (\alpha_1, \alpha_2, \dots \alpha_d)$ are vectors of frequency and range of oscillation. Then the objective function is approximated with a polynomial function around θ using the reactions of the objective function to the oscillation of the decision vector. The approximated polynomial function could then be used in estimating the derivatives. See Andradottir (1998) for more details and a discussion about how the parameters of this method could be chosen.

4. Perturbation Analysis

Perturbation Analysis is another class of gradient estimation methods whose the most well-known procedure is Infinitesimal Perturbation Analysis (IPA). In IPA, all partial gradients are estimated in a

single simulation run. The main idea is that if a decision variable is perturbed infinitesimally, the sensitivity of the objective function could be evaluated through tracing the propagation of perturbation. This sensitivity is a function of the fraction of propagation which is wiped off before being effective on the objective function. IPA assumes that an infinitesimal perturbation in a decision variable does not change the chorological order of the events in simulation, but only changes the time of some events softly. In addition to IPA, there are other types of this method like Smoothed Perturbation Analysis.

The advantage of Perturbation Analysis is that it needs only one simulation run to estimate the gradient. However, the applicability of this method is dependent on the type of stochastic system and is not generic. To see an example of how this method works in optimization of a steady-state queue with a single server, see Chong and Ramadge (1992 and 1993). For a discussion about this method, see Andradottir (1998).

5. Likelihood Ratio (Score Function)

Likelihood Ratio method (also called Score Function) involves expressing the gradient as an expectation that can be estimated via simulation (Glynn 1987). Andradottir (1998) illustrate this method as follows. Consider SO problem $\min_{\theta\in\Theta} f(\theta) = \mathrm{E}\,[X(\theta)]$ where $X(\theta)$ is a random variable with density and cumulative functions f_θ and F_θ respectively. Assuming the possibility of interchanging gradient and integral, we have

$$f'(\theta) = \nabla f(\theta) = \nabla \int x f_\theta(x)dx = \int x \nabla f_\theta(x)dx\ .$$

In Likelihood Ratio, the gradient $f'(\theta)$ should be expressed in such a way that could be estimated via simulation. This can be accomplished by multiplying and dividing by $f_\theta(x)$ in the above integral, yielding:

$$f'(\theta) = \int x \frac{\nabla f_\theta(x)}{f_\theta(x)} f_\theta(x)dx = \mathrm{E}\left[X(\theta)\frac{\nabla f_\theta(X(\theta))}{f_\theta(X(\theta))}\right].$$

Therefore, one can generate independent observations $X_1(\theta), X_2(\theta), \dots, X_N(\theta)$ from the distribution F_θ and use $\hat{f}'(\theta) = \sum_{j=1}^{N} X_j(\theta)\frac{\nabla f_\theta\left(X_j(\theta)\right)}{N f_\theta\left(X_j(\theta)\right)}$ as the estimate of the gradient.

The likelihood method may not be applied in general to all SO problem. This is because simulation in this method is not viewed as a black-box and the applicability of the method depends on the stochastic system of interest.

- **Gradient-based Methods**

We just mention two methods in this category.

1. **Stochastic Approximation**

The first Stochastic Approximation method was proposed by Robins and Monro (1951). This method was used to find the roots of a noisy function. The generalized Stochastic Approximation algorithm for SO problem $\min_{\theta\in\Theta} f(\theta)$ is as follows (Andradottir 1998):

Algorithm 1.1: *Stochastic Approximation Path*

1. *Select:*
   a. *an initial guess* $\theta_1 \in \Theta$,
   b. *a gain sequence* $\{a_n\}$ *of real numbers where* $\sum_{n=1}^{\infty} a_n = \infty$ *and* $\sum_{n=1}^{\infty} a_n^2 < \infty$,
   c. *a termination condition for the algorithm.*
2. *Set* $n = 1$.
3. *Find an estimate of* $f'(\theta_n)$ *using a gradient estimation method and denote it by* $\hat{f}'(\theta_n)$.
4. *Set* $\theta_{n+1} = \pi_\Theta\left(\theta_n - a_n\hat{f}'(\theta_n)\right)$ *where* $\pi_\Theta(\theta) \in \Theta$ *,for all* $\theta \in \mathbb{R}^d$ *, is the closest point to* θ *such that when* $\theta \in \Theta$ *, then* $\pi_\Theta(\theta) = \theta$.
5. *If the termination condition holds, return* θ_{n+1} *as the optimum and exit the algorithm; otherwise let* $n = n + 1$ *and go to step 3.*

For a discussion about choosing the parameters of this algorithm, see Andradottir (1998a) and the references therein. For the necessary and sufficient conditions of convergence of this method, see Chen (2005).

In step 3 of the algorithm, any gradient approximation method could be used. For instance, Spall (1998) reviews Stochastic Approximation methods using Simultaneous Perturbation. Most of the methods in Spall (1998) need two simulation runs to estimate the gradients. Spall (1997) also provides a Stochastic Approximation with Simultaneous Perturbation which needs only one simulation run per gradient estimation. Stochastic Approximation has also been used for discrete problems (Gerencsér et. al. 1999, 2001 and 2004). Convergence analysis of Stochastic Approximation has been analyzed in Chen et. al. (2005).

2. **Stochastic Quasi-Newton**

Deterministic Quasi-Newton (QN) method has shown to be efficient in locating a local optimum of a non-linear function (Nocedal and Wright 1999). This method uses the first derivatives of the objective function to find the second derivatives. Using the second derivatives, an (approximated) Hessian matrix is constructed and is used to find direction of search toward a local optimum. Stochastic QN is an adapted version of Deterministic QN for SO problems. Safizadeh and Signorile (1994) used QN method to accelerate the convergence of Response Surface Methodology in the proximity of the optimum. In 1997, Kao et. al. use a complete version of Stochastic QN method in SO for the first time in the literature (Kao et. al. 2005).

2.2.1.2 Statistical Selection of the Best

When the number of design points are very limited, say less than 20, statistical methods of SO are useful. In these methods, all designs are simulated at least once. This is the first difference between statistical methods and heuristic methods in which only a subset of all designs are simulated. The second difference is that decision variables in Selection of the Best are binary, either we select a design or not; but in heuristic methods, decision variables attribute features of a system. Many methods fall in the category of statistical methods such as Subset Selection, Indifference Zone procedures and Multiple Comparison Methods. We explain the ideas of some methods here.

- **Subset Selection**

In subset selection, a subset of the set of design options is selected such that the probability of selecting the best design exceeds a pre-specified minimum. Mathematically, if there are k different designs, the jth of which has mean objective function μ_j such that, unknown to us, $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_k$ (i.e. design one is the best assuming a minimization problem), the goal is then to obtain a subset $\mathrm{I} \subset \{1,2,\ldots,k\}$ that satisfy $\Pr\{1 \in \mathrm{I}\} \geq 1-\alpha$ where $1/k < 1-\alpha < 1$. Ideally, $|\mathrm{I}|$ is small, the best case being $|\mathrm{I}| = 1$. Gupta (1965a and 1965b) developed the first method of Subset Selection. The limitation of this method is its fundamental assumptions which require the output of simulation to be normally distributed with equal variance for all designs. Most of the other methods of Subset Selection try to relax these fundamental assumptions (see Chang and Huang 2001 or Futschik and Pflug 1995). For instance, Chang and Huang (2001) develop a Subset Selection method with unequal variances in which the selection criteria are different functions of mean and standard deviations of the design set. For a comparison of Subset Selection methods and Bayesian Selection methods, see Gupta and Miescke (2002).

- **Indifference Zone**

A disadvantage of subset selection procedure is that the retained set I may, and likely will, contain more than one system design. Indifference Zone (IZ) techniques have been proposed to address this problem by softening the goal. In these methods which only select one design, the system one is selected with at least $100(1-\alpha)\%$ chance if $\mu_2 - \mu_1 \geq \gamma$ where $\gamma > 0$,the so called IZ parameter, is the smallest difference the experimenter feels is worth detecting. Mathematically, IZ methods guarantee $\Pr_{\mu_2-\mu_1\geq\gamma}\{\text{select } 1\} \geq 1-\alpha$.

Many different IZ methods have been proposed in the literature. Bechhofer (1954) proposed the very first one under very strong assumptions of normality of responses with equal variances. This method determines the number of required simulation replication for each competing design in one step. However, most subsequent IZ methods in the literature have been developed based on the ideas used in Dudewicz and Dalal (1975) and Rinott (1978) which try to improve the efficiency of IZ methods. Chen and Kelton (2000, 2003 and 2005) use differences between sample means in their two-stage methods. Kim and Nelson (2001) developed a fully sequential IZ method in which there is a screening procedure to detect bad designs and ignore them during the procedure before the algorithm terminates. Pichitlamken and Nelson (2001) and Pichitlamken et. al. (2005) introduce Sequential Selection with Memory which guarantees selection of the best with a pre specified probability when some designs have already been simulated. Goldsman et. al. (2000) have extended Ranking and Selection methods such that designs are compared with results obtained in steady-steady simulation. Morrice et. al. (1998 and 1999) have also developed a Ranking and Selection for multi-criteria problems which is a combination of utility theory and IZ procedures.

- **Ordinal Optimization**

This method tries to use comparison of the order of the objective functions of the simulated points in order to find the optimum (Li et. al. 2002). The underlying idea of this method is as follows: Estimating the order of the quality of designs is easier than estimating the values of the objective function values. In most SO cases, the goal is finding the optimum design, the goal is not necessarily finding the value of the objective function of the optimum design. It could be shown that the search for the order converges exponentially while the search for the values converges proportional to the square root of the number of samples used in estimations.

- **Optimal Computing Budget Allocation**

In this method, the number of simulation runs available for the whole optimization (budget) is limited. The goal is finding near best allocation of this budget to designs such that an approximated probability of correct selection is maximized (Chen et. al. 1997 and 1998 and 2000). The main idea is that a bigger share of the budget should be allocated to designs which are more critical in distinguishing the optimum. For the case of multi-criteria problems, see Lee et. al. (2004).

- **Bayesian Methods**

Bayesian methods are used in different steps of a simulation study (Chick 2004). These applications include input modeling, uncertainty analysis, design of experiments for data collection, selection procedures and response surface estimation. One method of SB has been proposed by Chick and Inoue (1999 and 2000) where Bayesian analysis has been used to develop two-step sequential SB methods. A promising area of using Bayesian analysis here is merging screen methods and Ranking and Selection in large scale problems. In a mixed method, some bad designs could be filtered through a screening phase and then, a Ranking and Selection procedure is used for the remaining designs. Alrefaei and Alawneh (2004) have used this idea. In a two-phase method, they first use a subset selection method to detect bad designs. Then they use an IZ method to find the best among remaining pool of designs.

- **Multiple Comparisons Methods**

Multiple Comparisons approaches the screening problem by forming simultaneous confidence intervals on the parameters $\mu_i - \mu_l$ for all $i \neq l$. These $k(k-1)/2$ confidence intervals indicate the magnitude and direction of difference between each pair of alternatives. The most widely used method for forming the intervals is Tukey's procedure, which is implemented in many statistical software packages. General references include Hochberg and Tamhane (1987) and Miller (1981).

2.2.1.3 (Meta)Heuristic Methods

We explain the main ideas of some heuristic methods used in the literature.

- **Simulated Annealing**

The algorithm of Simulated Annealing (SA) proposed by Krikpatrick et. al. (1983) is simple. First, an arbitrary initial solution (current point) is selected (although it is desirable to choose a good solution, but it is not necessary). In each iteration, the algorithm evaluates the objective function of one of the immediate neighboring points of the current point. Two cases may happen: a) The neighbor is a better

point than the current point in terms of the quality of the objective function. In this case the algorithm moves to the neighboring point and this point would be the current point. b) The neighbor is a worse point; then the algorithm moves to the neighbor with a small probability. In fact this probability is the chance of accepting bad points in SA. After moving to the new point or sticking to current point, an iteration of the algorithm terminates and another iteration has to start if no termination condition holds.

The main idea of SA is that the mentioned probability of accepting bad point is initially large; meaning the algorithm is willing to explore more areas of the feasible region by accepting bad moves and escaping local optima. As a search path in SA progresses, the probability is reduced to let the algorithm exploit the information around good points. This optimization idea is rooted in annealing of metals in which as the temperature (equivalent to the probability) is decreased gradually, a more rigid crystal of the metal is formed.

- **Genetic Algorithm**

The fittest survive; this is the rule of nature. Genetic Algorithm has been constructed based on this idea (Holland 1992). It is a set-based search algorithm, where at each iteration it simultaneously generates a number of solutions (Olafsson 2006). In each iteration, a subset of the current set of solutions is selected based on their performance and these solutions are combined into new solutions. The operators used to create the new solutions are survival, where a solution is carried to the next iteration without change, crossover, where the properties of two solutions are combined into one, and mutation, where a solution is modified slightly. The same process is then repeated with the new set of solutions. The crossover and mutation operators depend on the representation of the solution, but not on the evaluation of its performance. They are thus the same even though the performance is estimated using simulation. The selection of solutions, however, does depend on the performance. The general principle is that high performing solutions (which in genetic algorithms are referred to as fit individuals) should have a better chance of both surviving and being allowed to create new solutions through crossover. To see different versions of GA, see Haupt and Haupt (2004) and Goldberg (1989).

- **Tabu Search**

Tabu search was introduced by Glover (1989, 1990) and Glover and Laguna (1997) to solve combinatorial optimization problems and it has been used effectively for simulation optimization. It is a

solution-to-solution method and the main idea is to make certain moves or solutions tabu, that is they cannot be visited as long as they are on what is called the tabu list (Olafsson 2006). The tabu list is dynamic and after each move, the latest solution, or the move that resulted in this solution, is added to the list and the oldest solution or move is removed from the list. Another defining characteristic of tabu search is that the search always selects the best non-tabu solution from the neighborhood, even if it is worse than the current solution. This allows the search to escape local optima, and the tabu list ensures that the search does not revert back. Tabu search has numerous other elements, such as long-term memory that restarts the search with a new tabu list at previously found high quality solutions, and a comprehensive treatment of this methodology can be found in Glover and Laguna (1997). Tabu search has been successfully used in SO problems. For instance, see Dengiz and Alabas (2000).

- **Scatter Search**

Scatter search developed by Glover (1997) operates on a set of solutions, the reference set, by combining these solutions to create new ones. When the main mechanism for combining solutions is such that a new solution is created from the linear combination of two other solutions, the reference set may evolve as illustrated in Figure 2.1. This Figure assumes that the original reference set of solutions consists of the circles labeled A, B and C. After a non-convex combination of reference solutions A and B, solution 1 is created. More precisely, a number of solutions in the line segment defined by A and B are created; however, only solution 1 is introduced in the reference set. In a similar way, convex and non-convex combinations of original and newly created reference solutions create points 2, 3 and 4. The resulting complete reference set shown in Figure 2.1 consists of 7 solutions (or elements).

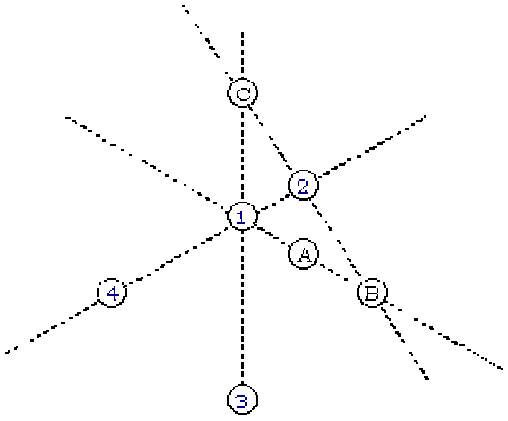


Figure 2.1: Two-dimensional reference set.

A scatter search implementation consists of the following five methods (Glover 1999 and Glover et. al. 2003): 1) A Diversification Generation Method to generate a collection of diverse trial solutions, using an

arbitrary trial solution (or seed solution) as an input. 2) An Improvement Method to transform a trial solution into one or more enhanced trial solutions. 3) A Reference Set Update Method to build and maintain a reference set consisting of the b "best" solutions found (where the value of b is typically small, e.g., no more than 20), organized to provide efficient accessing by other parts of the method. Solutions gain membership to the reference set according to their quality or their diversity. 4) A Subset Generation Method to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions. 5) A Solution Combination Method to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.
Some versions of this algorithm have also been developed for continuous decision variables (Herrera et. al. 2006).

- **Nelder-Mead Simplex**

The Nelder-Mead algorithm (Nelder and Mead 1965) is designed to solve the classical unconstrained optimization problem of minimizing a nonlinear function. This method should not be confused with Simplex method developed by Dantzig (see Taha 1997) for linear programming. The method uses only function values and does not try to form an approximate gradient at any of these points. Hence it belongs to the general class of direct search methods. This method is simplex-based. A simplex S in $\mathbb{R}^d$ is defined as the convex hull of $d+1$ vertices $\theta_0, \theta_1, \dots, \theta_d \in \mathbb{R}^d$. For example, a simplex in $\mathbb{R}^2$ is a triangle, and a simplex in $\mathbb{R}^3$ is a tetrahedron. A simplex-based direct search method begins with a set of $d+1$ points $\theta_0, \theta_1, \dots, \theta_d$ that are considered as the vertices of a working simplex S, and the corresponding set of function values at the vertices $f(\theta_j)$ for $j = 0,1, \dots, d$. The initial working simplex S has to be non-degenerate, i.e., the points $\theta_0, \theta_1, \dots, \theta_d$ must not lie in the same hyperplane.

The method then performs a sequence of transformations of the working simplex S, aimed at decreasing the function values at its vertices. At each step, the transformation is determined by computing one or more test points, together with their function values, and by comparison of these function values with those at the vertices. This process is terminated when the working simplex S becomes sufficiently small in some sense, or when the function values $f(\theta_j)$ are close enough in some sense (provided f is continuous).

The Nelder-Mead algorithm typically requires only one or two function evaluations at each step, while many other direct search methods use d or even more function evaluations. This method could be transformed into a global search method by incorporating stochastic search techniques into it (Luersen,

and Riche 2004). Humphrey and Wilson (1998) introduce a revise Nelder-Mead Method which is believed to be more efficient than original method.

- **Nested Partitions**

Introduced by Shi and Olafsson (2000a), the Nested Partition method (NP) is another metaheuristic for combinatorial optimization that is readily adapted to simulation optimization problems (Shi and Olafsson 2000b). The key idea behind this method lies in systematically partitioning the feasible region into subregions, evaluating the potential of each region, and then focusing the computational effort to the most promising region (Olafsson 2006). This process is carried out iteratively with each partition nested within the last. The computational effectiveness of the NP method relies heavily on the partitioning, which, if carried out in a manner such that good solutions are clustered together, can reach a near optimal solution very quickly.

In the kth iteration of the NP algorithm, a region $r(k) \in \Theta$ is considered most promising. What this means is that the algorithm believes that this is the most likely part of the solution space to contain the optimal solution, and thus the computation effort should be concentrated in this region. As in the beginning nothing is known about the location of the optimal solution, the algorithm is initialized with $r(0) \in \Theta$. The most promising region is then partitioned into M subsets or subregions and the entire surrounding region $\Theta \backslash r(k)$ is aggregated into one. Thus, in each iteration $M + 1$ disjoint subsets that cover the entire feasible region are considered. Each of these $M + 1$ regions is sampled using some random sampling scheme to obtain sets of solutions, and the simulated performance function values at randomly selected points are used to estimate the promising index for each region. This index determines the most promising region in the next iteration by taking the subregion scoring highest on the promising index. If the surrounding region rather than the subregion is found to have the highest promising index, the method backtracks to a previous solution corresponding to a larger region. The new most promising region is partitioned and sampled in a similar fashion. This generates a sequence of set partitions, with each partition nested within the last.

- **Particle Swarm**

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy (1995), inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as GA. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA,

PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.
Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called lbest. When a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest.
The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations.

2.2.1.4 Other Methods

- **Sample Path Optimization**

Sample Path Optimization (SPO) is a method which utilizes the benefits of deterministic optimization techniques in SO (Ferris et. al. 2000). In this method, the SO problem is first transformed into a deterministic optimization problem, then the powerful arsenal of deterministic methods are employed to solve the problem. Consider $\min_{\theta\in\Theta} f(\theta) = \mathrm{E}\,[h(\theta, X)]$ where X is the random vector of the stochastic system of interest. In SPO, samples $x_1, x_2, \dots, x_n$ are taken from the distribution of X. Then the deterministic problem $\min_{\theta\in\Theta} \frac{1}{n}\sum_{j=1}^{n} h(\theta, x_j)$ is solved using deterministic optimization methods. The optimum of the deterministic problem should be an estimate of the optimum. This estimate would be more accurate if n is large. For more discussion, see Andradottir (1998a). This method has also been extended to the problems with stochastic constraints (Gurkan et. al. 1999 and Andradottir et. al.).

- **Response Surface Methodology**

In Response Surface Methodology (RSM), the goal is finding a surrogate for the objective function and optimizing the surrogate (Montgomery and Myers 2002). This surrogate is sometimes called a metamodel which mimics what simulation does. Regression (Cheng 1999) and Artificial Neural Networks (Fausett 1994) are two examples of the widely used metamodels in the literature. Once a metamodel is constructed, deterministic optimization methods can be used to find the optimum. In Sequential RSM methods (Fu 2002), response surface is constructed in multiple stages. For instance, a first order regression model is first fitted on the response surface, then a second order regression model is fitted in a local area of the feasible region which has good quality points based on the output of the fitted first

order model. This process of constructing more accurate metamodels could be continued to reach a (local) optimum.

Some RSMs are automatic in which the user does not update the algorithm of the method in the middle of the search process. However, there are still some RSMs which need the user to adapt the metamodel and the process as the new information of the response surface is obtained. Automatic RSMs are usually more efficient (Neddermeijer et. al. 2000 and Nicolai et. al. 2004). Since it is important in SO to reduce the number of times simulation is run, some versions of RSM have been developed to meet this goal and decrease computational expense of the SO process (Allen and Yu 2000).

2.2.1.5 Mixed Methods

In practice, most SO methods are a combination of ideas of many methods. SO software packages combine different methods in a search algorithm.

For instance, the idea of using metamodel in RSM have been incorporated in various stages of SO or system design (Nasereddin and Mollaghasemi 1999). Laguna and Marti (2002) use a neural network with a hidden layer to screen all solution points and detect bad points before simulation. Wang (2005) combines a neural net with GA. Guikema et. al. (2004) uses a similar technique with regression models. They also use GA as the core optimizer.

Combinations of heuristic methods and other methods have resulted in new methods which are sometimes more efficient. Rosen and Harmonosky (2005), for instance, combine RSM and Simulated Annealing in a two-step method. In the first step in this algorithm, the search process applies a Linear Approximation in a limited search space. The solutions of the first step are usually located in the neighborhood of local optima. Detecting these neighborhoods in the first step, Simulated Annealing tries to find the local optima in the neighborhoods. Olafsson and Gopinath (2000) combine Nested Partitions and Ranking and Selection methods. They used sampling with Ranking and Selection to reduce simulation runs needed for finding the promising region in Nested Partition method. Shi et. al. (1999) uses a similar technique using Optimal Computing Budget Allocation. Combining Simulated Annealing and Stochastic Approximation, Jones and White (2004) claim they developed a method which outperforms OptQuest, a commercial SO software package. Combining different statistical SB methods is another idea used in the literature (Pichitlamken and Nelson 2002).

2.2.2 Single-Objective Qualitative-Variable Methods

In mathematical programming, decision variables are usually quantitative and the objective function is an analytical function of these variables (Azadivar and Tompkins 1999). Available methods in the

literature of SO mostly deal with quantitative decision variables. In fact the underlying assumption of most of the methods is that the decision variables are quantitative. However, real world problems are not that simple. In practice, qualitative variables are widely seen in SO problems. For instance, consider a SO problem of an assembly line in a manufacturing company. The transportation system of the line might be one of the decision variables of interest. This variable may take 1 if conveyor is used, 2 if crane is used or 3 if automated guided vehicle is used. This is a kind of qualitative variable. Many such examples exist in reality. However, a small fraction of research efforts in SO have been focused on these kinds of problems. This is why most practitioners of SO question the capability of literature to address real problems. In addition, what makes SO with qualitative variables difficult is that when a qualitative variable changes its value, a substantial portion of simulation model must be changed (Azadivar and Tompkins 1999, Riche et. al. 2003).

Some methods in the literature can handle qualitative variables. Statistical methods with binary decision variables can potentially be good options for qualitative variables. We believe metamodeling is a potential remedy for these problems. The challenge is that metamodel should be able to effectively learn the response surface of objective functions with qualitative variables. Some researchers have tried to tackle this challenge (Tunali and Batmaz 2003), but more efforts are needed to implement these ideas in practice.

Azadivar and Tompkins (1999) introduce four methods for qualitative-variable problems: Simulated Annealing, Genetic Algorithm, Rule-based Systems and Random Search. In Rule-based Systems, the knowledge and analysis capability of the modeler is combined into a Rule-based System. These systems use “if-then” rules of experts. For instance, if the utility of servers in a queuing system is less than 10%, then reduce the number of servers by one. After obtaining the results of a simulation run, this expert system (Keller 1987) detects the malfunction of the system and propose new configuration of the system to address failures.

2.2.3 Multiobjective Methods

Multiobjective methods are challenged not only by difficulties of single-objective problems, but also the conflict between criteria Azadivar (1999). Many multiobjective-SO methods have roots in deterministic multiobjective routines. Azadivar (1999) classifies these methods into three categories:

1. Optimize one objective, constrain the rest

2. Goal programming: designate a target value for each objective; optimize a linear combination of deviation of each objective from its targeted value (Hillier and Liberman 2001).
3. Desirability/utility function: define utility of each objective, optimize a combination of the utilities (Pasandideh and Niaki 2005).

3. Commercial Softwares of Simulation Optimization

Table 2.1 contains the information about some commercial software packages available in the market (Fu 2001 and 2002).

Optimization Package (Simulation Platform)	Seller	Search Process
AutoStat (AutoMod)	AutoSimulations, Inc. (www.autosim.com)	Evolutionary, Genetic Algorithm
OptQuest (Arena, Crystal Ball, et al.)	Optimization Technologies, Inc. (www.opttek.com)	Scatter Search, Tabu Search, Neural Network
OPTIMIZ (SIMUL8)	Visual Thinking International Ltd. (www.simul8.com)	Neural Networks
SimRunner (ProModel)	PROMODEL Corp. (www.promodel.com)	Evolutionary, Genetic Algorithm
Optimizer (WITNESS)	Lanner Group, Inc. (www.lanner.com/corporate)	Simulated Annealing, Tabu Search

Table 2.1: Simulation Optimization Software Packages

Chapter 3: Hybrid Probabilistic Search

In this chapter, we propose an algorithm that merges Ranking and Selection procedures with a large class of Random Search methods for continuous simulation optimization problems. Under a mild assumption, we prove the convergence of the algorithm in probability to a global optimum. The new algorithm addresses the noise in simulation outputs while benefits the proven efficiency of Random Search methods. In chapter 4, we propose Golden Region search as a new method of simulation optimization, prove that it is a Probabilistic Search method and conclude its convergence.

1. Introduction

Designing a new SO algorithm or selecting an SO method among available routines involve a tradeoff between two conflicting criteria: efficiency vs. convergence (Andradottir 2006). A convergent method can guarantee that the randomness in simulation responses does not mislead the optimization process and global optimum could eventually be reached. This nice property, however, comes at a cost because rigorous mathematical proof of convergence requires periodic simulation of points that have already been simulated (Olafsson 2006) or are located in close proximity of old simulated points. These unwelcome consequences decrease the efficiency of convergent SO methods (Kabirian 2006, Kabirian and Olafsson 2007a and 2007c). Available methods in the literature usually have tendency to one of the two criteria. For instance, many heuristic methods (including metaheuristics) applied to SO problems are among the efficient class that have been implemented in commercial SO software packages (Fu et. al. 2005); in contrast, methods such as Sample Path Optimization and its variants provably have very nice convergence properties (Robinson 1996, Shapiro 1996), though criticized to be inefficient (Deng and Ferris 2007, Azadivar 1999).

In this chapter, I bridge the gap between practical appeal of a large class of heuristic methods called Probabilistic Search (PS) techniques and theoretical convergence guarantee by proposing a methodology to link these methods with Ranking and Selection (R&S). The remainder of the chapter is organized as

follows. We formally define PS algorithms in section 2. In section 3, R&S methods are briefly introduced and a methodology for merging R&S with PS algorithms is presented. Section 4 contains numerical results.

2. Probabilistic Search Methods

Consider the deterministic continuous optimization problem below:

$$\min_{\theta \in \Theta} f(\theta) \tag{3.1}$$

where $f: \Theta \to \mathbb{R}$ and $\Theta \subset \mathbb{R}^n$ is the compact set of feasible points. Let $\theta^* \in \Theta$ be one of the possibly many global optimums of (3.1).

Random Search methods are a class of iterative optimization algorithms in whose k th iteration, a finite number of points $\theta_k^{(1)}, \theta_k^{(2)}, \dots, \theta_k^{(H_k)} \in \Theta$ are selected via a specific sampling strategy Ψ_k and evaluated (see Andradottir 2006 for SO version).

Algorithm 3.1: Random Search Methods

Step 0: (Initialize). Choose the initial sampling strategy Ψ_1 *and let* $k = 1$.

Step 1: (Sample). Select points $\theta_k^{(1)}, \theta_k^{(2)}, \dots, \theta_k^{(H_k)} \in \Theta$ *according to sampling strategy* Ψ_k

Step 2: (Evaluate). Compute $f\left(\theta_k^{(j)}\right)$ *for* $j = 1,2, \dots, H_k$.

Step 3: (Update). Considering the quality of the evaluated points thus far, pick and introduce current optimum of the search process. If termination condition(s) of the algorithm hold(s), stop the algorithm, otherwise choose an updated strategy Ψ_{k+1} *, let* $k = k + 1$ *and go to step 1.* ■

This class is broad enough to include many heuristic and metaheuristic methods. We narrow the definition a bit to what we call Probabilistic Search (PS) methods as follows (Kabirian 2009):

Definition 3.1: *Let* W *be a Lebesgue measure. The Probabilistic Search methods are a subclass of Random Search methods for problem (1) such that for any arbitrary subset* $G \subseteq \Theta$ *with* $W(G) > 0$*, the followings hold:*

1) $Pr\left\{\bigcup_{j=1}^{H_k}\left(\theta_k^{(j)} \in G\right)\right\} > 0 \quad \forall\, k = 1,2, \dots$

2) $lim_{k\to\infty} Pr\left\{\cup_{j=1}^{H_k}\left(\theta_k^{(j)} \in G\right)\right\} \neq 0.$ ■

Many Random Search based methods satisfy the conditions of Definition 3.1. Others can also be a PS method if, in addition to their core optimization process, a point is also selected randomly from Θ and evaluated in each iteration.

Now, we prove that every PS algorithm almost surely visits a point with objective function within an arbitrary neighborhood of a global optimum if a regularity condition is met.

Definition 3.2: *Call the interval* $O_\varepsilon = [f(\theta^*)\,,\ f(\theta^*) + \varepsilon]$ *where* $\varepsilon > 0$*, the* ε*-optimum interval.* ■

Definition 3.3: *Let* $B(\theta; r) = \left\{y \in \mathbb{R}^n \middle| |\theta - y| \leq r\right\}$ *be a ball of radius* $r > 0$ *centered at the point* θ*.* ■

Definition 3.4: *The discrete random variable* $X \sim NHG(\{p_k\}_{k=1}^{\infty})$ *has Non-Homogenous Geometric (NHG) distribution with success probability* p_k *in* k*-th trial if*

$$\Pr(X = x) = \begin{cases} p_x \prod_{k=1}^{x-1}(1 - p_k) & if\ x = 1,2,\dots \\ 0 & o.w. \end{cases}.\blacksquare \tag{3.2}$$

Assumption 3.1: *A ball* $\mathrm{B}(\theta^*; r)$ *exists such that* $W(\mathrm{B}(\theta^*; r) \cap \Theta) > 0$ *and* $f(\theta)$ *is continuous for all* $\theta \in \mathrm{B}(\theta^*; r) \cap \Theta$*.* ■

Lemma 3.1: *For problem (1) under Assumption 3.1, there exists another ball* $B(\theta^*; r_\varepsilon)$ *for any given* $\varepsilon > 0$ *such that:*

1) $B(\theta^*; r_\varepsilon) \subseteq B(\theta^*; r)$

2) $f(\theta) \in O_\varepsilon$ *for* $\theta \in B(\theta^*; r_\varepsilon) \cap \Theta$

3) $W(B(\theta^*; r_\varepsilon) \cap \Theta) > 0$

Proof: The proof is a direct result of the continuity of f on $\mathrm{B}(\theta^*; r)$. ■

Lemma 3.2: *Let* $X \sim NHG\ (\{p_k\}_{k=1}^{\infty})$ *where* $p_k > 0\ \ \forall k = 1,2,\dots$ *and* $lim_{k\to\infty} p_k \neq 0$*. Then* $Pr(X < \infty) = 1.$

Proof: Using Equation 2, the proof is straightforward. ■

Theorem 3.1: *Let* X_ε *be the number of iterations required by a PS method to visit a point* θ *such that* $f(\theta) \in O_\varepsilon$ *for a given* $\varepsilon > 0$*. Under Assumption 3.1,* $Pr(X_\varepsilon < \infty) = 1.$

Proof: By Lemma 3.1, a $\mathrm{B}(\theta^*; r_\varepsilon)$ where $W(\mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta) > 0$ exists for given $\varepsilon > 0$. Then, observe that $X_\varepsilon \sim \mathrm{NHG}(\{p_k\}_{k=1}^{\infty})$ where $p_k \geq \Pr\left\{\cup_{j=1}^{H_k}\left(\theta_k^{(j)} \in \mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta\right)\right\} > 0$ for $k = 1,2,\dots$. Since

$W(\mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta) > 0$, Definition 3.1 implies $p_k > 0$ for $k = 1,2, \ldots$ and $\lim_{k\to\infty} p_k \neq 0$. Hence, $\Pr(X_\varepsilon < \infty) = 1$ following Lemma 3.2. ■

Now let us consider the SO version of problem (3.1):

$$\min_{\theta\in\Theta}\{f(\theta) = \mathrm{E}(L(\theta))\} \tag{3.3}$$

where L is a function of decision variables and random variables of a stochastic system and it is a consistent estimator of f, also $\mathrm{E}(.)$ is the mathematical expectation operator. Indeed, we assume the closed form of function $f(\theta)$ is not available and can only be numerically estimated by $L(\theta)$ through averaging a number of sample performance functions obtained via simulating design point θ.

If estimated values of the objective function are used, PS methods can be applied to SO problems too. However, in order to prove the convergence properties of the PS methods, the number of simulation replications for each solution point must be controlled. The real challenge is that although a PS method can visit a point in any ε-optimum interval by Theorem 3.1, it may not be able to actually introduce (select) the visited point or a better one as the optimum because of the random noise in objective function values. A remedy that could be used for discrete and bounded decision variables is to guarantee (in probability or almost surely) that each single feasible point is simulated with infinite simulation runs as the number of iterations of the search algorithm goes to infinity. Then following the laws of large numbers, asymptotic convergence to a global optimum is easy to establish. However, we need to deal with infinite number of points in continuous-variable problems and the remedy for discrete settings may not work by its own. In the subsequent sections, we propose merging PS methods with specific R&S procedures. This way, as we will show later, the asymptotic convergence is guaranteed for continuous cases.

3. Analysis of Hybrid Probabilistic Search Methods

A number of researchers have proposed merging Statistical Methods with other SO methods. For instance, Olafsson and Gopinath (2000) and Olafsson (2004) combine Nested Partitions and Ranking and Selection methods. They used sampling with Ranking and Selection to reduce simulation runs needed for finding the promising region in Nested Partition method. Shi et. al. (1999) uses a similar technique using Optimal Computing Budget Allocation.

In this section, we propose merging PS methods with R&S methods. Then the convergence properties of the integrated method are studied.

3.1. Algorithm of Hybrid Probabilistic Search

Since simulation responses are noisy, the number of replications (samples) in a simulation run for each design point proposed by a PS can significantly affect two conflicting goals: convergence properties and efficiency of the algorithm. Efficiency recommends less replications in a simulation run, but this means more noise in the estimated values of objective function, more chance of wrongly announcing the best solution and consequently poor convergence. We propose a statistical procedure to assess the number of replications of each design point that help the algorithm guarantee asymptotic convergence. Whenever the PS sends a new point to simulation module for the first time, only a constant number of replications denoted by η_R are initially used for estimation of the objective function; further replications are subject to necessity. We use an R&S method (specifically Indifference Zone (IZ) technique) after a fixed number of iterations in order to determine the best point among already simulated points.

The statistical procedure merges with PS core optimizer as the algorithm below outlines:

Algorithm 3.2: Hybrid Probabilistic Search

Step 0: Define two sequences called error rate denoted by $\{\alpha_h\}_{h=1}^{\infty}$ *and IZ (parameter) denoted by* $\{\gamma_h\}_{h=1}^{\infty}$ *where* $\lim_{h\to\infty}\alpha_h = \lim_{h\to\infty}\gamma_h = 0$ *,* $0 < \alpha_{h+1} \le \alpha_h \le 1$ *and* $0 < \gamma_{h+1} \le \gamma_h$*. Define the number of replications between the R&S implementations and denote it by* τ *.*

Step 1: Choose the initial sampling strategy Ψ_1 *and let algorithm iteration counter* $k = 1$ *and R&S implementation counter* $h = 1$*.*

Step 2: Denote the introduced optimum of the algorithm after iteration k *by* $\hat{\theta}_k^*$*. Set* $\hat{\theta}_1^* = \{\}$*.*

Step 3: Select new points $\theta_k^{(1)}, \theta_k^{(2)}, \dots, \theta_k^{(H_k)} \in \Theta$ *according to sampling strategy* Ψ_k *and let* $\mathrm{Z}_k = \left\{\theta_{k'}^{(j)} \; for \; k' = 1,2,\dots,k \; and \; j = 1,2,\dots,H_{k'}\right\}$*.*

Step 4: For $j = 1,2,\dots,H_k$ *simulate* $\theta_k^{(j)}$ *with* η *replications.*

Step 5: Let $R_k\left(\theta_{k'}^{(j)}\right)$ *be the number of simulation replications done for* $\theta_{k'}^{(j)}$ *by the end of iteration* k. *Set replication counter* $R_k\left(\theta_k^{(j)}\right) = \eta$ *for* $j = 1,2,\dots,H_k$. *Set* $R_k\left(\theta_{k'}^{(j)}\right) = R_{k-1}\left(\theta_{k'}^{(j)}\right)$ *for* $k' = 1,\dots,k-1$ *and* $j = 1,\dots,H_{k'}$.

Step 6: Denote by $L_r\left(\theta_k^{(j)}\right)$, *the objective function of* $\theta_k^{(j)}$ *estimated in* r*-th simulation replication. Set sample mean* $L\left(\theta_k^{(j)}\right) = \sum_{r=1}^{R_k\left(\theta_k^{(j)}\right)} \frac{L_r\left(\theta_k^{(j)}\right)}{R_k\left(\theta_k^{(j)}\right)}$ *for* $j = 1,2,\dots,H_k$.

Step 7: If $k \neq \tau h$, *then let* $\hat{\theta}_k^* \in \arg\min_{\theta \in \mathrm{Z}_k} L(\theta)$ *and go to step 8; otherwise do:*

7.1. Design an R&S method and apply it to Z_k *such that a difference of* γ_h *or more in the mean objective functions of the best point in* Z_k *and all other points in* Z_k *is detected with probability* $1-\alpha_h$ *or more. For* $k' = 1,2,\dots,k$ *and* $j = 1,2,\dots,H_{k'}$, *let* $\lambda_{hjk'}$ *denote the total number of new simulation replications consumed in the current R&S implementation for* $\theta_{k'}^{(j)} \in \mathrm{Z}_k$ *on top of* $R_k\left(\theta_{k'}^{(j)}\right)$ *old simulation replications available for this point and let* $R_k\left(\theta_{k'}^{(j)}\right) = R_k\left(\theta_{k'}^{(j)}\right) + \lambda_{hjk'}$.

7.2. Set $L\left(\theta_{k'}^{(j)}\right) = \sum_{r=1}^{R_k\left(\theta_{k'}^{(j)}\right)} \frac{L_r\left(\theta_{k'}^{(j)}\right)}{R_k\left(\theta_{k'}^{(j)}\right)}$ *for all* $\theta_{k'}^{(j)} \in \mathrm{Z}_k$. *Then let* $\hat{\theta}_k^* \in \arg\min_{\theta \in \mathrm{Z}_k} L(\theta)$.

7.3. Set $h = h+1$ *and go to step 8.*

Step 8: If a termination condition holds, introduce $\hat{\theta}_k^*$ *as the optimum and exit the algorithm. Otherwise choose an updated strategy* Ψ_{k+1} , *let* $k = k+1$ *and go to step 3.* ■

Generally, any IZ procedure that uses old simulation replications along with new replications and guarantees selection of the best with a given probability when the true objective function of the best and the rest of the designs are distanced at least by an IZ parameter could be used in Hybrid PS algorithm. Boesel et. al. (2003) proposes such procedures. Bayesian methods are other alternatives in which a posterior probability of correct selection is guaranteed (Chick and Inoue 2001a and 2001b, 1999, 2000).

However, as far as we are aware, all statistical selection of the best procedures in the literature (including in Boesel et. al. and Chick and Inoue 2001a and 2001b) assume that simulation outputs are normally distributed. Nelson et. al. (2001) and Nelson and Goldsman (2001) study the robustness of

normality assumption and conclude that probability of correct selection could approximately be retained with mild departures from normality. The main justification for normal assumption in many simulation studies is that interesting simulation outputs are usually averages of a large number of observations; hence, central limit theorem suggests normality holds asymptotically. In addition, we need to assume that the second moment of the objective function for all feasible values of decision variable is finite.

3.2. Convergence of Hybrid Probabilistic Search

Theorem 3.2: *If a hybrid PS method is applied to SO problem of Equation 1 under Assumption 3.1, the sequence* $\{f(\hat{\theta}_k^*)\}_{k=1}^{\infty}$ *converges in probability to* $f(\theta^*)$*, that is* $lim_{k\to\infty} Pr\{|f(\hat{\theta}_k^*) - f(\theta^*)| < \varepsilon\} = 1$ *for all* $\varepsilon > 0$.

Proof: Following Theorem 3.1, the number of iterations before the algorithm visits a point with mean objective function in O_ε would be finite with probability 1 (w.p.1); say for iterations $k \geq k'$ where $\Pr(k' < \infty) = 1$, a point θ' has already been simulated such that $f(\theta') \in O_\varepsilon$. Since the IZ sequence $\{\gamma_h\}_{h=1}^{\infty}$ converges to zero, the number of iterations before the algorithm visits a point θ' with mean objective function in O_ε and IZ parameter becomes less than $f(\theta^*) + \varepsilon - f(\theta')$ is finite w.p.1; say these two events have happened for any iteration $k \geq k''$ where $\Pr(k'' < \infty) = 1$. Let $h' = \left\lceil \frac{k''}{\tau} \right\rceil$ where $\lceil . \rceil$ is round up function. Then for $h \geq h'$, the R&S procedure implementations guarantee introduction of θ' or another point with objective function in O_ε as the optimum with probability $1 - \alpha_{h'}$ or more because at least there exist θ' in the set of simulated points such that it is at least $f(\theta^*) + \varepsilon - f(\theta') > \gamma_{h'}$ far away from all the other simulated points outside O_ε . Hence: $\lim_{k\to\infty} \Pr\{|f(\hat{\theta}_k^*) - f(\theta^*)| < \varepsilon\} \geq \Pr(k'' < \infty) \lim_{h\to\infty}(1 - \alpha_h) = 1$ and consequently $\lim_{k\to\infty} \Pr\{|f(\hat{\theta}_k^*) - f(\theta^*)| < \varepsilon\} = 1$. ■

4. Example Procedure

In this section, we showcase the effectiveness of the Hybrid PS methods. Of course, any PS method could potentially be used here; but we are more interested to see how well the wedding between R&S methods and PS procedures work. Therefore, we select the simplest possible PS method which is called Naïve Random Search (NRS). The sampling strategy of NRS picks one point ($H_k = 1$ for $k = 1,2, \ldots$) uniformly randomly from the compact feasible region. The method is called "Naïve" because it ignores

the information of past searches in future sampling strategies. When applied to SO problems, NRS introduces the point with lowest estimated objective function as the current optimum.

Algorithm 3.3: Naïve Random Search

Step 1: Let $k = 1$.

Step 2: Select one point from the feasible region uniformly randomly. Denote this point by $\theta_k^{(1)}$ *and let* $\mathrm{Z}_k = \left\{\theta_{k'}^{(1)} \; for \; k' = 1,2, \dots, k\right\}$.

Step 3: Simulate $\theta_k^{(1)}$ *with* η *replications and let* $L\left(\theta_k^{(1)}\right) = \sum_{r=1}^{\eta} \frac{L_r\left(\theta_k^{(1)}\right)}{\eta}$ *be the estimated objective function.*

Step 4: Let $\hat{\theta}_k^* \in \arg\min_{\theta \in \mathrm{Z}_k} L(\theta)$.

Step 5: If termination condition(s) of the algorithm hold(s), introduce $\hat{\theta}_k^*$ *as the optimum and stop the algorithm; otherwise let* $k = k + 1$ *and go to step 2.* ■

We are interested in comparing the performance of NRS with the so called Hybrid NRS defined below.

Definition 3.5: *Hybrid NRS is a kind of Hybrid PS method in which the sampling strategy of each iteration selects one point uniformly randomly from the feasible region.* ■

In our experiments, we use the IZ procedure of Boesel et. al. (2003) as the R&S procedure required for the Hybrid NRS. Also, we set $\eta = 2$, $\tau = 100$, $\alpha_1 = 0.50$, $\gamma_1 = 0.10$ and for $h = 1,2, \dots$ we use $\alpha_{h+1} = 0.9\alpha_h$ and $\gamma_{h+1} = 0.9\gamma_h$. To accelerate the experiments, we replace simulation with a noisy objective function. Specifically, we use a closed form 2-dimensional objective function with a unique global optimum. Whenever the simulation output is required for a point, we generate a zero-mean normal random variable with variance 10 and add it to the objective function value computed via the closed-form formula. Figure 3.1 shows the objective function of the problem we used in our experiment. The decision variables are both bounded between 0 and 10. The global optimum of the problem is (5,5) with the objective function 1. We terminated the optimization process of both algorithms when a budget of 1000 simulation replications was spent. Both methods were run 10000 times in order to get robust results. For each algorithm, we computed the average of the expected value of the objective function of the introduced optimum after each simulation replication. Figure 3.2 shows logarithmic values of these averages as the optimization process progresses.

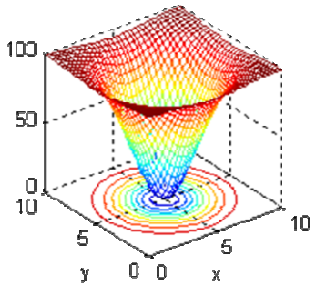


Figure 3.1: The objective function of the test problem of HPS

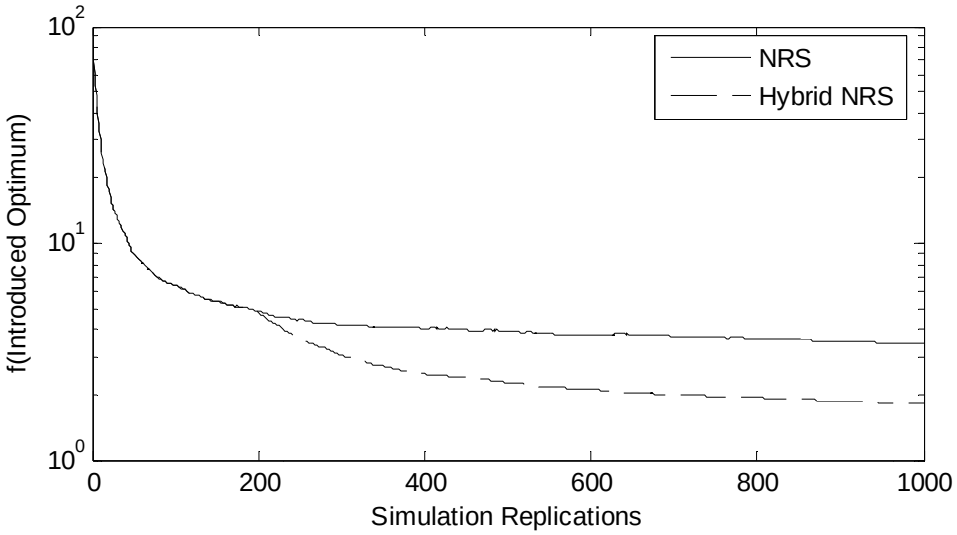


Figure 3.2: The performance curves of Naïve Random Search and Hybrid Random Search methods

Figure 3.2 suggests that when R&S procedure is applied for the first time in Hybrid NRS, convergence is accelerated. As we expected, the disadvantage of solely using NRS is that it may simulate a bad point which turn out to have a very good estimated objective function such that even near optimum points no longer could outperform the misleading estimated good quality of the bad point. In fact, R&S helps clean up the quality of the points NRS simulates.

Chapter 4: Golden Region Search Method

Motivated by the ideas explained in chapter 1 for improving the efficiency of SO methods, a new SO approach called Golden Region search is developed for continuous problems in this chapter. Golden Region divides the feasible region into a number of (sub) regions and selects one region in each iteration for further search based on the quality and distribution of simulated points in the feasible region and the result of scanning the response surface through a metamodel. The experiments show the Golden Region method is efficient compared to three well-established approaches in the literature. We prove that Golden Region satisfies the conditions of Probabilistic Search methods and hence convergence in probability to a global optimum.

1. Introduction

The problem of interest in this Chapter is still minimization of a stochastic and expensive objective function defined in Equation 3.3 which is the SO problem $\min_{\theta\in\Theta}\{f(\theta) = \mathrm{E}(L(\theta))\}$.

Our aim in this chapter is to propose a method which is both efficient and provably convergent. The chapter is organized as follows. In section 2, notations of the methodology are introduced. In section 3, an overview of the new algorithm is presented. In this section, I intend to provide the reader with a big picture of what happens in the algorithm. However, algorithm with more details is presented in section 4. Some implementation issues are discussed in section 5. Theory of the method and its convergence analysis is presented in section 6. In chapter 5, we will test the method and provide numerical results.

2. Notation

This chapter introduces a new PS method to address SO problem (3.3). The parameters and variables of the method are collectively introduced in tables 4.1 and 4.2 respectively.

Parameter	Name

Θ_i	i th (sub)region of the feasible region
υ	Size of initial population
U^*	Maximum number of simulated points in the saturation neighborhood of the best found point
δ_L	Lower bound of the criterion coefficients
δ_U	Upper bound of the criterion coefficients
δ, δ'	Constant terms in criterion coefficient change
$\mathrm{N}_{ii'}$	Proximity measure of regions i and i'
η	Number of replication for the simulation of new points
τ	Number of consecutive iterations until R&S is applied
α_h	h-th Probability of wrong selection in the R&S procedure
γ_h	h-th Indifference zone
χ	Number of regions
β_i	Number of metamodel samples in i-th region
β	Number of samples selected when Space Score dominates
ω	Covariance matrix of multivariate normal distribution in quality criterion
ρ_L	Lower bound of standardized objective function approximated by metamodel
ρ_U	Upper bound of standardized objective function approximated by metamodel
σ	Saturation scaling vector
ξ	Saturation radius
ν	Mean Squared Error in metamodel training

Table 4.1: Parameters of the proposed algorithm

Other Notations	Name
n	Number of decision variables
θ	Vector of decision variables
Θ	Feasible region
f	Objective function
θ^*	Optimum Decision Variables
k	GR iteration index
i	Region index
j	Index of simulated design point
r	Simulation Replication Index
J_k	Set of indices of simulated points up to iteration k
$f(\theta)$	Objective function of θ
θ_j	j-th simulated point (if $j > \upsilon$, this point is simulated in iteration $(j - \upsilon)$ of the algorithm)
$\bar{L}_k(\theta)$	Estimator of objective function of θ at the beginning of iteration k
$L_r(\theta)$	Objective function obtained in r-th simulation replication of θ
$R_k(\theta)$	Number of simulation replications performed for θ up to the beginning of iteration k
A_{ijk}	Metamodel-based approximation of the objective function of the j-th sample of i-th region in iteration k
A'_{ijk}	Standardized value of A_{ijk}
M_{ik}	Metamodel score of i-th region in iteration k

m_k	Coefficient of metamodel criterion at the beginning of iteration k
S_{ik}	Space Score of i-th region in iteration k
s_k	Coefficient of space criterion at the beginning of iteration k
Q_{ik}	Quality Score of i-th region in iteration k
q_k	Coefficient of quality criterion at the beginning of iteration k
U_{jk}	Saturation indicator of j-th point in iteration k
U_{jk}^*	Upper bound for U_{jk}
u_{jk}	Number of simulated point in the saturation neighborhood of j-th point at the beginning of iteration k
ϕ_{ik}	Total score of i th region in iteration k
π_{ik}	Probability of selecting i-th region in iteration k
T_{ik}	Tally of the number of simulated points inside region i at the beginning of iteration k
Y_{ik}	Density of the simulated points inside and surrounding the region i in iteration k
f'_{jk}	Standard objective function of j-th simulated point at the beginning of iteration k

Table 4.2: Other Notations of GR

3. Overview of the Golden Region Algorithm

The Golden Region (GR) method is an iterative search-based optimization process (see Figure 4.1). Candidate solutions are simulated in a black box simulation module that returns estimations of objective functions to the optimizer. GR utilizes a meta-model learner that mimics the role of simulation. The benefit of training the metamodel is that the optimizer can utilize trained metamodel to cheaply evaluate the goodness of sampled points of feasible region before actually running expensive simulation.

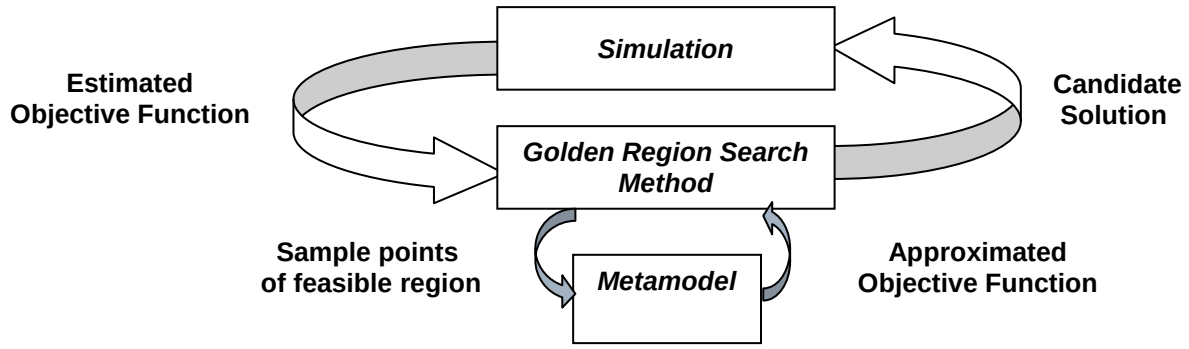


Figure 4.1: The framework of the method

The feasible region of the problem in the GR method is divided into a number of user defined sub regions which we simply call them regions in this chapter. The algorithm starts out with a (random)

population of feasible points. Simulating the design points of this initial population, the algorithm can proceed to an iterative search. In each iteration, one of the regions is selected, called promising region, then a new point in the promising region is chosen and simulated. Simulating this point, the iteration terminates and the algorithm continues until one of the terminating conditions holds. The key questions to be addressed here and in the next section are how the promising region and the new point inside it are selected.

In each iteration of the algorithm, a probability is assigned to each region, and then the promising region is selected randomly using these probabilities. We propose three criteria that affect the probability of selecting a particular region. These are called space score, meta-model score, and quality score. These three criteria are combined into a single indicator called total score for the region. The higher the total score, the more promising the region would be in terms of hiding the global optimum and a higher probability of selection for further search is assigned to the region.

The space score of a region represents how much the region has been visited. By assigning higher space score to the regions that are less visited, that is, have fewer number of already simulated points inside the region and other regions in close proximity of the region, the algorithm seeks to explore unvisited areas of the feasible region.

Motivated by the observation that running simulations is computationally expensive relative to other activities, GR method induces a global metamodel to the input-output of simulation. The metamodel score of a region is calculated using the metamodel based prediction of the objective functions of a number of sampled points inside the region. The better the predicted objective function of the samples of a region, the higher this criterion would be. In fact, this criterion monitors the quality of the whole feasible region.

Finally, the quality score for a region represents the quality of the estimated objective function of simulated points inside the region. The better the quality of these points, the higher the quality score. This criterion exploits the information of high quality simulated points with the hope of discovering local (or maybe global) optima near these points.

To construct a total score, we use a criterion coefficient for each of the three criteria, which are changed between iterations based on the effectiveness of using every criterion.

After selecting the promising region, a point is selected inside the promising region and simulated. Selection of this point is based on the most effective criteria of the promising region in the total score of the region, called dominating criterion, and consequently the probability of selecting the region. In fact, the algorithm needs to know why the promising region has been selected by determining the criterion which contributes most. If the dominating criterion is quality, it means the promising region has some high quality points that might be located in vicinity of a local optimum, so the new point is selected near the high quality point. If metamodel dominates, this is a signal that there are some promisingly good points in the promising region based on the global monitoring metamodel, so a promising point is simulated. Finally, in the case of space-criterion domination, the algorithm hopes to simulate a good quality random point in the unvisited area of the promising region. However, the algorithm is designed such that the new simulation point is not too close to already simulated points.

4. Detailed Golden Region Algorithm

The feasible region of the problem Θ is divided into χ regions by the user in GR method. The space of i-th region is denoted by Θ_i. Section 5.1 provides some guidelines of partitioning.

In the first step, a number of points, say υ points, are selected from the feasible region and simulated with a fixed number of replications. The initial population could be selected randomly or from certain regions of the feasible region to have a specific level of diversity in the population. After simulating initial population, the algorithm in each iteration selects one region called promising region among all regions, simulates a point inside the promising region and goes to next iteration if none of the terminating conditions hold (e.g. maximum possible iterations, maximum stalled searches, …). In the next sections, we discuss how the promising region and the point inside it are selected in a typical iteration k.

4.1. Promising Region Selection

As noted above, for selecting a new promising region, we use three criteria called space score, quality score and metamodel score. The algorithm assigns a value between zero and one for each criterion to each region based on some indicators and procedures that will be elaborated upon in the next sections. Denote by S_{ik}, Q_{ik} and M_{ik}, the values of space score, quality score and metamodel score assigned to i-th region respectively. We define a total score, denoted by ϕ_{ik} for region i as follows:

$$\phi_{ik} = \max\{s_k S_{ik}\ , q_k Q_{ik}\ , m_k M_{ik}\} \quad \forall i \tag{4.1}$$

where s_k, q_k and m_k are space, quality and metamodel criterion coefficients which take positive values and add up to one. In the first iteration, these coefficients are set to 1/3, but their values are changed at the beginning of the subsequent iterations based on the success or failure of the dominating criterion of the previous iteration in finding a good design point (see section 4.3 for details).

A probability of selection, denoted by π_{ik} for i-th region, is calculated for each region by normalizing the total score:

$$\pi_{ik} = \frac{\phi_{ik}}{\sum_{i'=1}^{\chi} \phi_{i'k}} \quad \forall i \ . \tag{4.2}$$

Using a uniform random number between zero and one, a promising region for current iteration is then selected with respect to this probability mass function.

4.1.1 Space Score

In addition the feasible region partitioning, the algorithm requires a proximity measure for any two regions including the proximity of a region to itself in order to calculate the space score. This measure represents how similar the design points of one region are to those of the other region. We let $\mathrm{N}_{ii'}$ denote the positive-valued proximity measure of each two regions where $i, i' = 1,2, \dots, \chi$. As an example of this measure, one may define a center point for each region (a vector of decision variables as the representative of the whole region), and relate the proximity measure of two regions to the reciprocal of the Euclidian distance between the center points of the regions for $i \neq i'$ and choose the proximity function of each region with itself such that $\mathrm{N}_{ii} > \max_{i' \neq i} \mathrm{N}_{ii'}$.

We let T_{ik} denote the tally of the number of simulated points inside region i. We define an indicator, Y_{ik} called the visit indicator which measures the "density" of the simulated points inside and surrounding the region by:

$$Y_{ik} = \frac{\sum_{i'=1}^{\chi} T_{i'k} \mathrm{N}_{\mathrm{ii}'}}{\sum_{i'=1}^{\chi} \mathrm{N}_{\mathrm{ii}'}} \quad \forall i \ .$$

Standardizing the visit indicator, we get the space score:

$$S_{ik} = \frac{\max_{i'} Y_{i'k} - Y_{ik}}{\max_{i'} Y_{i'k} - \min_{i'} Y_{i'k}} \quad \forall i \ .$$

4.1.2 Quality Score

Bayraksan and Morton (2006) provide procedures for constructing confidence intervals on optimality gap of a given good quality point. Here we use a simple procedure as follows to determine the quality of a region without spending more simulation efforts.

For $k \geq 0$, let $\mathrm{J}_k = \{1,2, \dots, v + k\}$ be the index set of simulated points up to iteration k. The standardized objective function of simulated point θ_j at the beginning of iteration k denoted by $f'_{jk}$is defined as:

$$f'_{jk} = \frac{\bar{L}_k(\theta_j) - \min_{j'} \bar{L}_k(\theta_{j'})}{\max_{j'} \bar{L}_k(\theta_{j'}) - \min_{j'} \bar{L}_k(\theta_{j'})} \quad \forall j \in \mathrm{J}_{k-1} .$$

where $\bar{L}_k(\theta_j) = \frac{\sum_{r=1}^{R_k(\theta_j)} L_r(\theta_j)}{R_k(\theta_j)}$ is the sample mean. The standard objective function values are updated in iterations that best or worst found points change or estimates of already simulation points are renewed.

We need the following definition for this score.

Definition 4.1: *The saturation neighborhood of a point $y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$ denoted by $\breve{E}_{\sigma,\xi}(y)$ is the region inside an ellipsoid centered at θ with scale parameter vector $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathbb{R}^n$ and radius $\xi \in \mathbb{R} \geq 0$, that is*

$$\breve{\mathrm{E}}_{\sigma,\xi}(y) = \left\{ x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n \;\middle|\; \sum_{l=1}^{n} \left(\frac{x_l - y_l}{\sigma_l}\right)^2 \leq \xi^2 \right\}. \blacksquare$$

If there is at least one simulated point in region i, the Quality Score is:

$$Q_{ik} = \max_{\{\theta_j \in \Theta_i \mid j \in \mathrm{J}_{k-1}\}} U_{jk}(1 - f'_{jk}) \quad \forall i \tag{4.3}$$

Otherwise it is set to zero.

In the above formula, U_{jk} that we call the saturation indicator of simulated point θ_j plays an important role.

The key idea of the saturation indicator is that searches near already simulated points should be limited. If such searches were always allowed, the algorithm might excessively search surrounding local optima. The better the quality of the objective function of a point, the higher the maximum possible searches near this point, and consequently more searches are allowed in associated area. We call already

simulated point θ_j "saturated" (and set $U_{jk} = 0$) if the number of simulated designs in the saturation neighborhood $\breve{\mathrm{E}}_{\sigma,\xi}(\theta_j)$ exceeds its maximum possible searches, otherwise the point is "unsaturated" (and $U_{jk} = 1$) and is allowed to contribute to the value of the quality score for the encompassing region (The parameters σ and ξ are defined by user). Mathematically, if u_{jk} represents the number of already simulated points in the saturation neighborhood of θ_j, the saturation indicator is then assigned by:

$$U_{jk} = \begin{cases} 1 & u_{jk} \le U_{jk}^* \\ 0 & \text{o.w.} \end{cases} \quad \forall j \in \mathrm{J}_{k-1}$$

where U_j^* is the upper bound of allowed searches in the saturation neighborhood of j-th simulated point. For instance, the graph of Figure 4.2 shows how a linear interpolation could be used to find U_{jk}^* for θ_j, that is $U_{jk}^* = 1 + (U^* - 1)\frac{\left(\max_{j'} \bar{L}_k(\theta_{j'}) - \bar{L}_k(\theta_j)\right)}{\max_{j'} \bar{L}_k(\theta_{j'}) - \min_{j'} \bar{L}_k(\theta_{j'})}$. Here, the number of possible searches for the best and worst found points are respectively set to a user defined parameter U^* and one.

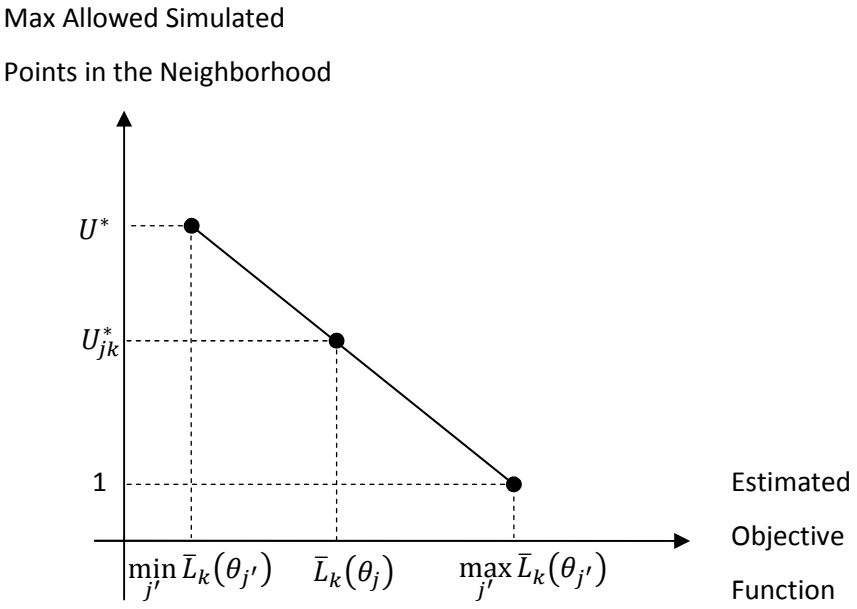


Figure 4.2: The graph of maximum possible searches in the saturation neighborhood of simulated points

4.1.3 Metamodel Score

While the quality criterion acts locally, the metamodel criterion models the response surface globally and monitors the whole feasible region.

Before the first iteration, a population of (random) sample points should be selected from the space of each region. Whenever the metamodel is updated, such as the first iteration when the metamodel is initially constructed, it predicts the objective function of sampled points from each region. Let β_i be the number of metamodel samples in i-th region which is a user-defined parameter. Let A_{ijk} be the objective function of the j-th sample of i-th region approximated by metamodel. Let $A'_{ijk} = \frac{A_{ijk}-\rho_L}{\rho_U-\rho_L}$ be the standardized value of A_{ijk} where ρ_L and ρ_U are user-defined parameters. The standardization makes sure A'_{ijk} values are positive and bounded, say $0 < \rho_L \leq A'_{ijk} \leq \rho_U < \infty$. The metamodel score is defined via

$$M_{ik} = \frac{\min_{i'}\{\min_j A'_{i'jk}\}}{\min_j A'_{ijk}} \quad \forall i \ . \tag{4.4}$$

4.2. New Point Generation

After selecting a new promising region, a point inside this region must be simulated. Recall from Equation 4.1 that the total score equals the largest product of each of the three scores and its coefficient. The largest value defines the dominating criterion and determines the generation of the next point.

If the dominating criterion is space, first β' sample points are selected randomly from the space of the promising region. Then, the sample point whose minimum distance with other already simulated points is maximum, is selected and simulated. This strategy tries to improve the significance of difference between the new simulated design point and other simulated designs.

If the dominating criterion is metamodel, the algorithm chooses the point inside the promising region that has had the best approximated objective function based on metamodel. Mathematically, if the code of the promising region is i , the sample point $j \in \arg\min_{j'} A'_{ij'k}$ of the promising region is simulated. Once the new point is simulated, a new sample point is drawn from the space of the promising region and is replaced with the new simulated point in the population of sample points of the promising region for metamodel.

Finally, if the dominating criterion is quality, then the best unsaturated simulated point of the promising region must first be found (see Equation 4.3). Denote this point by $\dot{\theta}_k \in \arg\max_{\theta_j \in \Theta_i} U_{jk}(1 - f'_{jk})$. Then a point is selected randomly with a multivariate normal distribution with mean $\dot{\theta}_k$ and covariance

matrix ω which is a user defined matrix. If this random point is feasible, it is simulated, otherwise the process of generating a point with the same distribution and checking the feasibility is repeated until a feasible point is found and simulated.

4.3. Adaptive Criteria Coefficients

The initial values of the coefficients s_k, m_k and q_k in Equation 4.1 are equal to 1/3, that is $s_1 = m_1 = q_1 = 1/3$, but these values may change at the beginning of next iterations based on the performance of the dominating criterion in previous iteration. The main rule here is that when quality (metamodel) dominates in an iteration and has served to find a "good" point, then the coefficient of the quality (metamodel) criterion is increased and those of the other two criteria are decreased. When a bad point is simulated with quality (metamodel) domination, the coefficient of the quality (metamodel) criterion is decreased and those of the other two criteria are increased. Whenever space dominates, we always decrease its coefficient and add to those of metamodel and quality criterion.

For $k = 1,2, \dots$, we set $s_k + m_k + q_k = 1$ and $0 < \delta_L \leq s_k, m_k, q_k \leq \delta_U \leq 1$ where δ_L and δ_U are user defined parameters. Also, we update coefficients this way:

$$q_{k+1} = q_k + \Delta q_{k+1}$$

$$m_{k+1} = m_k + \Delta m_{k+1}$$

$$s_{k+1} = s_k + \Delta s_{k+1}$$

where Δq_{k+1}, Δm_{k+1}, and Δs_{k+1} are the amount of change in associated coefficients defined as follows. We may face one of 3 cases (assume region i has been the promising region in iteration k):

Case 1. Quality is the dominating criterion of previous iteration, that is $\phi_{ik} = q_k Q_{ik}$. Assume j-th simulated point located in the promising region has been the best unsaturated point in iteration k and the new simulation point j' has been selected with respect to j.

a. If estimated objective function of point j' is better than point j'', that is $f'_{j'k} \leq f'_{jk}$ then the algorithm increases (incentivizes) the coefficient of Quality criterion and simultaneously decreases the coefficients of the other two criteria as follows:

$$\Delta q_{k+1} = \delta(\delta_U - q_k) \max\left\{\frac{f'_{jk} - f'_{j'k}}{f'_{jk}}, 1\right\}$$

$$\Delta m_{k+1} = -\frac{m_k - \delta_L}{s_k + m_k - 2\delta_L} \Delta q_{k+1}$$

$$\Delta s_{k+1} = -\frac{s_k - \delta_L}{s_k + m_k - 2\delta_L}\Delta q_{k+1}$$

where the parameter δ is a constant defined by user.

b. On the other hand, if the estimated objective function of new point j' is worse than the best unsaturated point j, that is $f'_{j'k} > f'_{jk}$ then the reverse way is done in which Quality Score is punished as follows:

$$\Delta q_{k+1} = -\delta(q_k - \delta_L)\max\left\{\frac{f'_{j'k} - f'_{jk}}{f'_{jk}}, 1\right\}$$

$$\Delta m_{k+1} = \frac{\delta_U - m_k}{2\delta_U - s_k - m_k}\Delta q_{k+1}$$

$$\Delta s_{k+1} = \frac{\delta_U - s_k}{2\delta_U - s_k - m_k}\Delta q_{k+1}$$

Case 2. Metamodel is the dominating criterion of previous iteration. The similar idea as case 1 is used, but here the last simulated point is compared with the best found simulated point.

Case 3. Space is the dominating criterion of previous iteration. We always reduce the coefficient of space criterion as follows:

$$\Delta s_{k+1} = -\delta'(s_k - \delta_L)$$

$$\Delta m_{k+1} = \frac{\delta_U - m_k}{2\delta_U - q_k - m_k}\Delta s_{k+1}$$

$$\Delta q_{k+1} = \frac{\delta_U - q_k}{2\delta_U - q_k - m_k}\Delta s_{k+1}$$

where constant δ' is defined by user.

4.4. Stopping Conditions

Many stopping criteria could be defined for GR. The followings are just a handful of examples:

- When the number of iterations exceed a pre-specified level. Equivalently, when the number of simulation replications reaches a maximum.
- When the number of stalled searches reaches a threshold, i.e. when the number of subsequent introduced optimums without any improvement in terms of estimated objective function reaches a maximum.
- When maximum total (simulation) optimization run time reaches.

An important question when the algorithm stops is which point should be introduced as the best. GR simply introduces the point with the best estimated objective function as the optimum, $\hat{\theta}_k^* \in$

$\arg\min_{j\in \mathrm{J}_k} \overline{L}_k(\theta_j)$. However, when GR is merged with ranking and selection procedures in the next section to form Hybrid GR, the introduced optimums sequence will converge to the global optimum.

5. Implementation Issues

In this Section, we provide some guidelines on how the GR could be implemented in practice.

5.1. How to partition

The method started out with partitioning the compact feasible region into a number of (sub) regions. There are a number of properties for a good partition. A region is the representative of a slice of feasible region in the competition of hunting new simulation runs allocated by the algorithm. In a good partition, the sizes (space measures) of the regions are equal. This way, we a priori let all the subsets of the feasible region compete for absorbing simulation runs fairly, that is proportional to their size (a kind of Bayesian thinking with uninformative priors). Then, the probability of selecting a partition would determine which regions are more worthy of further simulation. Moreover, the variation of the objective function of the points located in a region is possibly little in a good partition. Furthermore, the GR algorithm is more sensible when the points in each region are near each other.

A simple example of feasible region is a hyper rectangle like the test beds of numerical experiments of next chapter in which every decision variable has finite upper and lower bounds. A simple partition for this simple case divides the interval of each decision variable into a number of sub intervals, then the set of regions is the Cartesian product of the sets of subintervals of decision variables.

As another example, when the constraints are linear and the feasible region is convex but complicated, a partition plan first finds all the facets and extreme points of the feasible region located on each facet. Then if a point is picked from the interior of the feasible region, this point along with the set of extreme points on each facet could be the extreme points of a region in a partition scheme. One can split these obtained regions with similar fashion to get smaller regions or merge neighboring regions to get larger ones. For convex feasible regions with nonlinear constraints, a point could first be selected from the interior of the feasible region in a simple partition scheme, then if a coordinate space with origin at the selected point is considered, the range of each decision variable is divided into a positive and a negative subintervals, and hence the regions could be the Cartesian product of these subintervals. Each obtained

region could further be partitioned similarly. This partitioning idea is also useful for non-convex feasible region if regions with no feasible solutions are removed from the list of regions.

When the feasible region has very complicated shape, the last resort to partition the feasible region might be the following. First uniformly randomly select a possibly large population of points from the feasible region. Then define the number of desired regions. Using clustering techniques such as k-means and k-medians, the points are grouped into the number of desired regions. Then, each cluster is a representative of a region. In this kind of feasible region, we believe GR has practical momentum because most of the available methods in the literature require nice shapes of feasible region to explore the feasible region efficiently.

The number of regions involves a tradeoff. Fewer regions mean fewer computations for optimization process. With more regions, one can expect less variability in the objective function of the points in each region which facilitates the process of hunting the golden region containing the global optimum. The more complicated an objective function, the more regions are required. Hence, an improvement in original GR method discussed throughout the paper is that the GR starts out with a few regions, then as the search moves on, the partitioning scheme and the number of partitions are revised periodically aiming at having less function variability in each region.

5.2. Metamodeling

The most common classes of metamodels that have been proposed in the literature are regression, artificial neural networks (Abraham 2004) and Kriging (Barton and Meckesheimer 2006). There have been some studies to compare these metamodels (Kumar 2005), but no method is overwhelmingly better than all others. use a feed-forward artificial neural network as a global metamodel that is trained by Levenberg - Merquardt Algorithm (Nocedal and Wright. 1999 and Kermani et. al. 2005). We use mean squared error (MSE) of prediction as the performance measure of training the neural network. The proposed network has an input layer, an output layer and one or more hidden layers. The activity function of hidden layer neurons is tan-sigmoid. The number of neurons in input layer is the same as the number of decision variables and there is only one neuron in the output layer. The structure of the network adaptively changes in the algorithm (see Ma and Khorasani 2003 for such methods). We start out the network in the first iteration with one hidden layer that has only two neurons (Figure 4.3).

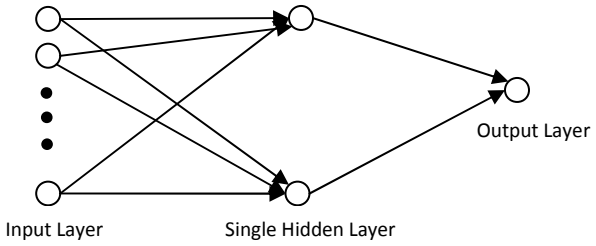


Figure 4.3: Initial neural network in iteration one

The initial population of points simulated before iteration one is trained to this network and the obtained Mean Squared Error (MSE) is compared with a user defined threshold (ν). If the obtained MSE is less than ν, the desired structure of the network has been found, otherwise, we train the network again with the same simulated points. This is repeated three times, if none of these three trials lead to a desired MSE, a neuron is added to the single hidden layer of the network. Then the algorithm tries to train the new structure at most three times until another neuron is added and so forth. If the number of neurons in the first hidden layer reaches a user defined maximum (e.g. two or three times the number of decision variable), the next neuron will start a second layer of neurons. Then we add new neurons to the second layer until this layer reaches maximum and so forth. This process continues until desired initial structure is found. Throughout the algorithm, the network can be updated in a similar manner.

5.3. Parameter Selection

For the size of initial population (υ), large values encourage exploration strategy at the beginning of the search, while small values help exploitation of the neighborhood of good quality points early in the search process. Of course for larger n values, more regions are needed. For 2-3 dimensions, $\upsilon \leq 20$ is recommended if nothing is known the complexity of the objective function. The value of η should be picked with regard to the intuition about the variability of the objective function.

The minimum value of criterion coefficients (δ_L) should be in (0,1/3). We recommend very little positive values for this parameter; this way we let one of the criterions overwhelmingly dominate the others if it can serve to find better and better points. The upper bound of the criterion coefficients should virtually be in (1/3,1). A value somewhere in the middle of this range should be fine. In addition, the parameters δ, δ' represent how rapidly the criterion coefficients can change. This parameter must be within (0,1].

We used a multivariate normal distribution with covariance matrix ω to pick a point near the best unsaturated point when quality dominates. We recommend off diagonal of zero for this matrix. The

variance components on the diagonal of this matrix must be adjusted with the scale of the objective function. Less value of a variance component makes the samples fall more close to the best unsaturated point.

The maximum number of possible searches in the saturation neighborhood of good quality points (U^*) involves a tradeoff between avoiding to search the vicinity of good points excessively with small values of U^* and exploiting the neighborhood of such points with larger values of U^*. In very early iterations of GR, more exploration is desired; therefore we might start out with very small values of U^* (say, less than 5) and increase this later on. Moreover, we recommend that the l-th component of the saturation scaling vector σ be set to the difference between the maximum and minimum feasible values of l-th decision variable (note that the feasible region is supposed to be compact). The value of $\xi \in (0,0.5)$ is like the radius of an n-sphere-shape saturation neighborhood as if the feasible region is a unit n-cube. We recommend a value around 0.05 for this parameter. If the user has intuition that the Kurtosis of the objective function near the local optima is high (low), (s)he should pick small (large) values for ξ. However, such a user must be God to have such an intuition in SO! ☺

6. GR Convergence Analysis and Hybrid GR

In this section, we study the convergence of GR method to a global optimum. We start with proving that GR satisfies the conditions of a PS method.

Theorem 4.1: GR is a PS method.

Proof: We first define the following events:

$\mathcal{H}_{ik}$: Region i is selected as promising region in iteration k

$\mathcal{C}_{ak}$: Criterion $a \in \{s, q, m\}$ dominates in iteration k

$\mathcal{D}_{il}^{\mathrm{G}}$: All l samples selected uniformly randomly from region i are in the set $\mathrm{G} \subseteq \Theta_i$

First consider an arbitrary subset $\mathrm{G} \subseteq \Theta_i$ with positive Lebesgue measure, that is $W(\mathrm{G}) > 0$. By definition of partitioning, at least a region i should be found such that

$$W(\Theta_i \cap \mathrm{G}) > 0 \tag{4.5}$$

Now:

$$\Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G}\} \geq \pi_{ik} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik}\} \tag{4.6}$$

Equation 4.4 and the fact $0 < \rho_L \leq A'_{ijk} \leq \rho_U < \infty$ (section 2.2.1.3) imply $0 < \frac{\rho_L}{\rho_U} \leq M_{ik}$. Then, because $0 < \delta_L$ (section 2.2.3), we have $\phi_{ik} \geq \delta_L M_{ik} \geq \delta_L \frac{\rho_L}{\rho_U} > 0$ (Equation 4.1). Since $s_k, S_{ik}, q_k, Q_{ik}, m_k, M_{ik} \leq 1$, Equation 4.1 implies $\phi_{ik} \leq 1$. Considering Equation 4.2, the fact that $\chi < \infty$ and the obtained lower and upper bound on ϕ_{ik} , we get:

$$\pi_{ik} \geq \frac{\delta_L \rho_L}{\chi \rho_U} > 0 \tag{4.7}$$

On the other hand, we have:

$$\begin{aligned} &\Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik}\} \\ &\quad = \Pr\{\mathcal{C}_{sk} | \mathcal{H}_{ik}\} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik} \cap \mathcal{C}_{sk}\} \\ &\quad + \Pr\{\mathcal{C}_{mk} | \mathcal{H}_{ik}\} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik} \cap \mathcal{C}_{mk}\} \\ &\quad + \Pr\{\mathcal{C}_{qk} | \mathcal{H}_{ik}\} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik} \cap \mathcal{C}_{qk}\} \end{aligned} \tag{4.8}$$

If space dominates, then selected candidates for next simulation point fall in $\Theta_i \cap \mathrm{G}$:

$$\begin{aligned} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik} \cap \mathcal{C}_{sk}\} &\geq \Pr\left\{\mathcal{D}_{i\beta'_i}^{\Theta_i \cap \mathrm{G}}\right\} \Pr\left\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} \middle| \mathcal{H}_{ik} \cap \mathcal{C}_{sk} \cap \mathcal{D}_{i\beta'_i}^{\Theta_i \cap \mathrm{G}}\right\} \\ &= \Pr\left\{\mathcal{D}_i^{\Theta_i \cap \mathrm{G}}\right\} = \left(\frac{W(\Theta_i \cap \mathrm{G})}{W(\Theta_\mathrm{i})}\right)^{\beta'} > 0 \ . \end{aligned} \tag{4.9}$$

If metamodel samples are selected uniformly randomly from each region, then we have:

$$\begin{aligned} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} | \mathcal{H}_{ik} \cap \mathcal{C}_{mk}\} &\geq \Pr\left\{\mathcal{D}_i^{\Theta_i \cap \mathrm{G}}\right\} \Pr\left\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} \middle| \mathcal{H}_{ik} \cap \mathcal{C}_{mk} \cap \mathcal{D}_i^{\Theta_i \cap \mathrm{G}}\right\} \\ &= \Pr\left\{\mathcal{D}_i^{\Theta_i \cap \mathrm{G}}\right\} = \left(\frac{W(\Theta_i \cap \mathrm{G})}{W(\Theta_\mathrm{i})}\right)^{\beta_i} > 0 \end{aligned} \tag{4.10}$$

Equation 4.2 and the fact that I use a multivariate normal distribution if quality dominates result in:

$$\Pr\left\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G} \middle| \mathcal{H}_{ik} \cap \mathcal{C}_{qk}\right\} \geq \min_{\theta \in \Theta_i} \left\{ \int_{x \in \Theta_i \cap \mathrm{G}} \Phi_X(x; \theta, \omega) dx \right\} > 0 \tag{4.11}$$

where $\Phi_X(x; \theta, \omega)$ is the probability density function of a multivariate normal random vector $X \in \mathbb{R}^n$ with mean θ and covariance matrix ω. Now let.

$$\varsigma_i(\mathrm{G}) = \min\left\{\left(\frac{W(\Theta_i \cap \mathrm{G})}{W(\Theta_i)}\right)^{\beta'}, \left(\frac{\mathrm{W}(\Theta_i \cap \mathrm{G})}{\mathrm{W}(\Theta_i)}\right)^{\beta_i}, \min_{\theta \in \Theta_i}\left\{\int_{x \in \Theta_i \cap \mathrm{G}} \Phi_X(x;\theta,\omega)dx\right\}\right\} \quad (4.12)$$

Equations 4.8, 4.9, 4.10, 4.11 and 4.12 and the fact that $\Pr\{\mathcal{C}_{sk}|\mathcal{H}_{ik}\} + \Pr\{\mathcal{C}_{mk}|\mathcal{H}_{ik}\} + \Pr\{\mathcal{C}_{qk}|\mathcal{H}_{ik}\} = 1$ imply:

$$\Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G}|\mathcal{H}_{ik}\} \geq \varsigma_i(\mathrm{G}) > 0 \quad (4.13)$$

Equations 4.6, 4.7 and 4.13 lead to:

$$\Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G}\} \geq \frac{\delta_L \rho_L \varsigma_i(\mathrm{G})}{\chi \rho_U} > 0 \quad (4.14)$$

Potentially, more than one region may be found such that $W(\Theta_i \cap \mathrm{G}) > 0$. Therefore:

$$\Pr\{\theta_{\upsilon+k} \in \mathrm{G}\} = \sum_{i=1}^{\chi} \Pr\{\theta_{\upsilon+k} \in \Theta_i \cap \mathrm{G}\} \geq \frac{\delta_L \rho_L}{\chi \rho_U} \sum_{i=1}^{\chi} \varsigma_i(\mathrm{G}) > 0 \text{ for } k = 1,2, \ldots \quad (4.15)$$

and clearly:

$$\lim_{k \to \infty} \Pr\{\theta_{\upsilon+k} \in \mathrm{G}\} \neq 0 \quad (4.16)$$

By Definition 1 and Equations 4.15 and 4.16, GR is a PS method. ■

Corollary 4.1: Let $X_{\mathrm{GR}}(\varepsilon)$ be the number of iterations required by GR method to visit a point θ in an ε-optimum interval for some $\varepsilon > 0$. Under Assumption 3.1, $\Pr\{X_{\mathrm{GR}}(\varepsilon) < \infty\} = 1$ by Theorem 3.1. ■

Proposition 4.1: Under Assumption 3.1, an upper bound on the expected number of iterations required by GR to visit a point in ε-optimum interval is:

$$\mathrm{E}\big(X_{\mathrm{GR}}(\varepsilon)\big) \leq \left(1 - \frac{W(\mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta)}{W(\Theta)}\right)^{\upsilon} \frac{\chi \rho_U}{\delta_L \rho_L \sum_{i=1}^{\chi} \varsigma_i(\mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta)}$$

where $\mathrm{B}(\theta^*; r_\varepsilon)$ is defined in Lemma 3.1.

Proof: Using the expected value of a geometric distribution and Equation 4.17, the inequality is easy to derive. ■

Note that the upper bound in proposition 1 could be computed only if $W(\mathrm{B}(\theta^*; r_\varepsilon) \cap \Theta)$ is known which is unusual in SO. Moreover, this bound is very conservative and based on pessimistically worst case scenarios. It essentially ignores many capabilities of different components of GR such as the usefulness of metamodel or quality criterion.

Finally, we propose GR to be used in a hybrid PS method for convergence purposes.

Definition 4.2: *When GR is merged with ranking and selection as the Hybrid PS Algorithm 3.2 outlines, the combination is called a Hybrid GR.* ■

Corollary 4.2: By Theorems 3.2 and 4.1 and under Assumption 3.1, the objective function of introduced optimum of any Hybrid GR converges in probability to that of the global optimum. ■

In next Chapter, we test the efficiency of GR method with a number of test problems.

Chapter 5: Efficiency of Golden Region Search

In this chapter, we test GR method versus a number of well-established methods in the literature. We first develop a theoretical basis for comparison of the methods. Then design two types of experiment based on the level of available information about the form of objective function. Numerical results of each experiment along with their interpretations are finally provided.

1. What is a good Simulation Optimization method?

There are many criteria for comparing different SO methods. Some researchers believe that SO methods should be provably convergent (Andradottir 2006). When an SO method is asymptotically convergent, we know that as the SO search progresses, the algorithm is "on the right path" to reach the optimum. But is a SO method "bad" if it doesn't converge or it cannot be shown to converge? Our answer is NO, though many SO researchers disagree. We believe there is a big difference between a method that "looks good" and a method that "is good". Mathematical asymptotic convergence theorems make an SO method "look good". This is neither enough, nor even necessary in practice.

Indeed, making an SO method convergent is not a big deal. We showed in Chapter 3 that any HPS method converges asymptotically. Any SO method can be a PS method if it is guaranteed that some random points from the feasible region are simulated periodically (Definition 3.1).

The ability of SO methods to visit good-quality points of the feasible region in less simulation runs is what we think one of the most important criteria for calling a method good. We refer to this ability as efficiency. Unfortunately, often times it is not easy to analytically show that an SO method is efficient compared to other methods in the literature. Not only does the literature lack such analytical studies as far as we am aware, but also little effort is seen in the literature to compare the efficiency of the newly proposed methods in a fair, comprehensive and statistically valid manner. In Section 3 of this Chapter,

we intend to provide a theory for comparing SO methods, and then we use this theoretical foundation to compare GR with three other methods.

In addition to efficiency of the search algorithm of an SO method, some SO practitioners prefer methods that are easy to implement. A user may find an SO method easy-to-implement if one or both of the followings are true:

1. Computer programming of the algorithm is simple or an SO software based on the method is available
2. Tuning the parameters of the SO method is easy, or the efficiency of the method is not too sensitive to its parameters.

Based on the first bullet point above, we think one of the disadvantages of the GR method is that coding GR is time-consuming because of the complexity of the search process of this method. We intend to address the second bullet point in the numerical results of this chapter.

2. Hypothesis Test

In this section, we propose a new test design to examine the efficiency of M competing optimization methods on a single test problem. Table 5.1 introduces the required notation.

Notation	Description
$\boldsymbol{m}$	Index of method
$\boldsymbol{X_m}$	Objective function of the last introduced optimum of method m
$\boldsymbol{p_m}$	Probability that method m beat the others
M	Number of competing methods
$\boldsymbol{x_{mr}}$	Observed value of the objective function of the last introduced optimum of method m
$\boldsymbol{R}$	Number of times each optimization method is tested
$\boldsymbol{\hat{p}_m}$	MLE estimate of p_m
$\boldsymbol{c}$	Critical value of the test
$\boldsymbol{\alpha'}$	Probability of Type-1 error
$\boldsymbol{\beta'_d}$	Probability of Type-2 error when $p_1 - \max_{m\neq 1} p_m = -d$
$\dot{\boldsymbol{\alpha}}$	Targeted Probability of Type-1 error
$\dot{\boldsymbol{\beta}}_{\boldsymbol{d}}$	Targeted Probability of Type-2 error when $p_1 - \max_{m\neq 1} p_m = -d$

Table 5.1: Notations of Experiment Design for Golden Region Method

Fix a simulation budget, which is the number of times simulation could be used to provide a value for the objective function of a point proposed by an optimization algorithm. Each method introduces an optimum when simulation budget is consumed; call this optimum the last introduced optimum. Each method is run R times on the test problem; let x_{mr} be the expected value of the objective function of the last introduced optimum in r-th repetition of method m. Assume X_m is the random variable from which observed values x_{mr} are drawn.

Let $p_m = \Pr\{X_m \leq \min_{m' \neq m} X_{m'}\}$ be the probability that method m does not introduce a worse point than points introduced by other methods when simulation budget is run out. Then, we are interested to test if a favorite method, say method 1, can beat all other methods. More precisely, our hypothesis test is:

$$\begin{cases} \mathrm{H}_0 : p_1 = \max p_m \\ \mathrm{H}_1 : p_1 < \max p_m \end{cases} \tag{5.1}$$

Let $\hat{p}_m = R^{-1} \sum_{r=1}^{R} \mathrm{I}[x_{mr} \leq \min_{m' \neq m} X_{m'}]$ where

$$\mathrm{I}[\mathcal{A}] = \begin{cases} 1 & \text{if } \mathcal{A} \text{ holds} \\ 0 & \text{o.w.} \end{cases}.$$

We accept the null hypothesis if and only if $\hat{p}_1 - \max \hat{p}_m \geq -c$ where $c \geq 0$ is a constant to be determined. Now we can define two error types of hypothesis tests.

Definition 5.1: *Type-one error in test (5.1) is probability(α'), that* H_0 *is rejected when it is correct. That is* $\alpha' = \Pr_{p_1 = \max p_m}\{\hat{p}_1 - \max \hat{p}_m < -c\}$. ■

Definition 5.2: *Type-two error in test (5.1) when* $p_1 - \max \hat{p}_m = -d < 0$ *is probability (* β'_d*) that* H_0 *is accepted. That is* $\beta'_d = \Pr_{p_1 - \max p_m = -d}\{\hat{p}_1 - \max \hat{p}_m \geq -c\}$. ■

The goal of the experiment design can now be defined:

Goal 5.1: *The goal is to find c and R such that type-one error be less than or equal to* $\dot{\alpha}$ *and type-two error be less than or equal to* $\dot{\beta}_d$ *when* $p_1 - \max p_m = -d$. ■

To reach this goal, we need to study the distribution of $\hat{p}_1 - \max \hat{p}_m$.

Theorem 5.1: $\hat{p}_m$ *is the MLE of* p_m.

Proof: See Example 7.2.7 in Casella and Berger (2001). ■

Theorem 5.2: *Let $\mathcal{P}$ be a vector of parameters of a certain distribution. If $\hat{\mathcal{P}}$ is MLE of $\mathcal{P}$, then for any function $h(\mathcal{P})$, the MLE of $h(\mathcal{P})$ is $h(\hat{\mathcal{P}})$.*

Proof: See proof of Theorem 7.2.10 in Casella and Berger (2001). ■

Corollary 5.1: *By theorems 5.1 and 5.2, $\hat{p}_1 - max\,\hat{p}_m$ is the MLE of $p_1 - max\,p_m$.* ■

MLEs have asymptotic normal distribution under some regularity conditions (Casella and Berger 2001). If R turns out to be a large number, the assumption of normality of $\hat{p}_1 - max\,\hat{p}_m$ is justifiable. So we simply assume this.

Assumption 5.1: *$\hat{p}_1 - max\,\hat{p}_m$ has a normal distribution.* ■

Proposition 5.1: $Var(\hat{p}_1 - max\,\hat{p}_m) \leq R^{-1}$.

Proof: $Var(\hat{p}_1 - max\,\hat{p}_m) = Var(\hat{p}_1) + Var(max\,\hat{p}_m) - 2Cov(\hat{p}_1\,, max\,\hat{p}_m) \leq Var(\hat{p}_1) + max\ \{Var(\hat{p}_m)\} - 2Cor(\hat{p}_1\,, max\,\hat{p}_m)\sqrt{Var(\hat{p}_1)Var(max\,\hat{p}_m)} \leq Var(\hat{p}_1) + max\ \{Var(\hat{p}_m)\} + 2\sqrt{Var(\hat{p}_1)\,max\,\{Var(\hat{p}_m)\}}$. Since p_m could be deemed as success probability of a binomial distribution, $Var(\hat{p}_m) = \hat{p}_m(1-\hat{p}_m)R^{-1}$ and hence $Var(\hat{p}_m) \leq 0.25R^{-1}$. Therefore, $max\ \{Var(\hat{p}_m)\} \leq 0.25R^{-1}$. Consequently,

$Var(\hat{p}_1 - max\,\hat{p}_m) \leq 0.25R^{-1} + 0.25R^{-1} + 2\sqrt{0.25^2R^{-2}} \leq R^{-1}$. ■

Theorem 5.3: *Under Assumption 5.1, the following values for R and c satisfy Goal 5.1.*

$$R = \left(\frac{Z_{1-\dot{\beta}_d} + Z_{1-\dot{\alpha}}}{d}\right)^2 \tag{5.2}$$

$$c = \frac{Z_{1-\dot{\alpha}}}{\sqrt{R}} \tag{5.3}$$

where Z_q is the q-th quantile of standard normal distribution.

Proof: Let Z be a random variable with standard normal distribution. Now:

$$c = \frac{Z_{1-\dot{\alpha}}}{\sqrt{R}} \xrightarrow{\textbf{Proposition 5.1}} c \geq Z_{1-\dot{\alpha}}\sqrt{\mathrm{Var}(\hat{p}_1 - \max\hat{p}_m)} \Longrightarrow \frac{-c-0}{\sqrt{\mathrm{Var}(\hat{p}_1 - \max\hat{p}_m)}} \leq -Z_{1-\dot{\alpha}}$$

$$\Rightarrow \Pr_{p_1=\max p_m}\left\{Z < \frac{-c-(p_1-\max p_m)}{\sqrt{\mathrm{Var}(\hat{p}_1-\max\hat{p}_m)}}\right\} \leq \dot{\alpha} \Longrightarrow \Pr_{p_1=\max p_m}\{\hat{p}_1 - \max\hat{p}_m < -c\}$$

$$\leq \dot{\alpha}$$

Also from (5.2) and (5.3)

$$\Rightarrow c = d - \frac{Z_{1-\dot{\beta}_d}}{\sqrt{R}} \;\; c \le d - Z_{1-\dot{\beta}_d}\sqrt{\mathrm{Var}(\hat{p}_1 - \max \hat{p}_m)} \;\Rightarrow\; \frac{-c+d}{\sqrt{\mathrm{Var}(\hat{p}_1 - \max \hat{p}_m)}} \ge Z_{1-\dot{\beta}_d}$$

$$\Rightarrow \; Pr_{p_1 - \max p_m = -d}\left\{ Z \ge \frac{-c - (p_1 - \max p_m)}{\sqrt{\mathrm{Var}(\hat{p}_1 - \max \hat{p}_m)}} \right\} \le \dot{\beta}_d$$

$$\Rightarrow \; Pr_{p_1 - \max p_m = -d}\{\hat{p}_1 - \max \hat{p}_m \ge -c\} \; \le \dot{\beta}_d$$

. ■

3. Experiment Design

3.1. Test Problems

There are two types of test problems that could possibly be used to compare the efficiency of SO methods:

1. Real simulation: The obvious choice is to test the methods on optimization problems defined on some stochastic systems. That is, a number of real or artificial stochastic systems of interest are considered, and then the optimization algorithms use simulation of the stochastic systems to get samples of objective function values and optimize the performance of the system.
2. Simulation surrogate: Some closed-form functions of interest are designed, then whenever an optimization algorithm wants a sample of the objective function of a certain point, the value of the closed-form function evaluated in this point is added to a random noise and is returned as the sample to the optimization algorithm. In fact, the form of the objective function is known, but this knowledge is not used in the experiments. The random noise mimics the role of the randomness in simulation outputs.

Each type of test problems has their own advantages and disadvantages. The advantages of real simulation are the followings:

1. The experiments would be more realistic. Closed-form functions may not accurately represent real-simulation outputs.
2. There are some simulation optimization methodologies which do not look at simulation as a black-box. Using real simulation, such methods could also be competed with newly developed methods in the experiments.

And the disadvantages of real simulation are:

1. The global optimum of the optimization problem defined on the stochastic system of interest is usually unknown or hard to be known. But we probably need to know the optimum to compare different methods based on.
2. Since little to nothing is known about the properties of the response surface of an optimization problem defined on a stochastic system, it is hard to diversify the test beds. Complexity of simulation models of the stochastic systems does not necessarily mean complexity and consequently diversity of the response surfaces. This is something which has been ignored in some papers (Rosen and Harmonosky 2005).

The advantages (disadvantages) of simulation surrogates are the disadvantages (advantages) of real simulation.

In the literature, both test beds have been used. For instance, Paul and Chanev (1998) use real simulation of a steel-making company to evaluate the efficiency of Genetic Algorithm. Ahmed and Alkhamis (2002) use both test beds to examine the effectiveness of their combined method of Simulated Annealing and Ranking and Selection. One of these test beds is a stochastic optimization problem with 20 designs and the other one is the inventory control of a stock. Rosen and Harmonosky (2005) have tested their Simulated Annealing method with simulation of a number of real manufacturing systems. Laguna and Marti (2002) also use both test beds. Keys and Rees (2004) have also used closed-form objective functions to test their nonparametric metamodels.

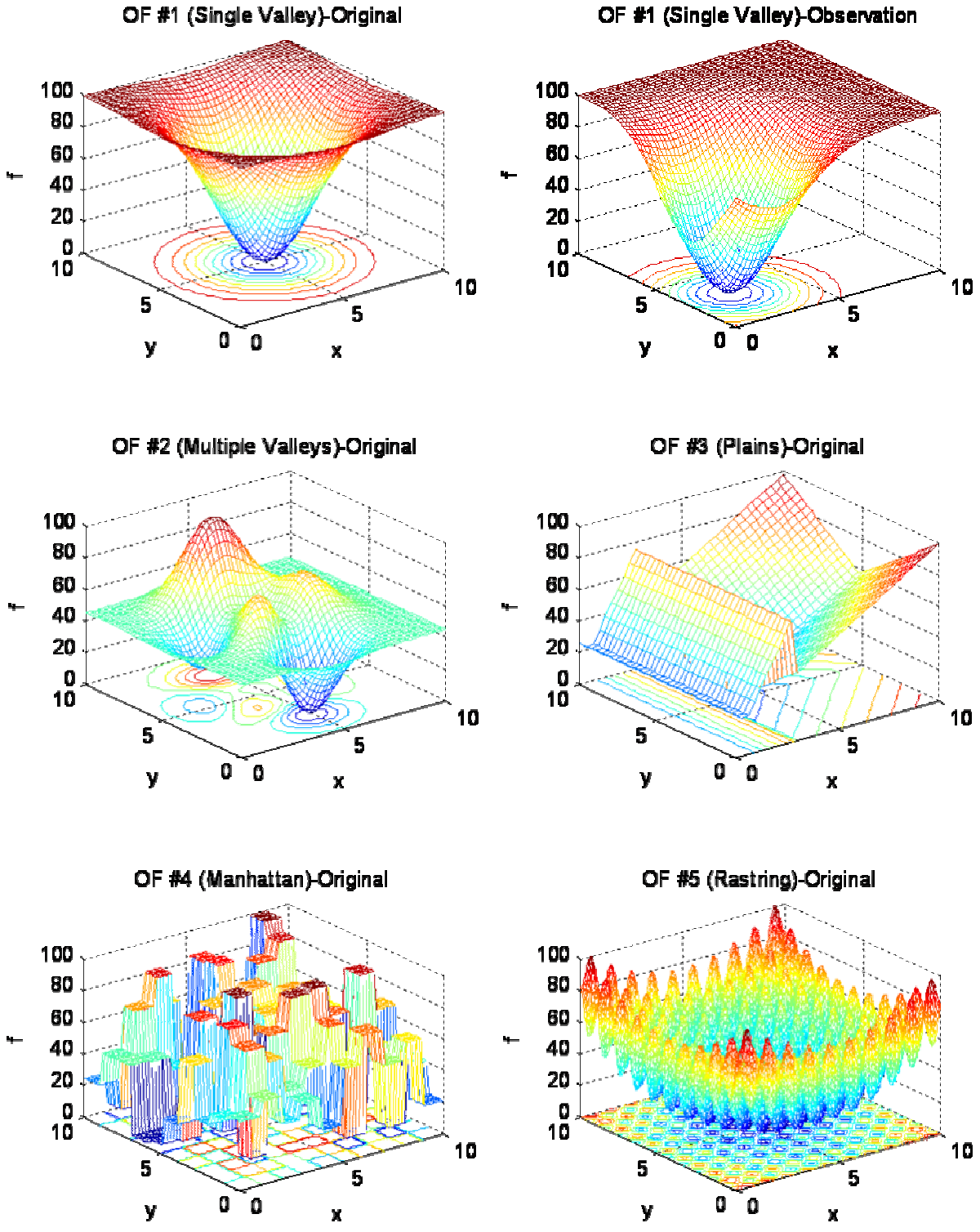


Figure 5.1: Test Problems

In our experiments, we use closed-form functions (simulation surrogates). Specifically, there are five two-dimensional objective functions and one three-dimensional problem. All the decision variables are bounded between 0 and 10. We test these six problems with a small random noise. We also test two of the two-dimensional problems with large random noise. Figure 5.1 shows the shape of the five two-dimensional test problems we will use. The Appendix of this dissertation contains MATLAB codes of two-dimensional objective functions. In summary, we have 8 test problems:

Test Problem 1: Objective function 1 (Single Valley) with small noise. This function represents the simplest possible objective functions with just one global optimum. We add a random number with standard normal distribution (low variance) whenever a sample of the objective function is required.

Test Problem 2: Objective function 2 (Multiple Valleys) with small noise. This function is a response surface with multiple local optima, but still with a smooth shape. We add standard normal random number again to this function.

Test Problem 3: Objective function 3 (Plains) with small noise. This one represents non-differentiable surfaces with some very bad points in the close neighborhood of the global optimum. We again add the standard normal noise to this function.

Test Problem 4: Objective function 4 (Manhattan) with small noise. This one challenges the capability of optimization methods in exploring the whole feasible region. The structure of the objective function is such that the decision variables are almost like qualitative variables. We add the standard normal noise again.

Test Problem 5: Objective function 5 (Rastring) with small noise. This is a kind of very complex objective function with numerous local optima. We add standard normal variable again.

Test Problem 6: Objective function 1 (Single Valley) with large noise. The function is like the first test bed except that we add a zero mean normal random number with variance 10 (high variance) to the expected value of the objective function.

Test Problem 7: Objective function 2 (Multiple Valleys) with large noise. The function is like the second test bed except that we add a zero mean normal random number with variance 5 (high variance) to the expected value of the objective function.

Test Problem 8: Objective function 6 (Three Dimensional-3D) with small noise. The function is like the first test bed except that it is 3-dimensional. We add standard normal variable again.

Each objective function has an "original" format with a certain location for the optimum. We have designed all 6 objective functions such that the the global optimum could be uniformly randomly relocated to anywhere in the feasible region. When relocated, the resulting objective function is called an "observation". In Figure 5.1, there is an observation of objective function 1.

3.2. Competing Methods and Their Parameters

3.2.1 Methods

Selection of methods to compete with GR in the experiments of this section is a difficult job. There are many methods in the literature. We certainly cannot bring into play all, this can be the subject of a new dissertation, so we have to select a subset. Then, an immediate question would be why did we choose these methods, why not the others? We simply don't have a firm answer to this question, all we can tell are our rough ideas about methods which seem more appropriate.

Naturally, methods for continuous SO problems are immediate choices because by definition, the target problem for GR is continuous. So we select Stochastic Approximation (SA) as the very first and sometimes the only method classified as a continuous method in most surveys. We believe metaheuristics are an important class of optimization in modern operations research. We pick Genetic Algorithm (GA) among a pool of very many metaheuristics. There are some studies which show GA outperforms some other metaheuristics for SO problems (Jones and White 2004 and Magoulas et. al. 2002). Also the fact that our coding platform, MATLAB software, has built-in functions for GA contributes in selecting this method. SA and GA are both representative of old and middle generation of SO methods. We'd like to pick one last competing method from a more recent method. Model Reference Adaptive Search (MR) (Hu et. al. 2005 and 2007) method is a newly proposed method which is convergent is believed to be efficient by some researchers. Hence we pick this method along with SA and GA.

We do not use R&S add-on in GR method because competing methods are not convergent too. We only test the ability of the methods in hitting good quality points, not necessarily their ability to recognize the quality of the hit points.

To read more about the GA algorithm used in our experiment, refer to GA toolbox in MATLAB7 Software Help. This software is developed by MathWorks^{TM} (www.mathworks.com). To see more details of MR, refer to the algorithm of Model Reference Adaptive Search 1 in Hu et. al. (2007).

We outlined the "Localized" Stochastic Approximation in Chapter 1-Algorithm 1.1. We propose algorithm below to make SA globalized. We used the same notation used there (which should not be confused with other notations in the dissertation) except some modifications explained in the algorithm. There are a number of user-defined parameters for Stochastic Approximation in Algorithm 5.1.

Algorithm 5.1: *Globalized Stochastic Approximation*

6. *Set path step counter $n = 0$. Set simulation counter $s = 0$. Set bad-move counter $m = 0$.*
7. *Select an initial guess $\theta_1 \in \Theta$ randomly from the feasible region. Set $s = s + 1$ and $n = n + 1$*
8. *Set $a_n = an^{-1}$.*
9. *If gradient estimation method $g = FM$, use Forward Method (FM) of gradient estimation by setting i-th coordinate of gradient estimator $\hat{f}_i'(\theta_n) = \frac{\hat{f}(\theta_n + cn^{-1/4}e_i) - \hat{f}(\theta_n)}{cn^{-1/4}}$ for $i = 1,2,\dots,d$ where e_i is i-th unit vector and $\hat{f}(\theta)$ is an estimate of the objective function. Also set $s = s + d + 1$. Otherwise if $g = CM$, use Central Method (CM) via $\hat{f}_i'(\theta_n) = \frac{\hat{f}(\theta_n + (c/2)n^{-1/6}e_i) - \hat{f}(\theta_n - (c/2)n^{-1/6}e_i)}{cn^{-1/6}}$ and set $s = s + 2d$.*
10. *Set $\theta_{n+1} = \pi_\Theta\left(\theta_n - an^{-1}\hat{f}'(\theta_n)\right)$ where $\pi_\Theta(\theta) \in \Theta$,for all $\theta \in \mathbb{R}^d$, is the closest point to θ such that when $\theta \in \Theta$, then $\pi_\Theta(\theta) = \theta$. If $f(\theta_{n+1}) > f(\theta_n)$, set $m = m + 1$ otherwise set $m = 0$.*
11. *If $s \geq SB$ where SB is the simulation budget for the optimization process, introduce an optimum based on a user-defined method and the first SB simulated points, then terminate the algorithm.*
12. *If $\hat{f}'(\theta_n) < MinG$ and/or $m > MaxB$, then set $n = 0$, $m = 0$, and go to step 2.*

3.2.2 Parameters

Due to the finite life that we have in this world, we cannot test all possible combination of parameters of the competing methods to pick the best. We mean assume a method has 10 parameters. If we have to consider 10 different values for each parameter, then we have to test 10^{10} combinations of parameters for each method. Observe that the problem of finding the best parameters of each method is a kind of optimization with expensive objective function! Expensive because knowing the quality of each combination requires us to test it on some test bed(s) many times. So we simply give up considering all possible combinations. What we do is that we first pick a subset of parameters of each method, a subset which we believe would be more effective on the performance of the method. Then we consider limited combination of different values of the selected parameters. Overall, we consider 20 sets of parameters for each method.

3.2.2.1 Golden Region (GR)

We believe some of the parameters of GR do not have a significant effect on the performance of the method. So we fix these and just consider variations in others. The insensitive parameter values are: $U^* = 10,\ \delta_L = 10^{-8},\ \delta_U = 0.6, \delta = 0.2, \delta' = 0.1, \beta_i = 50, \beta' = 50$, for all i, $\omega = 8I$ where I is identity matrix,

$\rho_L = 0.1, \rho_U = 1, \sigma_l = 10\ \ \forall l, \xi = 0.05$. We do not use R&S procedure at all in these experiments, therefore parameters τ, α_h, γ_h are inapplicable. We use simulation surrogate as test functions, so we just take a single sample per point. Hence $\eta = 1$. The training data of the neural net is scaled between -1 and 1 and we used $\nu = 0.005$. We increased this MSE threshold if the number of attempts to reach a suitable structure of the network reaches a maximum. We update the neural network for GR method for a typical iteration k if $k/\left(1 + \left\lfloor \frac{k}{100} \right\rfloor\right)$ is equal to $\left\lfloor k/\left(1 + \left\lfloor \frac{k}{100} \right\rfloor\right)\right\rfloor$ where $\lfloor . \rfloor$ is round-down function. We set $\mathrm{N}_{ii} = 1$ for all i and let $\mathrm{N}_{ii'}$ for $i \neq i'$ be relative to reciprocal of Euclidean distance of the center points of square-shape regions i and i'.

We consider partitioning scheme and the number of initial population as two sensitive settings. All decision variables in test beds are bounded between 0 and 10. We fix three positive integer values a_1, a_2 and a_3. Then we partition the 0-10 length of each decision variable into a_c pieces. Then every combination of one piece from each decision variable would be a (sub)region of the GR algorithm. For the two-dimensional test beds $(d = 2)$, we set $a_1 = 3\ a_2 = 5, a_3 = 10, a_4 = 15$. For test bed 8 $(d = 3)$, we set $a_1 = 4, a_2 = 5, a_3 = 6, a_4 = 7$. We also consider 3 values for $\upsilon \in \{2,5,20,100\}$. Table 5.1 shows the 20 sets of parameters for GR method.

Set	υ	χ
1	a_1^d	2
2	a_1^d	5
3	a_1^d	20
4	a_1^d	100
5	a_1^d	200
6	a_2^d	2
7	a_2^d	5
8	a_2^d	20
9	a_2^d	100

10	a_2^d	200
11	a_3^d	2
12	a_3^d	5
13	a_3^d	20
14	a_3^d	100
15	a_3^d	200
16	a_4^d	2
17	a_4^d	5
18	a_4^d	20
19	a_4^d	100
20	a_4^d	200

Table 5.1: Sets of Parameters for GR

3.2.2.2 Stochastic Approximation (SA)

We fix $MinG = 0.05$, $MaxB = 20$. The other parameters of this method are varied as Table 5.2 shows.

Set	g	a	c
1	FM	2	3
2	FM	2	5
3	FM	2	7
4	FM	5	3
5	FM	5	5
6	FM	5	7
7	FM	7	3
8	FM	7	5
9	FM	7	7
10	FM	9	7
11	CM	2	3
12	CM	2	5
13	CM	2	7
14	CM	5	3

15	CM	5	5
16	CM	5	7
17	CM	7	3
18	CM	7	5
19	CM	7	7
20	CM	9	7

Table 5.2: Sets of Parameters for SA

3.2.2.3 Model Reference Adaptive Search 1 (MR)

we use multivariate normal distribution as the "family of parameterized distributions" in MRAS1 (see Hu et. al. 2007 for definitions and notations) and used the same parameter settings as those used in section 6 of Hu et. al. (2007) except the some parameters which are varied as Table 5.3 shows:

Set	N_0	ρ_0	ε
1	10	0.1	0. 1
2	10	0.1	0. 01
3	10	0.05	0. 1
4	10	0.05	0. 01
5	20	0.1	0. 1
6	20	0.1	0. 01
7	20	0.05	0. 1
8	20	0.05	0. 01
9	50	0.1	0. 1
10	50	0.1	0. 01
11	50	0.05	0. 1
12	50	0.05	0. 01
13	100	0.1	0. 1
14	100	0.1	0. 01
15	100	0.05	0. 1
16	100	0.05	0. 01
17	200	0.1	0. 1
18	200	0.1	0. 01

19	200	0.05	0. 1
20	200	0.05	0. 01

Table 5.3: Sets of Parameters for MR

3.2.2.4 Genetic Algorithm (GA)

Genetic Algorithm in MATLAB is a powerful environment for optimization of virtually any function using GA. The problem is that the method has a lot of parameters. We use default settings in MATLAB for most of the parameters except those varied in Table 5.4. We also use "Uniform" as mutation function.

Set	Population Size	Mutation Rate	Crossover Function
1	10	0.01	Scattered
2	10	0.1	Single Point
3	10	0.1	Scattered
4	10	0.3	Single Point
5	10	0.3	Scattered
6	20	0.01	Scattered
7	20	0.1	Single Point
8	20	0.1	Scattered
9	20	0.3	Single Point
10	20	0.3	Scattered
11	50	0.01	Scattered
12	50	0.1	Single Point
13	50	0.1	Scattered
14	50	0.3	Single Point
15	50	0.3	Scattered
16	100	0.01	Scattered
17	100	0.1	Single Point
18	100	0.1	Scattered
19	100	0.3	Single Point
20	100	0.3	Scattered

Table 5.4: Sets of Parameters for GA

3.3. Parameter Tuning and Experiments

The selection of parameters of an optimization method usually has a huge effect on the efficiency of the method. In SO where the objectives function is deemed unknown, tuning the parameters of the optimization method of interest is more challenging. We will consider two types of experiments based on the level of knowledge about the form of the objective function in SO. In the 1st experiment, we assume the SO practitioner knows the shape of the objective function, but he/she doesn't know the location of the optimum. We specifically tune the parameters of each method for each Test Bed. This set of Parameters is called Specially Best Parameters (SBP). Then we use each method with its SBPs for each Test Bed. In the second Experiment, we assume little is known about the form of the objective function and hence the parameters of each method may not specially be adjusted for each Test Bed. We use an Overall Best Parameter (OBP) set for each method on Test Beds.

Before discussing this issue, we present an algorithm to find SBPs and OBP for a typical method m.

Algorithm 5.2: *Finding Specially and Overall Best Parameters of Each Method*

1. *For* $b = 1,2,\dots,8$ *, do:*
   - *1.1. Take 100 random observations of the objective function in test bed* b*.*
   - *1.2. Let* (m, s) *represent method* m *with parameter set* s*.*
   - *1.3. For* $s = 1,2,\dots,20$*, do*
     - *1.3.1.For* $r = 1,2,\dots,20$*, do:*
       - *1.3.1.1. Using a simulation budget of 1000 runs, optimize observation* r *of step 1.1 with method* (m, s) *with designated random noise of Test Bed* b*. Let* $t_{(m,s)r}^{(b)}$ *be the expected value of the objective function of the best simulated point when the simulation budget is consumed in this optimization process.*
     - *1.3.2.End For*
   - *1.4. End For*
2. *End For*
3. *Let* $s_m^{(b)}$ *denote the SBP set of method* m *on Test Bed* b*. Then* $s_m^{(b)} \in \arg\min_{s=1,2,\dots,20} \sum_{r=1}^{20} t_{(m,s)r}^{(b)}$ *for* $b = 1,2,\dots,8$ *.*
4. *Let* s_m^* *be the OBP set of method* m *on the first 5 test beds. Then* $s_m^* \in \arg\min_{s=1,2,\dots,20} \sum_{b=1}^{5} \sum_{r=1}^{20} t_{(m,s)r}^{(b)}$.

We do two types of experiment for two different situations:

Experiment 1: *Specially Best Parameters-Examining the efficiency of the competing methods on all 8 test problems with Specially Best Parameters (SBP) for each method on each test problem.*

Algorithm 5.3:

1. *For* $b = 1,2, \dots, 8$ *, do:*
   - *1.1. Take* R *random observations of the objective function in test bed* b*.*
   - *1.2. Let* (m, s) *represent method* m *with parameter set* s*. Let* $\hat{p}_{(m,s)}^{(b)}$ *be the domination probability of method* (m, s) *on test bed* b*.*
   - *1.3. Let* $\hat{p}_{(m,s_m^{(b)})}^{(b)} = 0$ *for* $m = GR, SA, MR, GA$*.*
   - 1.4. *For* $r = 1,2, \dots, R$*, do:*
     - 1.4.1.For $m = \text{GR, SA, MR, GA}$
       - *1.4.1.1. Using a simulation budget of 1000 runs, optimize observation* r *of step 1.1 with method* $\left(m, s_m^{(b)}\right)$ *with designated random noise of test bed* b*. Let* $x_{\left(m,s_m^{(b)}\right)r}^{(b)}$ *be the expected value of the objective function of the best simulated point when the simulation budget is consumed in this optimization process.*
     - *1.4.2.End For*
     - *1.4.3.Let* $m' \in \arg\min_{m=\text{GR,SA,MR,GA}} x_{\left(m,s_m^{(b)}\right)r}^{(b)}$.
     - *1.4.4.Let* $\hat{\text{p}}_{(m',s_{m'}^{(b)})}^{(\text{b})} = \hat{\text{p}}_{(m',s_{m'}^{(b)})}^{(\text{b})} + 1$. *For* $m = \text{GR, SA, MR, GA}$ *and* $m \neq m'$ *let* $\hat{\text{p}}_{(m,s_m^{(b)})}^{(\text{b})} = \hat{\text{p}}_{(m,s_m^{(b)})}^{(\text{b})}$.
   - *1.5. End For*
   - *1.6. For* $m = \text{GR, SA, MR, GA}$ *let* $\hat{\text{p}}_{(m,s_m^{(b)})}^{(\text{b})} = \frac{\hat{\text{p}}_{(m,s_m^{(b)})}^{(\text{b})}}{R}$ *and the average value* $\bar{x}_{(m,s_m^{(b)})}^{(b)} = R^{-1} \sum_{r=1}^{R} x_{\left(m,s_m^{(b)}\right)r}^{(b)}$
2. *End For*

Experiment 2: *Examining the efficiency of the competing methods on test problems 1 to 5 with an Overall Best Parameter set (OBP) for each method on all 5 test problems Specially Best Parameters-*

Algorithm 5.4:

1. *For* $b = 1,2, \dots, 5$ *, do:*

1.1. Take R random observations of the objective function in test bed b.

1.2. Let (m, s) represent method m with parameter set s. Let $\hat{p}_{(m,s)}^{(b)}$ be the domination probability of method (m, s) on test bed b.

1.3. Let $\hat{p}_{(m,s_m^*)}^{(b)} = 0$ for $m = GR, SA, MR, GA$.

1.4. *For $r = 1,2, \dots, R$, do:*

1.4.1.For $m = \text{GR, SA, MR, GA}$

1.4.1.1. Using a simulation budget of 1000 runs, optimize observation r of step 1.1 with method (m, s_m^*) with designated random noise of test bed b. Let $x_{(m,s_m^*)r}^{(b)}$ be the expected value of the objective function of the best simulated point when the simulation budget is consumed in this optimization process.

1.4.2.End For

1.4.3.Let $m' \in \arg\min_{m=\text{GR,SA,MR,GA}} x_{(m,s_m^*)r}^{(b)}$.

1.4.4.Let $\hat{\text{p}}_{(m',s_{m'}^*)}^{(\text{b})} = \hat{\text{p}}_{(m',s_{m'}^*)}^{(\text{b})} + 1$. For $m = \text{GR, SA, MR, GA}$ and $m \neq m'$ let $\hat{\text{p}}_{(m,s_m^*)}^{(\text{b})} = \hat{\text{p}}_{(m,s_m^*)}^{(\text{b})}$.

1.5. End For

1.6. For $m = \text{GR, SA, MR, GA}$ let $\hat{\text{p}}_{(m,s_m^*)}^{(\text{b})} = \frac{\hat{\text{p}}_{(m,s_m^*)}^{(\text{b})}}{R}$ and the average value $\bar{x}_{(m,s_m^*)}^{(b)} = R^{-1} \sum_{r=1}^{R} x_{(m,s_m^*)r}^{(b)}$

2. End For

4. Numerical Results

Table 5.4 shows SBP and OBP values found using Algorithm 5.2.

	GR	SA	MR	GA
SBP for Test Bed 1	7	4	6	6
SBP for Test Bed 2	8	14	9	8
SBP for Test Bed 3	18	11	5	13
SBP for Test Bed 4	11	10	13	5
SBP for Test Bed 5	18	1	13	14
SBP for Test Bed 6	7	4	6	6
SBP for Test Bed 7	8	14	9	8

SBP for Test Bed 8	3	4	6	11
OBP	12	5	11	10

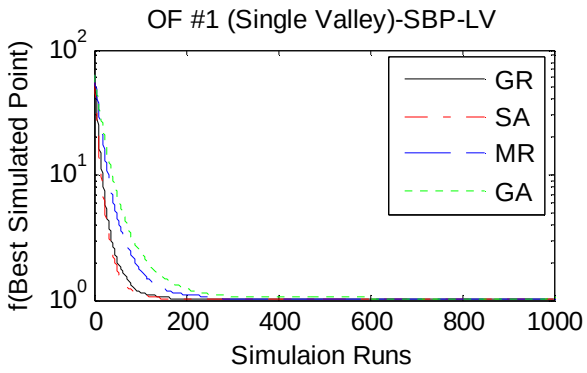
Table 5.5: SBP and OBP of the Experiments

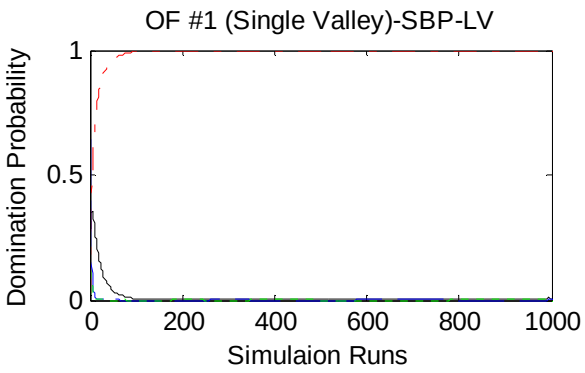
We provide two graphs for each test bed of each Experiment.

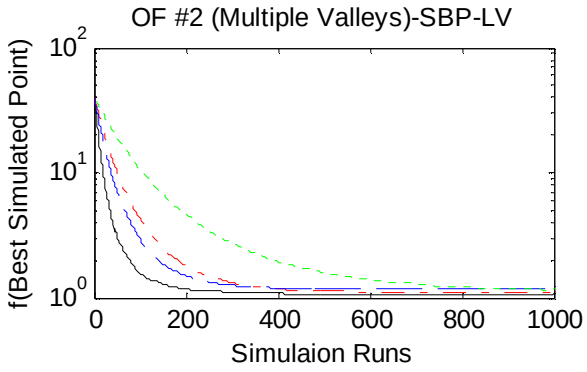
4.1. Experiment 1 with best parameters for each Objective Function

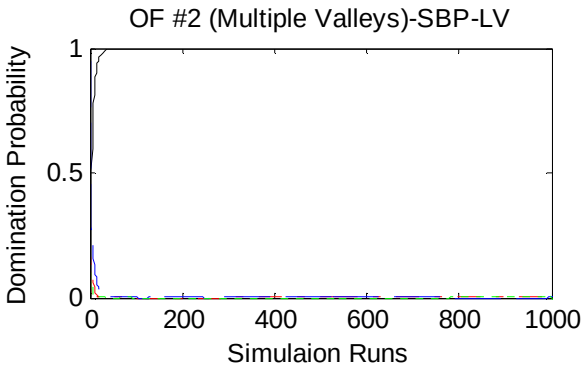
Figure 5.2 contains the two plots for each test bed in Experiment 1. The left-column plots show average of the expected value of the objective function of the best simulated points as more simulation budget is injected in the SO process, that is the values $\bar{x}^{(b)}_{(m,s_m^{(b)})}$. The right graphs show domination probabilities $\hat{p}^{(b)}_{(m,s_m^{(b)})}$. Our interpretations of these graphs are as follows:

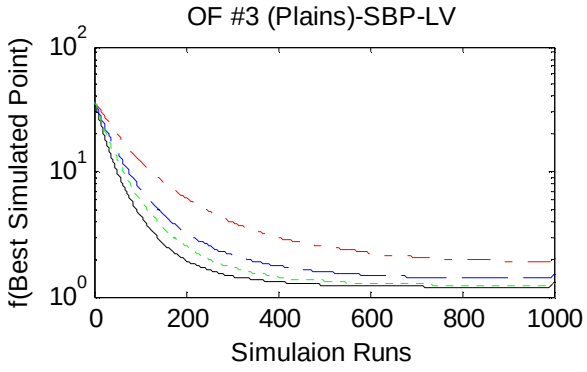
1. In Test Bed 1, the hypothesis test (5.1) is rejected (assume GR is method 1). This means SA does a better job than GR when its parametrs are tuned specifically for this Test Bed. In Test Bed 5, GA finds better solutions in the second half of the simulation budget and hypothesis (5.1) is rejected. In every other test bed, GR dominates other methods after a short initial warm-up period in the optimization process.
2. Even in Test Beds 1 and 5 where SA and GA beat GR respectively, their performance might not be considered significantly better. Looking into the $\bar{x}^{(b)}_{(m,s_m^{(b)})}$ graphs reveals that the performance curve of GR is just mariginally above that of diminating method.
3. When there is a High Variance (HV) in the outputs of simulation, SA is more vulnerable. This could be seen by comparing test bed 1 and 6 for instance.
4. In Test Bed 8 with dimension 3, the performance is pretty much proportionally similar to Test Bed 1.

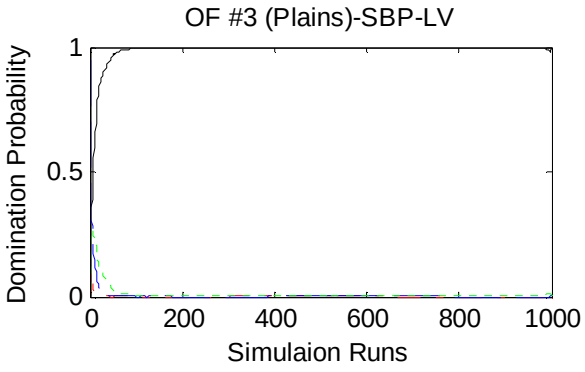

OF #1 (Single Valley)-SBP-LV
f(Best Simulated Point)
Simulaion Runs
GR
SA
MR
GA

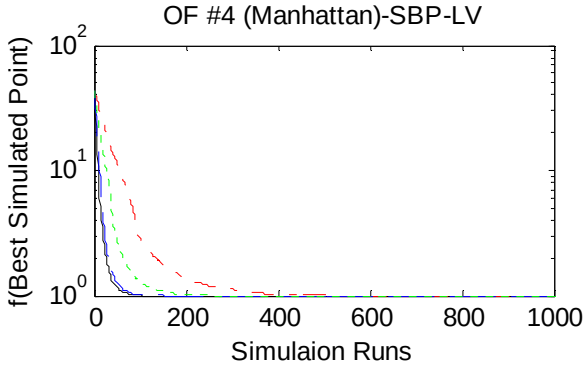

OF #1 (Single Valley)-SBP-LV
Domination Probability
Simulaion Runs

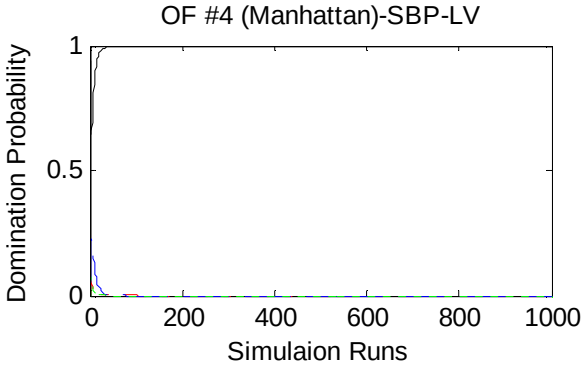

OF #2 (Multiple Valleys)-SBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #2 (Multiple Valleys)-SBP-LV
Domination Probability
Simulaion Runs


OF #3 (Plains)-SBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #3 (Plains)-SBP-LV
Domination Probability
Simulaion Runs


OF #4 (Manhattan)-SBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #4 (Manhattan)-SBP-LV
Domination Probability
Simulaion Runs

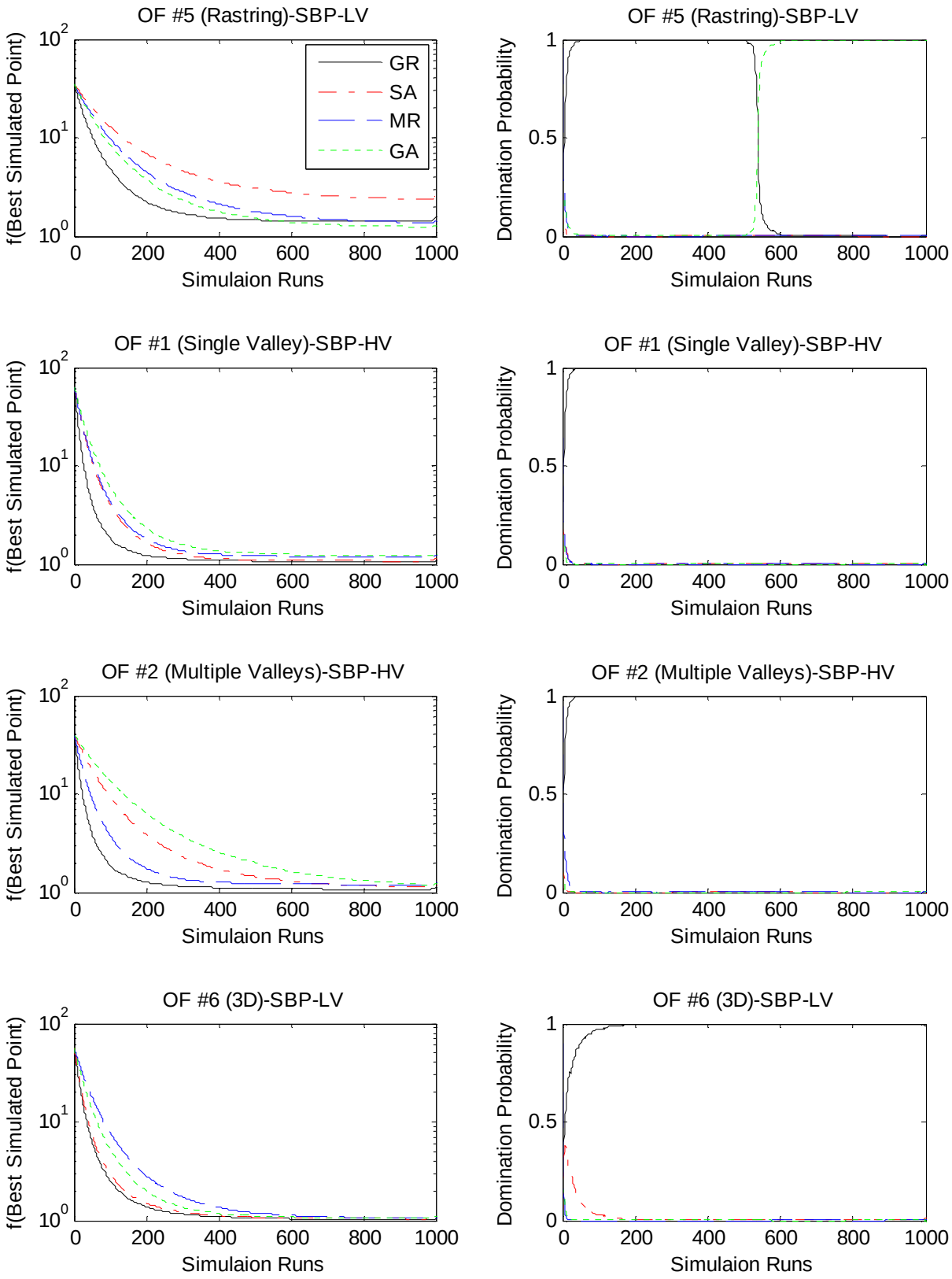
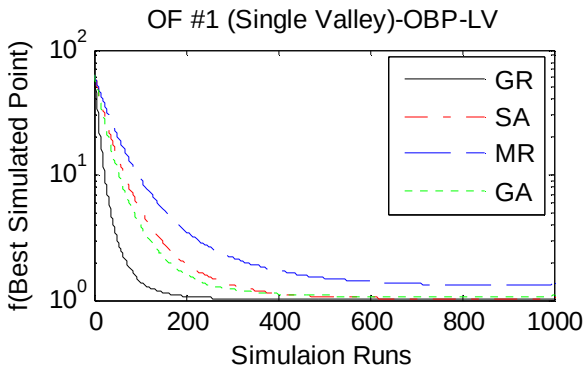


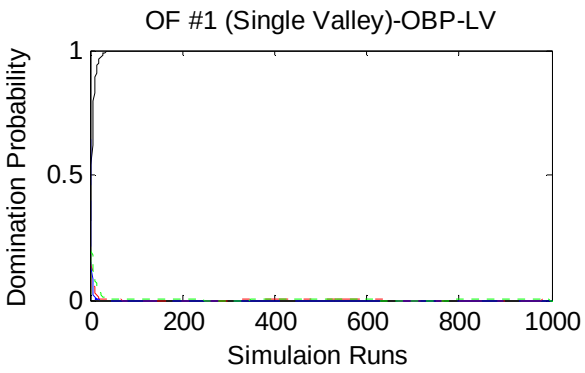
Figure 5.2: Plots of Best Function Values and Domination Probabilities in Experiment 1

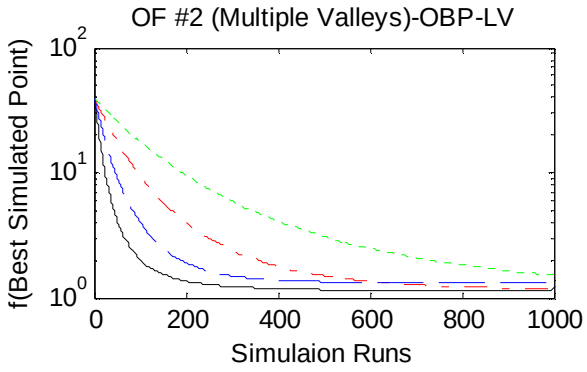
4.2. Experiment 2 with Overall Best Parameters

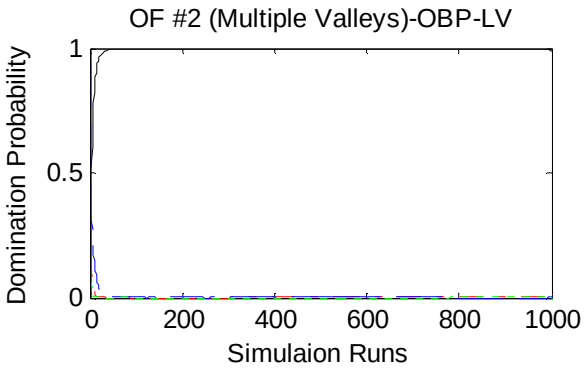
Figure 5.3 contains the two plots for each test bed in Experiment 1. The left-column plots show average of the expected value of the objective function of the best simulated points as more simulation budget is injected in the SO process, that is the values $\bar{x}^{(b)}_{(m,s^{*}_{m})}$. The right graphs show domination probabilities $\hat{p}^{(b)}_{(m,s^{*}_{m})}$. Our interpretations of these graphs are as follows:

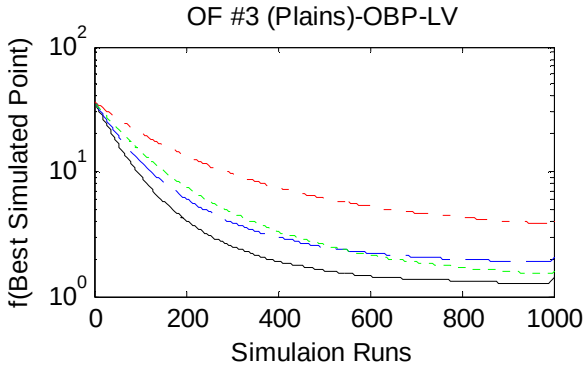
1. GA outperforms all other methods in Test Bed 5 after around 700 simulation runs and the hypothesis (5.1) is rejected. For less than around 700 runs in this test bed and for every other test bed, GR is the dominating method.
2. Even in the case of Test Bed 5 after 700 runs, the performance of GA and GR are slightly different.
3. When the parameters of all methods are not tuned specifically for a test bed, their performances are worse than the associated test beds in Experiment 1.
4. GR is less sensitive to its parametrs. It can have almost similar performance with different settings of its parameters.

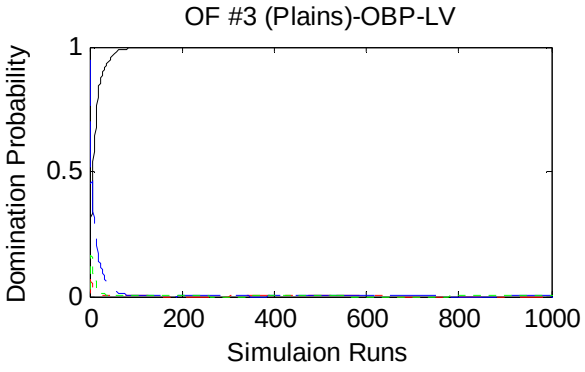

OF #1 (Single Valley)-OBP-LV
f(Best Simulated Point)
Simulaion Runs
GR
SA
MR
GA

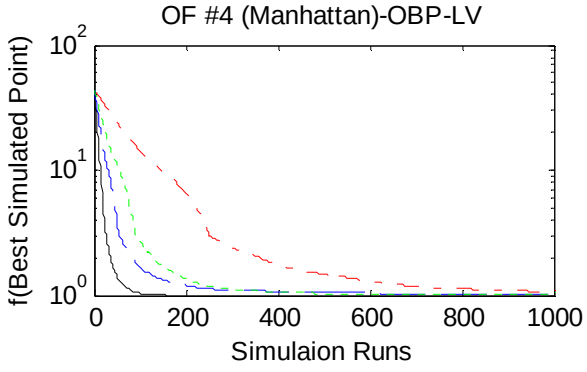

OF #1 (Single Valley)-OBP-LV
Domination Probability
Simulaion Runs

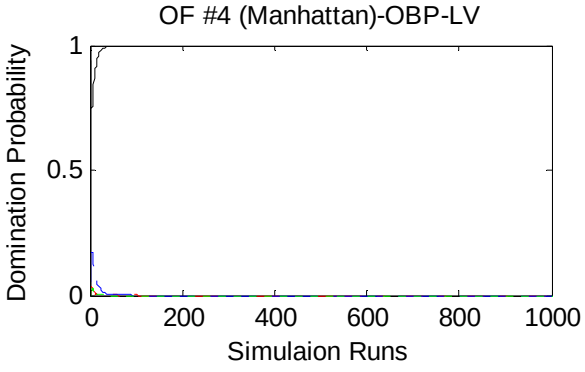

OF #2 (Multiple Valleys)-OBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #2 (Multiple Valleys)-OBP-LV
Domination Probability
Simulaion Runs


OF #3 (Plains)-OBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #3 (Plains)-OBP-LV
Domination Probability
Simulaion Runs


OF #4 (Manhattan)-OBP-LV
f(Best Simulated Point)
Simulaion Runs


OF #4 (Manhattan)-OBP-LV
Domination Probability
Simulaion Runs

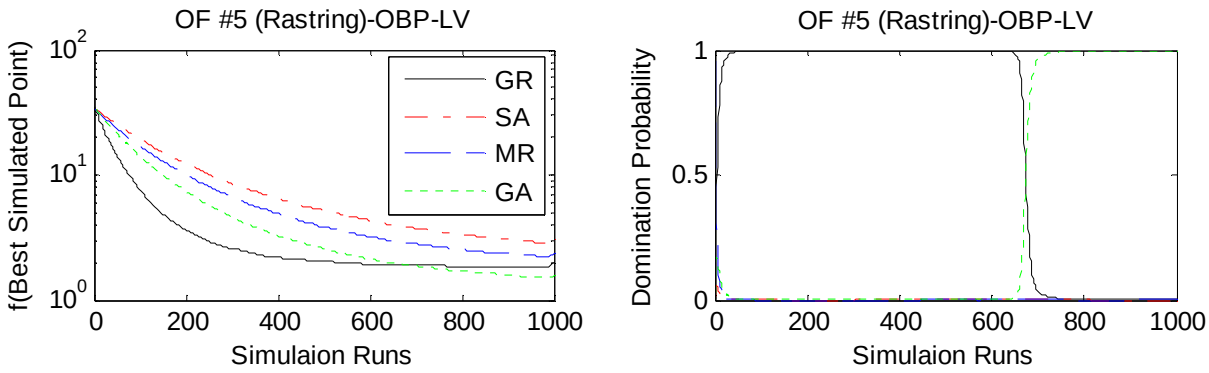


Figure 5.3: Plots of Best Function Values and Domination Probabilities in Experiment 2

5. Bottom-Line Remarks on Efficiency

If a user of SO methods has reason to believe that his/her SO problem is similar to Test Bed 1, then we would say that GR is not efficient. The user is recommended to tune the parameters of SA specifically for the problem and apply this method.

It is strange for us to believe that a user has correct information about the shape of his/her objective function when it looks like objective function 5. If the user is not reasonably sure that the objective function of his/her SO problem is similar to objective function 5, we strongly recommend using GR.

For test beds other than 1 and 5, GR is more efficient than other methods. We also observed that GR is not very sensitive to its parameters; when we used generic parameters in Experiment 2, almost similar performance was obtained for GR. Also, GR seems to be able to tackle high variance in the response surface of SO problems. Our intuition is that this is mostly done through metamodel used in GR. Finally, the performance of GR in higher dimensions is expected to be proportionally similar to that of lower dimensions.

Chapter 6: Conclusion

In this short chapter, concluding remarks about this dissertation are provided. First, the contributions of this research are summarized. We go over main findings and the techniques developed in this dissertation and discuss why Golden Region is a good method for simulation optimization. There are a number of shortcomings in this research which are presented in next part of this Chapter. Finally, directions for future research are discussed.

1. Contribution Summery

In this dissertation, we considered the problem of minimizing expected value of a random variable which is a function of random elements and controllable variables of a stochastic system of interest. We assumed the expected-value function is not known, but fixing the controllable settings, it can be estimated through an expensive input-output model such as discrete-event simulation.

The contributions of this PhD dissertation are the following:

1. Introducing Probabilistic Search methods as a large subclass of Random Search methods, merging them with Indifference Zone methods and proving their asymptotic convergence.
2. Introducing Golden Region method as a new Simulation Optimization method for continuous problems.
3. Proving the convergence of the Golden Region method to the set of global optimums.
4. Developing a framework for comparing the efficiency of Simulation Optimization methods.
5. Testing the efficiency of Golden Region method and comparing it with other methods.

The analysis of this dissertation starts with the definition of Random Search methods. In these iterative methods, a population of new points is selected from the feasible region based on a certain sampling strategy. Then these points are simulated and the algorithm goes to next iteration if no termination

condition holds. This class of optimization methods is broad enough to include many metaheuristic methods such as Genetic Algorithm, Simulated Annealing, Nested Partitions, etc.

In the next step, we narrowed the definition of Random Search methods to a subclass, called Probabilistic Search methods. In simple words, any Random Search method is also a Probabilistic Search method if two conditions hold: 1) the probability of sampling from any arbitrary part of the feasible region in any iteration of the method is positive and 2) this probability does not converge to zero.

We showed that if an Indifference Zone method is merged with a Probabilistic Search method, the resulting method which we call Hybrid Probabilistic Search method converges to the set of global optimum when a local neighborhood of a global optimum is continuous. In fact the Probabilistic Search method guarantees to visit any neighborhood of the global optimum and the Indifference Zone guarantees to recognize the best visited point as the optimum. These two procedures work together to help a Hybrid Probabilistic Search method converge to a global optimum asymptotically.

In the next step, we introduce a new Probabilistic Search method called Golden Region search. In fact the main contribution of the dissertation is development of Golden Region search method based on some ideas to improve the efficiency of simulation optimization methods. GR is primarily aimed at continuous optimization problems. The method combines different search strategies to explore the feasible region. The proposed method requires partitioning of the feasible region and uses a metamodel along with the information of the distribution and goodness of past simulation points to find the Golden Region containing the global optimum. We proposed a neural network framework as metamodel with structuring and training procedures.

In Chapter 5, we designed a hypothesis test to limit type one and two errors when two or more SO methods are compared. This test provides the number of samples required from each competing method in the experiments along with critical value of the test. We used closed-form functions instead of real simulation for our experiment mainly because we needed a diverse set of functions. In the first of two sets of experiment, the parameters of the competing methods were tuned specifically for each method. In most cases, GR revealed better performance than its competitors. In the second experiments, a single set of parameters were used for each method on all test problems in the experiment. As expected, the performances of all methods were worse than Experiment 1. However, GR again outperformed other methods in most cases. This showed that when little to nothing is known

about the shape of the objective function, GR is a robust method. In summary, Hybrid GR is a good SO method because:

1. There are not many convergent methods in the literature for continuous problems. Hybrid GR is provably convergent.
2. The computational results of Chapter 5 partially reveal the efficiency of the GR when compared to its competing methods.
3. The performance of GR is not very sensitive to many of its parameters; there are guidelines for selecting the other parameters of this method.
4. Most importantly, GR is designed based on the idea that valuable information of past simulation points must be used before proposing new points for simulation. There are not many methods in the literature which truly imply this principle.
5. GR has been designed to be efficient only when objective function is expensive. When a method is dual-intent, that is, it is designed to be efficient for cheap functions as well, we have strong doubts if it could be efficient for expensive functions. Optimizations of cheap and expensive functions are totally different philosophies which require specialized approaches.

In next section, some shortcomings of the analysis of this research are provided.

2. Shortcomings of This Research

No research work is perfect and this dissertation is obviously not an exception. When I was studying my Bachelor's program, I thought my PhD dissertation will change the world or win a prize comparable to Noble. I wanted to be the best in my research area. This turned out to be just a childish dream or like ambitions of historical Figures such as Alexander the great, Cyrus the great or Hitler.

If I had to redo my PhD program, I wouldn't have chosen SO as the area of my PhD dissertation. Honestly, more than half of the contribution of this research attributes to my Master's dissertation. So, the first shortcoming might be the lack of enough contribution on top of my previous research. Part of the reason I didn't choose another topic is because of the limited number of years our PhD program could finance my education.

The way that I did this dissertation follows the standard of a literature I myself am critical of. Our main argument is that an SO method should be efficient, but did I prove my method is efficient? or I just

added one more method to the set of available methods in the literature. In the experiments of Chapter 5, GR seems to be more efficient than some other methods, but this is not a guarantee that GR is the best. In the literature, the current standard is that a new method is proposed, its convergence is shown and its performance is compared with limited number of other methods in the literature using a couple of test problems. Pretty similar approach is used here. The standard that should be used in the literature and must have been used in my dissertation is the following: there must be a number of standardized SO test problems approved by simulation researchers. Every researcher who proposes a new method should publish a user-friendly code of the method in a standard programming language (there are tens of methods in the literature; a single researcher may not be able to code all the methods). The researcher must also provide guidelines for choosing the parameters of his/her proposed method. She/he must test the method with a standard experiment design and report standard indicators of efficiency of the newly proposed method. This way, efficient methods could be recognized. One potential research area is to work on such a standard and propose it formally to SO community. I think the framework I proposed in Chapter 5 is a good place to start.

The algorithm proposed for GR is not flawless. I sacrificed some of the principles of improving efficiency in order to simplify the algorithm. In fact, there are many more algorithmic details that I believe could be added to GR to improve its efficiency, but I didn't add because I wanted to keep the algorithm simple. For instance, recall that when Quality Score is the dominating criterion in an iteration, the new point is randomly selected using a multivariate normal distribution with mean fixed at the location of the best unsaturated point of the promising region. What if the new point falls very close to an already simulated point? This contradicts the principles mentioned in Chapter 1. Taking care of these kinds of details makes the algorithm more complex, but probably more efficient. This might be worthwhile in practice.

3. Future Works

A number of research directions remain open. The GR method can be extended to the problems with stochastic constraints (Kabirian and Olafsson 2007b and 2008). We have developed the main ideas of this extension, but still working on more experimental results.

Metamodeling can improve the efficiency of the SO methods in many ways (Barton and Meckesheimer 2006). We used a neural network approach with a specific updating procedure. The appropriateness of

other metamodel updating procedures as well as metamodel types deserves more investigation. We used batch training procedure, however incremental training might be more efficient for our method.

Optimal Computing Budget Allocation (Chen and Lin 2000) is one of the potential ranking and selection methods that could be merged with GR to form a Hybrid GR. The idea is to define an increasing simulation budget periodically to spend on “cleaning up” the already simulated points aiming at increasing the approximated probability of correct selection.

We believe that extreme values of decision variables are more likely to be the optimal solutions of real SO problems. It might be beneficial if SO methods first try the extreme values before their main optimization routines. Since many of the heuristic methods of SO including our proposed method in this paper document use an initial population of points, the extreme solutions might be included in these populations.

References

Abraham, A. (2004) Meta learning evolutionary artificial neural networks, Neurocomputing 56 (2004), pp. 1 – 38

Ahmed, M. A., T. M. Alkhamis (2002) Simulation-based optimization using simulated annealing with ranking and selection, Computers & Operations Research 29 (2002), pp. 387-402

Allen, T., L. Yu (2000) Low cost response surface methods for and from simulation optimization, Proceedings of the 2000 Winter Simulation Conference, pp. 704-714

Alrefaei, M. H., A. J. Alawneh (2004) Selecting the best stochastic system for large scale problems, DEDS Mathematics and Computers in Simulation 64 (2004) pp. 237–245

Andradottir, S. (1998a) Handbook of Simulation/edited by Jerry Banks. Chapter9: Simulation Optimization, John Wiley and Sons Inc.

Andradottir, S. (1998b) A review of simulation optimization techniques, Proceedings of the 1998 Winter Simulation Conference, 151-158.

Andradottir, S. (2006) An overview of simulation optimization via random search, In Henderson and Nelson (eds.), Handbooks in operations research and management science, 13, 617 – 632, Elsevier.

Andradottir, S., D. Goldsman, S-H Kim (2005) Finding the best in the presence of a stochastic constraint, Proceedings of the 2005 Winter Simulation Conference, 732-738

April, J. et. al. (2004) New advances and applications for marrying simulation and optimization, Proceedings of the 2004 Winter Simulation Conference, pp. 80-86.

April, J. et. al. (2001) Simulation/optimization using real-world applications, Proceedings of the 2001 Winter Simulation Conference. pp. 134-138

April, J. et. al. (2003) Practical introduction to simulation optimization, Proceedings of the 2003 Winter Simulation Conference, pp. 71-78

April, J. M., M. Better, F. Glover, J. Kelly, M. Laguna (2006) Enhancing business process management with simulation optimization, Proceedings of the 2006 Winter Simulation Conference, 642-649

Azadivar, F. (1999) Simulation optimization methodologies, Proceedings of the 1999 Winter Simulation Conference. 93-100.

Azadivar, F., G. Tompkins (1999) Simulation optimization with qualitative variables and structural model changes: a genetic algorithm approach, European Journal of Operations Research 113 (1999), pp. 169-182

Banks J. (1998) Handbook of simulation/edited by jerry banks. chapter1: principles of simulation, John Wiley and Sons Inc.

Barton, R. R., M. Meckesheimer (2006) Metamodel-based simulation optimization, In Henderson and Nelson (eds.), Handbooks in operations research and management science, 13, 617 – 632, Elsevier.

Bayraksan, G., D. P. Morton (2006) Assessing solution quality in stochastic programs, Mathematical Programming, Ser. B 108, 495–514

Bechhofer, R. E. (1954) A single-sample multiple decision procedure for ranking means of normal populations with known variances, Annals of Mathematical Statistics 25:16-39.

Carson J. S. (2004) Introduction to modeling and simulation, Proceedings of the 2004 Winter Simulation Conference

Carson, Y., A. Maria (1997) Simulation optimization: methods and applications, Proceedings of the 1997 Winter Simulation Conference. pp. 118- 126

Chang, Y. P., W.-T. Huang (2001) Generalized subset selection procedures under heteroscedasticity, Journal of Statistical Planning and Inference 98 (2001) pp. 239–258

Chen, E. J. , W. D. Kelton (2003) Inferences from indifference-zone selection procedures, Proceedings of the 2003 Winter Simulation Conference, pp. 456-464

Chen, E. J., W. D. Kelton (2000) An enhanced two-stage selection procedure, Proceedings of the 2000 Winter Simulation Conference pp. 727–735.

Chen, E. J., W. D. Kelton (2005) Sequential selection procedures: using sample means to improve efficiency, European Journal of Operational Research 166 (2005) pp. 133–153

Chen, H. C. et. al. (1997) New development of optimal computing budget allocation for discrete event simulation, Proceedings of the 1997 Winter Simulation Conference

Chen, H. C. et. al. (1998) Computing budget allocation for simulation experiments with different system structures, Proceedings of the 1998 Winter Simulation Conference, pp. 735-741

Chen, H. C. et. al. (2000) Simulation budget allocation for further enhancing the efficiency of ordinal optimization, Journal of DEDS, V. 10, #3, July 2000, pp 251-270

Chen, N., et. al. (2005) Sufficient and necessary condition for the convergence stochastic approximation algorithms, Statistics & Probability Letters, Article in Press, www.sciencedirect.com

Cheng, R. C. H. (1999) Regression metamodelling in simulation using bayesian methods, Proceedings of the 1999 Winter Simulation Conference, pp. 330-335

Chick, S. E. (2004) Bayesian methods for discrete event simulation, Proceedings of the 2004 Winter Simulation Conference, pp. 89-100

Chick, S. E., K. Inoue (2000) New results on procedures that select the best system using CRN, Proceedings of the 2000 Winter Simulation Conference, pp. 554-561

Chick, S. E., K. Inoue (1999) A decision theoretic approach to screening and selection with common random numbers, Proceedings of the 1999 Winter Simulation Conference, pp. 603-610

Chick, S.E., K. Inoue (2001a) New two-stage and sequential procedures for selecting the best simulated system, Operations Research, 49(5): 732-743.

Chick, S.E., K. Inoue (2001b) New procedures for identifying the best simulated system using common random numbers, Management Science, 47(8): 1133–1149.

Chong, E. K. P. , P. J. Ramadge (1992) Convergence of recursive algorithms using IPA derivative estimates, Discrete Event Dynamic Systems, Vol. 2, pp. 339-372

Chong, E. K. P., P. J. Ramadge (1993) Optimization of queues using an infinitesimal perturbation analysis-based stochastic algorithm with general update times, SIAM Journal on Control and Optimization

Dengiz, B., C. Alabas (2000) Simulation optimization using tabu search, Proceedings of the 2000 Winter Simulation Conference, pp. 805-810

Montgomery D. C., R. H. Myers (2002) Response surface methodology: process and product optimization using designed experiments, Wiley Series in Probability and Statisics

Dudewicz, E. J., S. R. Dalal (1975) Allocation of observations in ranking and selection with unequal variances, Sankhyii, B37, 28-78.

Eberhart, R. C., J. Kennedy (1995) A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine And Human Science, IEEE service center. Piscataway, NJ, Nagoya, Japan, pp. 39–43.

Fausett, L. (1994) Fundamentals of neural networks; Prentice Hall International, Inc

Ferris, M. C., T. S. Munson, K. Sinapiromsaran (2000) A practical approach to sample-path simulation optimization, Proceedings of the 2000 Winter Simulation Conference

Fu, M. C. (2001) Simulation optimization, Proceedings of the 2001 Winter Simulation Conference. pp. 53-61

Fu, M. C. (2002) Optimization for simulation: theory vs. practice, INFORMS Journal on Computing, Vol. 14, No. 3, Summer 2002, pp. 192-215

Fu, M. C. (2006) Gradient estimation, In Henderson and Nelson (eds.), Handbook in operations research and management science, 13, 575-616, Elsevier.

Fu, M. C., C-H. Chen, L. Shi, (2008) Some topics for simulation optimization, Proceedings of the 2008 Winter Simulation Conference, S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds., 27-38

Fu, M. C., F. W. Glover, J. April (2005) Simulation optimization: a review, new developments, and applications, In Proceedings of the 2005 Winter Simulation Conference. 83-95

Fu, M. C., S. Andradottir, J. S. Carson, F. Glover, C. R. Harrell, Y. C. Ho, J. P. Kelly, S. M. Robinson (2000). Panel session-integrating optimization and simulation: research and practice, Proceedings of the 2000 Winter Simulation Conference. pp. 610-616

Futschik, A., G. Pflug (1995) Confidence sets for discrete stochastic optimization. Annals of Operations Research 56(1995), pp. 95-108

Gerencsér, L., et. al. (1999), Optimization over discrete sets via SPSA, Proceedings of the IEEE Conference on Decision and Control, 7-10 December 1999, Phoenix, AZ, pp. 1791-1795

Gerencsér, L., et. al. (2001) Discrete optimization via SPSA. Proceedings of the American Control Conference, 25-27 June 2001, Arlington, VA, pp. 1503-1504

Gerencsér, L., et. al. (2004) Discrete optimization, SPSA, and Markov chain Monte Carlo methods. Proceedings of the American Control Conference, 29 June–2 July 2004, Boston, MA, pp. 3814–3819

Ghosh, A. (2009) Optimal prices and production rate in a closed loop supply chain with re-manufacturing under heavy traffic, In graduate seminar series of Industrial and Manufacturing Systems Department at Iowa State University

Glover F. (1997) A template for scatter search and path relinking; Lecture Notes in Computer Science 1997 / edited by Hao J. K. , E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, 1363: 1-53.

Glover, F. (1989) Tabu search – part I, ORSA Journal on Computing 1: 190-206.

Glover, F. (1990). Tabu search –part II, ORSA Journal on Computing 2: 4-32. Goldsman, D., B. L. Nelson. (1998). Comparing systems via simulation, In Banks (edited by), Handbooks of simulation, chapter 13, John Wiley and Sons.

Glover, F., M. Laguna (1997) Tabu search, Kluwer Academic Publishers

Glover, F. et. al. (1999) New advances for wedding optimization and simulation, Proceedings of the 1999 Winter Simulation Conference. pp. 255-260

Glover, F., M. Laguna, R. Martí (2003) Advances in evolutionary computation: theory and applications, / edited by Ghosh, A., and S. Tsutsui; Springer-Verlag, New York, pp. 519-537 (2003)

Glover, F. (1990) Tabu search: a tutorial, Interfaces, 20(4), pp. 74-94

Glover, F. (1999) Scatter search and path relinking; new ideas in optimization / edited by D. Corne, M. Dorigo and F. Glover, McGraw Hill, 1999, Chapter 19

Glynn, P. W. (1987) Likelihood ratio gradient estimation: an overview, Proceedings of the 1987 Winter Simulation conference, 366-375.

Goldberg, D. E. (1989) Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Publishing Co. Inc.

Goldsman, D., B. L. Nelson (1998a) Handbook of simulation/edited by Jerry Banks. Chapter8: Comparing Systems via Simulation, John Wiley and Sons Inc.

Goldsman, D., B. L. Nelson (1998b) Statistical screening, selection, and multiple comparison procedures in computer simulation, Proceedings of the 1998 Winter Simulation Conference, pp. 159-166

Goldsman, D., B. L. Nelson (2001) Statistical selection of the best, Proceedings of the 2001 Winter Simulation Conference, pp. 139-146

Goldsman, D. et. al. (2000) Ranking and selection for steady-state simulation, Proceedings of the 2000 Winter Simulation Conference pp. 544-553

Gosavi A. (2003). Simulation-based optimization: parametric optimization techniques and reinforcement learning, Kluwer Academic Publishers

Guikema, S. D., R. A. Davidson, Z. Çagnan (2004) Efficient simulation-based discrete optimization, Proceedings of the 2004 Winter Simulation Conference, pp. 536-544

Gupta, S. S., K. J. Miescke (2002) On the Performance of subset selection rules under normality, Journal of Statistical Planning and Inference 103 (2002) pp. 101–115

Gupta, S. S. (1956a) On a decision rule for a problem in ranking means, Doctoral dissertation, Institute of Statistics, Univ. of North Carolina, Chapel Hill, NC.

Gupta, S. S. (1965b) On some multiple decision (ranking and selection) rules, Technometrics 7:225{245. Gurken, G., A.Y Ozge, and S. M. Robinson. 1994. Sample path optimization in simulation. Proceedings of 1994 Winter Simulation Conference, ed. J.D. Tew, S. Manivannan, D.A. Sadowski, and A.F. Seila, 247-254.

Gurkan, G., A. Y. Ozge, S. M. Robinson (1999) Solving stochastic optimization problems with stochastic constraints: an application in network design, Proceedings of the 1999 Winter Simulation Conference

Haupt, R. L., S. E. Haupt (2004) Practical genetic algorithms; Second Edition, A John Wiley & Sons, Inc., Publication 2004

Henderson and Nelson (2006) Handbook in operations research and management science, 13, Elsevier

Herrera, F., M. Lozano, D. Molina (2006) Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies, European Journal of Operational Research Volume 169, Issue 2 , pp. 450-476

Hillier, F. S., G. J. Liberman (2001) Introduction to operations research, McGraw-Hill

Hochberg, Y., A. C. Tamhane (1987) Multiple comparison procedures, Wiley, New York

Holland, J. H. (1992) Genetic algorithms, Scientific American, July, pp. 66-72.

Hu, J., M C. Fu, S I. Marcus (2005) Stochastic optimization using model reference adaptive search, Proceedings of the 2005 Winter Simulation Conference, 811-818

Hu, J., M.C. Fu, S.I. Marcus (2007) A model reference adaptive search algorithm for global optimization, Operations Research, 55, 549-568.

Humphrey, D. G., J. R. Wilson (1998) A revised simplex search procedure for stochastic response-surface optimization, Proceedings of the 1998 Winter Simulation Conference, pp. 751-759

Jones, M. H., K. P. White (2004) Stochastic approximation with simulated annealing as an approach to global discrete-event simulation optimization, Proceedings of the 2004 Winter Simulation Conference, 500-507

Kabirian, A. (2006) A review of current methodologies and developing a new heuristic method of simulation optimization, Master of Science Dissertation, Jan. 2006, Sharif University of Technology, Tehran, Iran

Kabirian, A. (2009) Hybrid Probabilistic search methods for simulation optimization, Journal of Industrial and Systems Engineering, www.jise.info, Vol. 2, No. 4, pp 259-270

Kabirian, A., M. R. Hemmati (2007) A strategic planning model for natural gas transmission networks, Energy Policy, Vol. 35, Issue 11, November 2007, pp 5656-5670, Elsevier.

Kabirian, A., S. Olafsson (2007a) Allocation of simulation runs for simulation optimization, Proceedings of the 2007 Winter Simulation Conference & PhD Colloquium & Poster Session. 363-371

Kabirian, A., S. Olafsson (2007b) Multiresponse golden region search under noisy constraints, working paper, Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA, USA

Kabirian, A., S. Olafsson (2007c). Simulation based optimization via golden region search, Under review

Kabirian, A., S. Olafsson (2008). Selection of the best with multiple stochastic constraints, Working Paper, Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA, USA

Kao, C., S.-P. Chen (2005) A stochastic quasi-newton method for simulation response optimization; European Journal of Operations Research, Article in Press, www.sciencedirect.com

Keller, R. (1987) Expert system technology: development and application; Yourdon Press

Kermani, B. G., S. S. Schiffman, H. T. Nagle (2005) Performance of the levenberg–marquardt neural network training method in electronic nose applications, Sensors and Actuators B 110 (2005) pp. 13–22

Keys, A. C., L. P. Rees (2004) A sequential-design metamodeling strategy for simulation optimization, Computers & Operations Research 31 (2004), pp. 1911–1932

Kiefer J., J. Wolfoxitz (1952) Stochastic estimation of the maximum of a regression function, Annals of Mathematical Statistics. vol. 23, pp. 462-466

Kim, S, B. L. Nelson (2006) Selecting the best system. In Henderson and Nelson (eds.), Handbooks in operations research and management science, 13, 501-534, Elsevier.

Kim, S. (2006) Gradient-based simulation optimization, Proceedings of the 2006 Winter Simulation Conference, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds., 159-167

Kim, S. (2006) Gradient-Based Simulation Optimization, Proceedings of the 2006 Winter Simulation Conference, 159-167

Kim, S.-H., B. L. Nelson (2001) A fully sequential procedure for indifference zone selection in simulation, ACM TOMACS 11:251-273.

Kim, S-H., B. L. Nelson (2007) Recent advances in ranking and selection, Proceedings of the 2007 Winter Simulation Conference, S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds., 162-172

Kirkpatrick, S., C. D. Gelatt, and J. M. P. Vecchi (1983) Optimization by simulated annealing, Science, 13 May 1983, Volume 220, Number 4598

Kleijnen, J. P. C. (1998) Handbook of simulation/edited by Jerry Banks. Chapter6: Experimental Design for Sensitivity Analysis, Optimization and Validation of Simulation Models, John Wiley and Sons Inc.

Kumar, U. A. (2005) Comparison of neural networks and regression analysis: A new insight, Expert Systems with Applications 29 (2005), pp. 424–430

Laguna, M., and R. Marti (2002) Neural network prediction in a system for optimizing simulations, IEE Transactions (2002) 34, pp. 273-282

Lee, L. H. et. al. (2004) Optimal computing budget allocation for multi-objective simulation models, Proceedings of the 2004 Winter Simulation Conference. pp. 586-594

Li, D. et. al. (2002) Constraint ordinal optimization, Information Sciences 148 (2002) pp. 201–220

Luenberger, D. J. (1984) Linear and nonlinear programming, Addison-Wiley, Second Edition

Luersen, M. A., R. L. Riche (2004) Globalized Nelder-Mead method for engineering optimization, Computers and Structures 82 (2004), pp. 2251-2260

Ma, L., K. Khorasani (2003) A new strategy for adaptively constructing multilayer feedforward neural networks, Neurocomputing 51 (2003), pp. 361 – 385

Magoulas, G. D. et. al. (2002) Global search strategies for simulation optimization, Proceedings of the 2002 Winter Simulation Conference

Miller, R. G. (1981) Simultanous statistical inference, 2nd Ed., Springer-Verlag, New York

Morrice, D. J. et. al. (1998) An approach to ranking and selection for multiple performance measures Proceedings of the 1998 Winter Simulation Conference pp. 719-725

Morrice, D. J. et. al. (1999) Sensitivity analysis in ranking and selection for multiple performance measures, Proceedings of the 1999 Winter Simulation Conference pp. 618-624

Nasereddin, M., M. Mollaghasemi (1999) The development of a methodology for the use of neural networks and simulation modeling in system design, Proceedings of the 1999 Winter Simulation Conference, pp. 537-542

Neddermeijer, H. G., G. J. van Oortmarssen, N. Piersma, R. Dekker (2000) A framework for response surface methodology for simulation optimization, Proceedings of the 2000 Winter Simulation Conference, pp. 129-136

Nelder, J. A, R. Mead (1965) A simplex method for function minimization, Computer Journal, 7: 308-313

Nicolai, R. P., R. Dekker, N. Piersma, G. J. van Oortmarssen (2004) Automated response surface methodology for stochastic optimization models with unknown variance, Proceedings of the 2004 Winter Simulation Conference, pp. 491-499

Nocedal, J., S. J. Wright. (1999) Numerical optimization, Springer-Verlag New York, Inc.

Olafsson, S., J. Kim (2002) Simulation optimization, Proceedings of the 2002 Winter Simulation Conference, pp. 1931-1936

Olafsson, S. (2004) Two stage nested partitions method for stochastic optimization, Methodology and Computing in Applied probability 6, 5-27

Olafsson, S. (2006) Metaheuristics. In Henderson and Nelson (eds.), Handbook in operations research and management science, 13, 633 –654, Elsevier.

Olafsson, S., N. Gopinath (2000) Optimal selection probability in the two-stage nested partitions method for simulation-based optimization, Proceedings of the 2000 Winter Simulation Conference, pp. 736-742

Olafsson, S., J. Kim (2002) Simulation optimization. In Proceedings of the 2002 Winter Simulation Conference, 1931-1936

Pasandideh, S. H. R., S. T. A. Niaki (2005) Multi-response simulation optimization using genetic algorithm within desirability function framework; Applied Mathematics and Computation

Paul J. P., T. S. Chanev (1998) Simulation optimization using a genetic algorithm, Simulation Practice and Theory 6, pp. 601-611

Pichitlamken, J., B. L. Nelson. (2001) Selection of the best procedures for optimization via simulation, Proceedings of the 2001 Winter Simulation Conference, pp. 401-407

Pichitlamken, J. et. al. (2005) A sequential procedure for neighborhood selection-of-the-best in optimization via simulation, European Journal of Operational Research, Article in Press, www.sciencedirect.com

Pichitlamken, J., B. L. Nelson. (2002) A combined procedure for optimization via simulation, Proceedings of the 2002 Winter Simulation Conference, pp. 292-300

Pierreval, H., J. L. Paris (2003) From simulation optimization to simulation configuration of systems, Simulation modeling Practice and Theory 11 (2003), pp. 5-19

Riche, R. L. et. al. (2003) An object oriented simulation optimization interface, Computers and Structures 81 (2003), pp. 1689-1701

Rinott, Y. (1978) On two-stage selection procedures and related probability-inequalities, Communication in Statistics: Theory and Methods, A7(8), 799-811.

Robins, H., S. Monro. (1951). A stochastic approximation method, Annals of Mathematical Statistics vol. 22, pp. 400-407, 1951

Rosen, S. L., C. M. Harmonosky (2005) An improved simulated annealing simulation optimization method for discrete parameter stochastic systems, Computers & Operations Research, Volume 32, Issue 2, February 2005, pp. 343-358

Safizadeh, M. H., R. Signorile (1994) Optimization of simulation via quasi-newton methods, ORSA Journal on Computing 6, pp. 398-408

Schruben, L. W., V. J. Cogliano (1991) Simulation sensitivity analysis: a frequency domain approach, Proceedings of the 1991 Winter Simulation Conference, 455-459.

Shapiro, A. (1996) Simulation based optimization, Proceedings of the 1996 Winter Simulation Conference, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J.Swain, 332-336.

Shi, L., C. H. Chen, E. Yücesan (1999) Simultaneous simulation experiments and nested partition for discrete resource allocation in supply chain management, Proceedings of the 1999 Winter Simulation Conference

Shi, L., S. Olafsson (2000a) Nested partitions method for global optimization, Operations Research 48, 390-407.

Shi, L., S. Olafsson (2000b) Nested partitioned method for stochastic optimization, Methodology and Computing in Applied probability 2, 271-291.

Spall, J. C. (1992) Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Transactions on Automatic Control, Vol. 37, No. 3, March 1992

Spall, J. C. (1997) A one-measurement form of simultaneous perturbation stochastic approximation, Automatica, Vol. 33, No. 1, pp. 109-112.

Spall, J. C. (1998) An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins APL Technical Digest, Volume 19, Number 4 (1998), pp. 482-492

Swisher, J. R., S. H. Jacobsen (1999) A survey of ranking, selection, and multiple comparison procedures for discrete- event simulation, Proceedings of the 1999 Winter Simulation Conference. pp. 492-501

Swisher, J. R. et. al. (2000) A survey of simulation optimization techniques and procedures, Proceedings of the 2000 Winter Simulation Conference. pp. 119-128

Taha, H. A. (1997) Operations research: an introduction. Prentice Hall, NJ.

Tunali, S., I. Batmaz; (2003) A metamodeling methodology involving both qualitative and quantitative input factors, European Journal of Operational Research 150 (2003), pp. 437–450

Wang, L. (2005) A hybrid genetic algorithm–neural network strategy for simulation optimization, Applied Mathematics and Computation 170 (2005), pp. 1329–1343

Appendix : Objective Functions

Objective Function 1 (Single Valley):

```
[x y]=meshgrid(0:2:100);
rand1=.5;rand2=.5;

xmin=0;xmax=100;ymin=0;ymax=100;ooptx=50;oopty=50;optx=100*rand1;opty=100*rand2;
bestf=-(1/(20*20*sqrt(2*pi)));worstf=0;
J=(((-(1/(20*20*sqrt(2*pi))).*exp((-.5/400).*((((xmax-xmin)/100)*(x+ooptx-optx)+xmin-
50).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin-50).^2)))-bestf)/(worstf-bestf))*99+1; %1-
monomodal
x=x/10; y=y/10;
%'Position',[0.10 0.10 0.4 0.6],...
figure1 = figure;
axes1 = axes(...
  'XTick',[0 5 10],...
  'YTick',[0 5 10],...
  'ZTick',[0 20 40 60 80 100],...
  'Parent',figure1);
title(axes1,'OF #1 (Single Valley)-Original');
xlabel(axes1,'x');
ylabel(axes1,'y');
zlabel(axes1,'f');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
meshc(x,y,J,'Parent',axes1)
```

Objective Function 2 (Multiple Valleys):

```
[x y]=meshgrid(0:2:100);
rand1=.5;rand2=.5;

xmin=-2.5;xmax=3;ymin=-2.8;ymax=2.8;bestf=-
6.5511333327650165;worstf=8.1062135891752;ooptx=54.56559;oopty=20.97255;optx=ooptx;
opty=oopty;
J=(((3.*(1-(((xmax-xmin)/100)*(x*(ooptx/optx))+xmin) ).^2.*exp(-((((xmax-
xmin)/100)*(x*(ooptx/optx))+xmin) .^2)-((((ymax-ymin)/100)*(((100-oopty)./(100-
opty)).*y+100.*((oopty-opty)./(100-opty)))+ymin) +1).^2)-10.*((((xmax-
xmin)/100)*(x*(ooptx/optx))+xmin)./5-(((xmax-xmin)/100)*(x*(ooptx/optx))+xmin).^3-(((ymax-
ymin)/100)*(((100-oopty)./(100-opty)).*y+100.*((oopty-opty)./(100-opty)))+ymin).^5).*exp(-
(((xmax-xmin)/100)*(x*(ooptx/optx))+xmin) .^2-(((ymax-ymin)/100)*(((100-oopty)./(100-
opty)).*y+100.*((oopty-opty)./(100-opty)))+ymin) .^2)-1./3.*exp(-((((xmax-
xmin)/100)*(x*(ooptx/optx))+xmin)+1).^2-(((ymax-ymin)/100)*(((100-oopty)./(100-
opty)).*y+100.*((oopty-opty)./(100-opty)))+ymin).^2))-bestf)/(worstf-bestf))*99+1;
```

```
x=x/10; y=y/10;
%'Position',[0.10 0.10 0.4 0.6],...
figure1 = figure;
axes1 = axes(...
  'XTick',[0 5 10],...
  'YTick',[0 5 10],...
  'ZTick',[0 20 40 60 80 100],...
  'Parent',figure1);
title(axes1,'OF #2 (Multiple Valleys)-Original');
xlabel(axes1,'x');
ylabel(axes1,'y');
zlabel(axes1,'f');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
meshc(x,y,J,'Parent',axes1)

Objective Function 3 (Plains):
[x y]=meshgrid(0:3:100);
rand1=.3;rand2=.5;

xmin=-1;ymin=-1;xmax=1;ymax=1;
bestf=-1;ooptx=50;oopty=50;optx=rand1*100;opty=rand2*100;
[v w]=meshgrid(0:100:100);
for i=1:4
   if (((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)>=0 & (((ymax-ymin)/100)*(w(i)+oopty-
opty)+ymin)>=0
      wf(i)=1-(((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)-(((ymax-ymin)/100)*(w(i)+oopty-
opty)+ymin);
   elseif (((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)>=0 & (((ymax-ymin)/100)*(w(i)+oopty-
opty)+ymin)<=0
      wf(i)=1-(((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)+(((ymax-ymin)/100)*(w(i)+oopty-
opty)+ymin);
   elseif (((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)<=-.5
      wf(i)=(((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)+1;
   else
      wf(i)=-3*(((xmax-xmin)/100)*(v(i)+ooptx-optx)+xmin)-1;
   end
end
worstf=min(wf);worstf=max([-worstf 1]);
for i=1:length(x)
   for j=1:length(y)
      if (((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)>=0 & (((ymax-ymin)/100)*(y(i,j)+oopty-
opty)+ymin)>=0
```

```
        J(i,j)=1-(((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)-(((ymax-ymin)/100)*(y(i,j)+oopty-
opty)+ymin);
      elseif (((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)>=0 & (((ymax-ymin)/100)*(y(i,j)+oopty-
opty)+ymin)<=0
        J(i,j)=1-(((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)+(((ymax-ymin)/100)*(y(i,j)+oopty-
opty)+ymin);
      elseif (((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)<=-.5
        J(i,j)=(((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)+1;
      else
        J(i,j)=-3*(((xmax-xmin)/100)*(x(i,j)+ooptx-optx)+xmin)-1;
      end
   end
end
J=((-J -bestf)/(worstf-bestf))*99+1;
x=x/10; y=y/10;
%'Position',[0.10 0.10 0.4 0.6],...
figure1 = figure;
axes1 = axes(...
  'XTick',[0 5 10],...
  'YTick',[0 5 10],...
  'ZTick',[0 20 40 60 80 100],...
  'Parent',figure1);
title(axes1,'OF #3 (Plains)-Original');
xlabel(axes1,'x');
ylabel(axes1,'y');
zlabel(axes1,'f');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
meshc(x,y,J,'Parent',axes1)
```

Objective Function 4 (Manhattan):

```
[x y]=meshgrid(0:1.5:100);
r= [0.4235   0.3798   0.3050   0.6435   0.2126   0.0439   0.0158   0.4544   0.4508   0.2393;
   0.5155   0.7833   0.8744   0.3200   0.8392   0.0272   0.0164   0.4418   0.7159   0.0498;
   0.3340   0.6808   0.0150   0.9601   0.6288   0.3127   0.1901   0.3533   0.8928   0.0784;
   0.4329   0.4611   0.7680   0.7266   0.1338   0.0129   0.5869   0.1536   0.2731   0.6408;
   0.2259   0.5678   0.9708   0.4120   0.2071   0.3840   0.0576   0.6756   0.2548   0.1909;
   0.5798   0.7942   0.9901   0.7446   0.6072   0.6831   0.3676   0.6992   0.8656   0.8439;
   0.7604   0.0592   0.7889   0.2679   0.6299   0.0928   0.6315   0.7275   0.2324   0.1739;
   0.5298   0.6029   0.4387   0.4399   0.3705   0.0353   0.7176   0.4784   0.8049   0.1708;
   0.6405   0.0503   0.4983   0.9334   0.5751   0.6124   0.6927   0.5548   0.9084   0.9943;
   0.2091   0.4154   0.2140   0.6833   0.4514   0.6085   0.0841   0.1210   0.2319   0.4398];
```

```
optx=floor(100*rand)+1;optx=min([100 optx]);opty=floor(100*rand)+1;opty=min([100 opty]);
for i=1:length(x)
   for j=1:length(y)
      v=floor(x(i,j)/10)+1;v=min([10 v]);w=floor(y(i,j)/10)+1;w=min([10 w]);
      if 10*(v-1)+w==optx
         J(i,j)=1;
      elseif 10*(v-1)+w==opty
         J(i,j)=100;
      else
         J(i,j)=1+99*r(v,w);
      end
   end
end
x=x/10; y=y/10;
%'Position',[0.10 0.10 0.4 0.6],...
figure1 = figure;
axes1 = axes(...
  'XTick',[0 5 10],...
  'YTick',[0 5 10],...
  'ZTick',[0 20 40 60 80 100],...
  'Parent',figure1);
title(axes1,'OF #4 (Manhattan)-Original');
xlabel(axes1,'x');
ylabel(axes1,'y');
zlabel(axes1,'f');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
meshc(x,y,J,'Parent',axes1)
```

Objective Function 5 (Rastring):

```
clear all
clc

rand1=.5;rand2=.5;
xmin=-5;xmax=5;ymin=-
5;ymax=5;ooptx=50;oopty=50;optx=100*rand1;opty=100*rand2;bestf=-1;
x=0;y=0;wf(1)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-
opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-
ymin)/100)*(y+oopty-opty)+ymin)));
x=0;y=100;wf(2)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-
ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
```

```
x=100;y=0;wf(3)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-
ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=100;y=100;wf(4)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-
ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=optx-5-10*floor((optx-5)/10);y=opty-10*floor(opty/10);wf(5)=20+(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-
10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-
ymin)/100)*(y+oopty-opty)+ymin)));
x=optx-5-10*floor((optx-5)/10);y=opty+10*floor((100-opty)/10);wf(6)=20+(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-
10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-
ymin)/100)*(y+oopty-opty)+ymin)));
x=optx+5+10*floor((100-optx-5)/10);y=opty-10*floor(opty/10);wf(7)=20+(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-
10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-
ymin)/100)*(y+oopty-opty)+ymin)));
x=optx+5+10*floor((100-optx-5)/10);y=opty+10*floor((100-opty)/10);wf(8)=20+(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-
10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-
ymin)/100)*(y+oopty-opty)+ymin)));

x=0;y=opty-10*floor(opty/10);wf(9)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-
ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=0;y=opty+10*floor((100-opty)/10);wf(10)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=100;y=opty-10*floor(opty/10);wf(11)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=100;y=opty+10*floor((100-opty)/10);wf(12)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=optx-5-10*floor((optx-5)/10);y=0;wf(13)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=optx-5-10*floor((optx-5)/10);y=100;wf(14)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
x=optx+5+10*floor((100-optx-5)/10);y=0;wf(15)=20+(((xmax-xmin)/100)*(x+ooptx-
optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-
xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));
```

```
x=optx+5+10*floor((100-optx-5)/10);y=100;wf(16)=20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin)));

worstf=max(wf);
[x,y]=meshgrid(0:.5:100);


J=(((20+(((xmax-xmin)/100)*(x+ooptx-optx)+xmin).^2+(((ymax-ymin)/100)*(y+oopty-opty)+ymin).^2-10.*(cos(2.*pi.*(((xmax-xmin)/100)*(x+ooptx-optx)+xmin))-cos(2*pi.*(((ymax-ymin)/100)*(y+oopty-opty)+ymin))))-bestf)/(worstf-bestf))*99+1;

x=x/10; y=y/10;
figure1 = figure;
axes1 = axes(...
  'XTick',[0 5 10],...
  'YTick',[0 5 10],...
  'ZTick',[0 20 40 60 80 100],...
  'Parent',figure1);

title(axes1,'OF #5 (Rastring)-Original');
xlabel(axes1,'x');
ylabel(axes1,'y');
zlabel(axes1,'f');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
meshc(x,y,J,'Parent',axes1)
```