Methods to Compare Expensive Stochastic Optimization Algorithms with Application to Road Design

by

Shangwei Xie

B.A. Hons., The University of Nottingham, 2012
M.Q.F., Rutgers University, 2014

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

MASTER OF SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

June 2017

The undersigned certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis entitled: Methods to Compare Expensive Stochastic Optimization Algorithms with Application to Road Design submitted by Shangwei Xie in partial fulfilment of the requirements of the degree of Master of Science

Dr. Warren Hare, Irving K. Barber School of Arts and Sciences

Supervisor

Dr. Jason Loeppky, Irving K. Barber School of Arts and Sciences

Co-supervisor

Dr. Yves Lucet, Irving K. Barber School of Arts and Sciences

Supervisory Committee Member

Dr. Ross Hickey, Irving K. Barber School of Arts and Sciences

University Examiner

June 2, 2017

(Date Submitted to Grad Studies)

Abstract

Analyzing test data of stochastic optimization algorithms under random restarts is challenging. The data needs to be resampled to estimate the behavior of the incumbent solution during the optimization process. The estimation error needs to be understood in order to make reasonable inference on the actual behavior of the incumbent solution. Comparing the performance of different algorithms based on proper interpretation of the estimator is also very important. We model the incumbent solution of the optimization problem over time as a stochastic process and design an estimator of it based on bootstrapping from test data. Some asymptotic properties of the estimator and its bias are shown. The estimator is then validated by an out-of-sample test. Three methods for comparing the performance of different algorithms based on the estimator are proposed and demonstrated with data from a road design optimization problem.

Preface

The identification and design of the program was a joint work of the author and his supervisors. The author was responsible for the design, theoretical analysis, numerical validation and illustration of the model developed in this thesis.

Contents

Abstract . . . . . **iii**

Preface . . . . . **iv**

Contents . . . . . **v**

List of Tables . . . . . **vii**

List of Figures . . . . . **viii**

Glossary of Notation . . . . . **ix**

Acknowledgements . . . . . **xi**

Chapter 1: Introduction **1**
1.1 Comparing Stochastic Optimization Algorithms 1
1.2 Road Design Optimization 3
1.2.1 Horizontal Alignment Optimization 3
1.2.2 Vertical Alignment Optimization 5
1.2.3 Implementation and Surrogate Cost Functions 6
1.3 Challenges and Organization 7

Chapter 2: Bootstrapping the Incumbent Solution Process . **9**
2.1 Preliminaries 9
2.2 Problem Definition 10
2.3 Bootstrap Estimation of the Incumbent Solution Process . . . 13
2.4 Theoretical Analysis 15

Chapter 3: Numerical Validation of the Bootstrap Method . **19**
3.1 Test Setup 19
3.2 Basic Test Results 20
3.3 Reliable Time Interval and Quantiles 25

Chapter 4: Methods to Compare Algorithms 30
4.1 Road Design Optimization Algorithms Compared 31
4.2 Prediction Band with Median 35
4.3 Integrated P-P Plot 38
4.4 Speed Quantification 40

Chapter 5: Conclusion 45
5.1 Contribution 45
5.2 Future Work 45

Bibliography 49

List of Tables

Table 4.1 Speed quantification by 50^{th} percentile 43
Table 4.2 Advanced speed quantification by 50^{th} and 90^{th} percentiles 44

List of Figures

Figure 1.1 3D road alignment 4
Figure 1.2 Horizontal road alignment 5
Figure 1.3 Vertical road alignment 6
Figure 1.4 3D alignment optimization framework 7

Figure 3.1 Six instances of the estimate versus the truth 22
Figure 3.2 Mean of estimates 23
Figure 3.3 Mean and standard deviation of estimates 24
Figure 3.4 Relative error for f_1 on $\mathbb{R}^5$ 26
Figure 3.5 Average-case relative error 27
Figure 3.6 Worst-case relative error 27
Figure 3.7 Reliable time intervals 29

Figure 4.1 The 80% prediction band with median for two algorithms 36
Figure 4.2 The 80% prediction band with median for ten algorithms 37
Figure 4.3 Integrated P-P plot 39

Figure 5.1 Uniqueness of λ 47

Glossary of Notation

Ω	Set of all possible restarts. Pg. 10
X	Feasible set. Pg. 10
$f(x)$	Objective function. Pg. 10
τ	Time since the start of the optimization process. Pg. 12
B	Number of bootstrap paths. Pg. 14
M	Number of sample solutions in each bootstrap path. Pg. 14
N	Number of restarts in the test. Pg. 13
N_s	Number of samples drawn in the numerical validation. Pg. 25
τ_{max}	Upper bound of focused time horizon. Pg. 14
Y	Algorithm's output. Pg. 11
T	Algorithm's running time. Pg. 11
$\{(Y_i, T_i)\}_{i=1}^{N}$	A sample of (Y, T). Pg. 13
$\{U_i\}_{i=1}^{\infty}$	Independent and identically distributed discrete uniform random variables between 1 and N. Pg. 13
$\Gamma(\tau)$	Incumbent solution process. Pg. 12
$\hat{\Gamma}(\tau; N)$	Theoretical approximator for $\Gamma(\tau)$. Pg. 13
$\hat{\Gamma}_i(\tau; N)$	The i^{th} unique path of $\hat{\Gamma}(\tau; N)$. Pg. 15
$\hat{\Gamma}_i(\tau; N, B)$	The i^{th} bootstrap path. Pg. 14
$C(\tau)$	Count of outputs generated by the algorithms at time τ. Pg. 16
$\hat{C}(\tau; N)$	Count of outputs in the path of $\hat{\Gamma}(\tau; N)$ by time τ. Pg. 16

$\hat{C}^*(\tau; N)$	Count of unique outputs in the path of $\hat{\Gamma}(\tau; N)$ by time τ. Pg. 17
$C^*(i)$	Count of unique values in $\{Y_j\}_{j=1}^{i}$. Pg. 17
$G_{Y,T}(y,t)$	CDF of (Y,T). Pg. 11
$G_{\Gamma}(y;\tau)$	CDF of $\Gamma(\tau)$. Pg. 13
$G_{\hat{\Gamma}}(y;\tau,N)$	CDF of $\hat{\Gamma}(\tau;N)$. Pg. 14
$\hat{G}_{\hat{\Gamma}}(y;\tau,N,B)$	ECDF of $\hat{\Gamma}_i(\tau;N,B)$; bootstrap estimator of the incumbent solution process. Pg. 14
$\hat{G}^{-1}_{\hat{\Gamma}}(p;\tau,N,B,i)$	Estimate based on the i^{th} random sample in the numerical validation. Pg. 25
$G_{\alpha}(y;\tau)$	Bootstrap estimator for algorithm α. Pg. 30
$G_{\alpha}(y)$	Integrated CDF for algorithm α. Pg. 38

Acknowledgements

I would like to thank my supervisors Dr. Warren Hare and Dr. Jason Loeppky for guiding me through my research towards this thesis. I would also like to thank my committee member Dr. Yves Lucet for helping me set up the environment for implementation of the algorithms tested in this work. In addition, I would like to thank my colleague Soroor Sarafrazi for helping me understand road design optimization models at the early stage of my research. Finally, I would like to thank our industrial partner Softree Technical Systems Inc. for providing test data.

This research is partially funded by a Collaborative Research and Development (CRD) Grant from the Natural Sciences and Engineering Research Council (NSERC) sponsored by Softree Technical System Inc., and by the Graduate Entrance Scholarship and the University Graduate Fellowship from the University of British Columbia. Part of the computation in this research was performed in the Computer-Aided Convex Analysis Laboratory funded by the Canadian Foundation for Innovation (CFI) and by the British Columbia Knowledge Development Fund (BCKDF).

Chapter 1

Introduction

1.1 Comparing Stochastic Optimization Algorithms

Comparing the performance of different optimization algorithms is an important task in modern research [VWD16]. Stochastic optimization algorithms are algorithms designed to solve optimization problems with the use of some random procedures. Since they rely on certain random procedures to seek the solutions to optimization problems, their outputs are usually random. Computational complexity is usually not sufficient for comparison between expensive algorithms, because they tend to belong to the same complexity class. Hence, numerical experiments are commonly used to compare the performance of expensive stochastic optimization algorithms. To properly characterize a stochastic algorithm's performance, test should be conducted over a range of random restarts. Since test data is also random, comparing the performance of different algorithms based on it is nontrivial, especially when limited data is available due to the expensiveness of the algorithm. In this thesis, a methodology is developed for comparing the performance of expensive stochastic optimization algorithms when random restarts are employed.

Researchers have proposed many comparison methods based on running time or number of iterations, e.g., operating characteristics [Gri78], time-to-target plots [FRS94], run-time distributions [HS98], performance profiles [DM02] and data profiles [MW09]. However, none of the above methods addresses how the behavior of stochastic algorithms are influenced by random parameters.

Some researchers have examined comparison methods for stochastic optimization algorithms. Since the performance of stochastic algorithms could be sensitive to random inputs such as random seed [FRK$^+$08], performance tests without considering this issue could give misleading results. The operational zone proposed by Sergeyev et al. [SKM16] handled this issue by applying random restarts in the test and showing the best and worst case

performances along with the average performance. This approach is based on the assumption that the end user will solve problems with a single algorithm run. If a problem is solved with a single run, then it would be reasonable to project the performance based on the probability distributions of the solution time and quality. However, in practice it is common to employ multiple random restarts when an optimization problem is solved by a stochastic algorithm [BHHR14, HJK96, KGB13, Spa03]. In particular, when the objective function is non-convex, multiple restarts will help the algorithm find more local minima and thus achieve a better solution [KG09].

For example, given different initial points, the algorithm may converge to different local minima. For algorithms relying on random seeds to determine search paths, different seeds will also result in different local minima. Even if the objective function is convex, different parameter settings may lead to different solutions or running times. For instance, a line search method may return different solutions when different values are used as the step size parameter. Alternatively, a pattern search algorithm using different random seeds may have the same output but different running time, because the pattern may move through different trajectories depending on the random numbers generated by the seed.

In these situations, the probability distributions of the solution time and quality alone cannot disclose sufficient information on the performance of the algorithms. With different restarts, both the solution and the running time can be different, which makes it nontrivial to compare the performance of different algorithms. In addition, if the algorithms being compared are expensive, only limited amount of test data is available for comparison and it becomes more challenging to compare them. Therefore, statistical methods need to be applied to process the test data and make inference about the performance.

Truchet et al. [TRC12] proposed a statistical method to analyze the parallel speedup of Las Vegas Algorithms [Bab79], i.e., algorithms that always give correct results but take random running times. This is the first work to analyze the performance of stochastic algorithms under multiple restarts. However, the method by Truchet et al. [TRC12] is not sufficient for handling the problem of interest in this thesis. Firstly, it focuses on the analysis of an algorithm's speed under parallel rather than sequential restarts. In addition, it is limited to a subclass of stochastic optimization algorithms. A stochastic optimization algorithm in general may return different outputs under different random restarts, thus a new approach is needed to analyze both the speed and solution quality of the algorithms.

Moreover, Truchet et al.'s work mainly relies on the assumption that the running time distribution is exponential or lognormal. This is further based on the assumption that both the solution in the search space and the search speed are uniformly distributed, which is not always satisfied in reality. On the contrary, the statistical method developed in this thesis does not impose any assumption about the distribution of the the algorithm's running time or the output.

1.2 Road Design Optimization

The comparison methods developed in this thesis will be applied to an optimization problem in road design. A good transportation system is essential to the economic development of a country [RCS06, Don10]. Road transportation forms a significant part of all forms of transportation. To build and maintain the road network, governments all over the world need to spend large amount of money. For example, in 2008 the Canadian government spent 20.9 billion dollars, which is 1.6% of the country's GDP to build and maintain 1 million kilometers of road [Can08]. When a road is designed, it is important to take effort in reducing construction costs.

In particular, road design optimization is the problem of minimizing the cost of building a road between two points given the ground profile and design constraints [ABS05]. The geometric specification of the road is called the alignment and is a curve in three-dimensional Euclidean space connecting the start and the end of the road. Usually, road alignment is modeled separately as the horizontal alignment and the vertical alignment [HEAEH98]. Consequently, a complete road design optimization problem is often defined as a bi-level optimization problem, of which the outer level is the horizontal alignment optimization problem and the inner level is the vertical alignment optimization problem [MLH15]. The algorithms tested in this thesis are based on the horizontal alignment optimization model in [MLH15] and the vertical alignment optimization model in [HHLR14]. A brief introduction of the two models are in the next two subsections.

1.2.1 Horizontal Alignment Optimization

The horizontal alignment is the projection of the three-dimensional road alignment onto the xy-plane, as shown in Figure 1.1. In particular, the x and y coordinates represent the location of an alignment and the z coordinate represents the elevation of points in the Figure 1.1. The blue curve is the three-dimensional road alignment in the xyz-space connecting the two

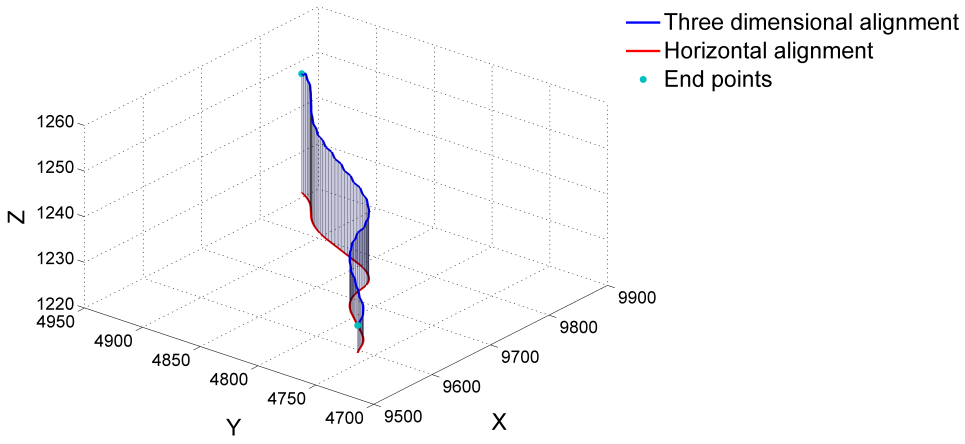


Figure 1.1: 3D road alignment. Picture from [Mon14], used with permission.

end points; the red line is defined on the xy-plane and is the projection of the three-dimensional road alignment. The horizontal alignment consists of straight lines and curves. The framework we present here models the curves as *arcs*. For the smoothness of the road, the straight lines are tangent to the arc(s) they connect to, so they are called *tangents*. To model a horizontal alignment with n turns from its start S and end E, n arcs and $n+1$ tangents are needed. Each arc is connected by two tangents. The first and the last tangent respectively connect S to the first arc and E to the last arc. For the i^{th} turn, the radius of the arc is denoted as $r_i \in \mathbb{R}_+$ and the intersection point of the two tangents connecting the arc is denoted by $P_i \in \mathbb{R}^2$. An example of horizontal alignment modeled by arcs and tangents is shown in Figure 1.2. In particular, the red lines are tangents and the blue curves are arcs. Given any horizontal alignment, a vertical alignment optimization problem can be derived as shown in the next subsection. The cost of a horizontal alignment is the minimum of the vertical alignment problem derived from it.

In particular, the horizontal alignment optimization problem is a black-box optimization problem, which is also known as derivative-free optimization problem [CSV09]. In this problem, the vertical alignment solver is treated as a black-box, which takes the data input and returns the minimum of the vertical alignment problem. A horizontal alignment with n turns can be specified by $\{(r_i, P_i)\}_{i=1}^n$. Given a vertical alignment opti-

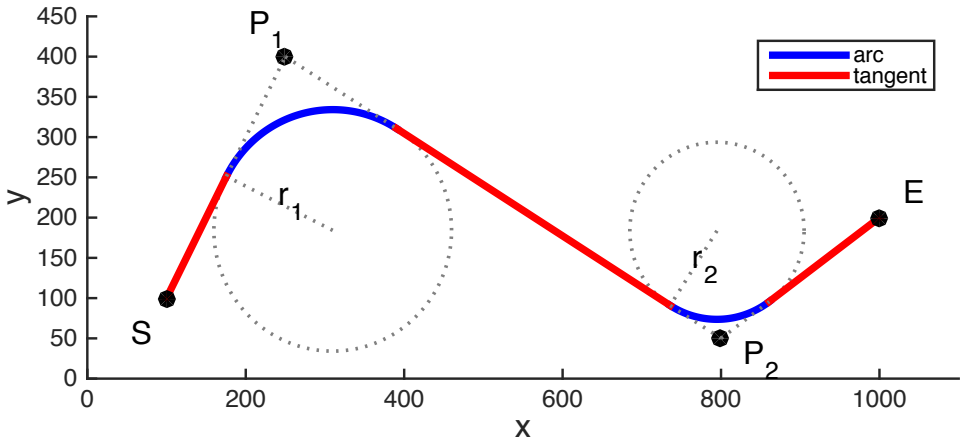


Figure 1.2: Horizontal road alignment

mization model, the cost of constructing a road can also be determined by $\{(r_i, P_i)\}_{i=1}^{n}$, therefore it is the collection of decision variables in the horizontal alignment optimization problem. Three sets of constraints are imposed to ensure that i) the road is continuous, ii) the location of the turns are desired, and iii) the turning radius is large enough for safety. For more details, please refer to [Mon14].

1.2.2 Vertical Alignment Optimization

Given a horizontal alignment, the vertical alignment represents the elevation of the road along the horizontal alignment [Kan08]. Mathematically, it is a function of the distance from the start of the road along its corresponding horizontal alignment. In particular, the model used in this thesis specifies the vertical alignment by quadratic splines, as shown in Figure 1.3.

In practice, terrain elevation data are sampled discretely, hence the costs are calculated only at certain sample points. A vertical alignment is divided into sections, of which each has a sample point for cost calculation. According to [HHLR14], the costs associated with road construction are usually earth cutting, filling and movement costs. For example, to build the road in Figure 1.3, earth needs to be filled to sections 0 to 3, cut from sections 4 to 6 and transported between the sections. Borrow pits and waste pits can be used (at a cost) to deal with absent or excess earth. The objective function of this optimization problem is the sum of all the above costs throughout the entire road. The decision variables are the coefficients of the

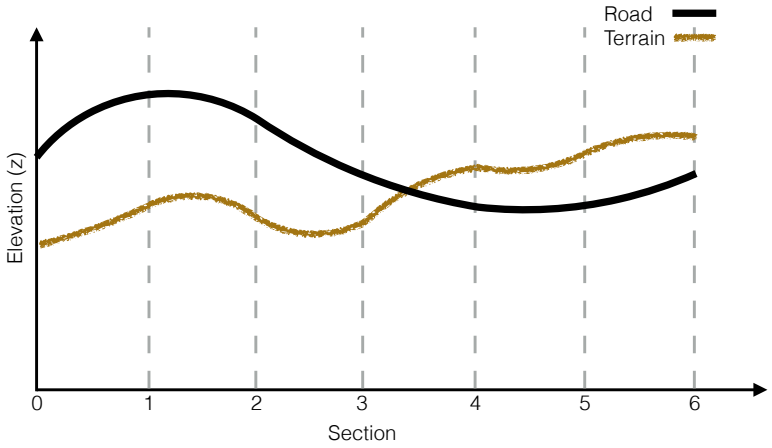


Figure 1.3: Vertical road alignment

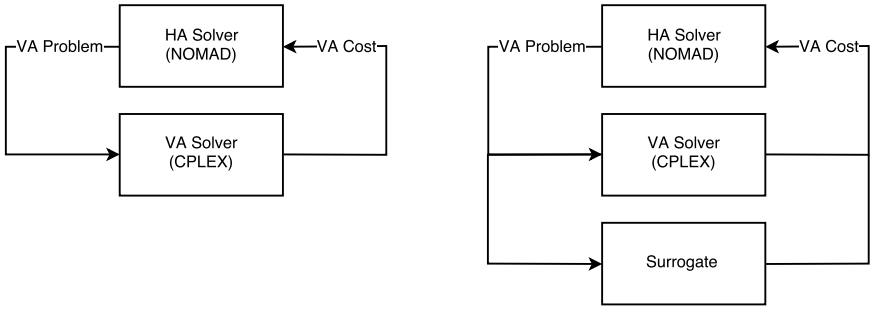
quadratic splines and the volume of earth moved between different sections, borrow pits and waste pits. Constraints are imposed for smoothness of the splines, curvature of the road for safety, balance of earth movements, maximum cut and fill volume, etc. This optimization problem is formulated as a mixed-integer linear program. For more details, please refer to [HHLR14].

1.2.3 Implementation and Surrogate Cost Functions

The vertical and horizontal alignment problems are respectively implemented in IBM ILOG CPLEX 12.5.1 (http://www.cplex.com) and NOMAD 3.7 [AAC+].

Figure 1.4(a) shows the relation between the horizontal and the vertical alignment optimization problems.

In black-box optimization, surrogates, or surrogate functions, are cheaper approximation functions for the actual objective function for the purpose of speeding up the minimization process [VDHL17]. A framework for usage of surrogates in black-box optimization can be found in [BDF+99]. Generally, in each iteration the black-box solver would evaluate the surrogate on a set of points and use the result to guide the evaluation of the actual objective function afterwards. Ideally, the solver can quickly identify potential descent directions with the surrogate and save time on the evaluation of the expensive objective function. In the case of road design optimization, a surrogate would be a function that can provide quick approximation for the minimal cost of the vertical alignment optimization problem. The model framework



(a) Original Model. (b) Surrogate Model.

Figure 1.4: 3D alignment optimization framework. HA and VA correspondingly stand for horizontal alignment and vertical alignment.

of the road alignment optimization problem with surrogate is illustrated in Figure 1.4(b).

Surrogates are usually categorized into two types [BDF+99]. One type is functions built through interpolating or smoothing sample points of the actual objective function. The other is objective functions based on simplified physical models. In addition, some surrogates are dynamic, i.e., able to adjust themselves according to the optimization results, and other surrogates are static, i.e., remain the same during the optimization process. In this thesis, all the surrogates designed are static surrogates based on simplified physical models and described in Section 4.1.

When the surrogate model is implemented the user needs to provide two objective functions, marking one as the true cost function and the other as the surrogate. In the process of optimizing a road alignment with surrogates, the use of the surrogate is managed internally by NOMAD. Specifically, NOMAD evaluates the surrogate function over points in the poll set and use the results to decide the order that the true objective function is evaluated. Section 6.2 of the NOMAD User Guide [ALT09] provides detailed instructions on the implementation of surrogates.

1.3 Challenges and Organization

There are three challenges in analyzing the test data. The first challenge is extracting meaningful information from the test data. During the test, we

observe a particular path of solutions (path is formally defined in Section 2.3), but it is only one realization of many possible paths and does not necessarily reflect the general case. Specifically, when different solutions appear in different orders, the path evolves in different ways. To avoid the bias of any single observation, in Chapter 2 a bootstrap [ET93] technique is developed to simulate numerous solution paths based on the test data, and thus to offer a more comprehensive view on the behavior of the solutions.

Another challenge is making proper use of the bootstrap result. Caution should be taken when using bootstrap to estimate the minimum or maximum of a statistical sample [Che07]. In our case, a point in the solution path is the minimum of all previously generated solutions, therefore we need to understand the limitations of our estimation to the incumbent solution. In Section 2.4, the asymptotic correctness and the bias of the bootstrap estimator are analyzed. In Chapter 3, the reliability of the estimator is demonstrated with an out-of-sample test. To the author's knowledge, this work is the first to analyze the estimation error of the test result for the performance of optimization algorithms.

The last challenge is interpreting the processed data for intuitive comparison of different algorithms. Many popular comparison methods [Gri78, FRS94, HS98, DM02, MW09] successfully compared the performance of optimization algorithms through their speeds for solving certain problems or solution qualities under a time budget. However, none of them are designed for the case where multiple restarts are employed. When the algorithm's output may be different under different random restarts, the user may employ multiple restarts to achieve better results. In this thesis, we analyze stochastic optimization algorithm's performance when multiple random restarts are employed. The major challenge arises when both speed and solution quality are simultaneously considered by modeling the solution path as a stochastic process. In Chapter 4, three comparison methods are demonstrated along with test data of 11 algorithms solving the road design optimization problem mentioned in the previous section. Before introducing the comparison methods, the algorithms being compared are described in Section 4.1.

Chapter 2

Bootstrapping the Incumbent Solution Process

In this chapter, we develop a bootstrap method to model the solution paths of the stochastic optimization algorithms being compared. Since the method will be applied to every algorithm in the same way, by "the algorithm", we mean a given stochastic optimization algorithm.

2.1 Preliminaries

The term incumbent solution usually means the best solution found so far during the process of solving an optimization problem, but the exact definition could be different under different scenarios. Below is a formal definition of the incumbent solution as used in this thesis.

Definition 2.1. When an optimization problem is solved by a stochastic algorithm with random restarts, the *incumbent solution* at a given time is the optimal output of all existing outputs generated by the algorithm since the first restart.

The cumulative distribution function is a very important concept in probability and statistics, since a probability distribution can be fully characterized by its cumulative distribution function.

Definition 2.2. The *cumulative distribution function (CDF)* of a real-valued random variable Z is defined as

$$G_Z(z) = \mathbb{P}(Z \leq z),$$

where $z \in \mathbb{R}$.

The *cumulative distribution function* for a vector of real-valued random variables $(Z_1, Z_2, \ldots, Z_n)$ is defined as

$$G_{Z_1,Z_2,\ldots,Z_n}(z_1, z_2, \ldots, z_n) = \mathbb{P}(Z_1 \leq z_1, Z_2 \leq z_2, \ldots, Z_n \leq z_n),$$

where $z_1, z_2, \ldots, z_n \in \mathbb{R}$.

Definition 2.3. The *indicator function* of a set A is defined as

$$\mathbb{I}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

Definition 2.4. Let $\{Z_i\}_{i=1}^n$ be independent, identically distributed (i.i.d.) real-valued random variables with a common distribution function. Their *empirical cumulative distribution function (ECDF)* is defined as

$$\hat{G}_n(z) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{(-\infty,z]}(Z_i) \ .$$

Definition 2.5. A set of real-valued random variables $Z_1, Z_1, \ldots, Z_n$ are *conditionally independent* given another random variable Z if

$$G_{Z_1,Z_2,\ldots,Z_n}(z_1, z_2, \ldots, z_n|Z) = \prod_{i=1}^{n} G_{Z_i}(z_i|Z),$$

where G is the cumulative distribution function and $z_1, z_2, \ldots, z_n \in \mathbb{R}$.

Definition 2.6. A sequence of random variables $\{Z_n\}$ *converges in probability* to the random variable Z, or $Z_n \to Z$ *in probability*, if

$$\lim_{n\to\infty} \mathbb{P}(|Z_n - Z| \geq \epsilon) = 0$$

for all $\epsilon > 0$.

Definition 2.7. Let $\hat{\theta}$ be an estimator of a real number θ. The *bias* of $\hat{\theta}$ is defined as

$$\mathbb{E}[\hat{\theta}] - \theta.$$

2.2 Problem Definition

The algorithm is used to solve the following problem with different random restarts:

$$\begin{aligned} &\min f(x) : \mathbb{R}^n \mapsto \mathbb{R} \\ &\text{subject to: } x \in X \subseteq \mathbb{R}^n. \end{aligned} \tag{2.1}$$

Let $(\Omega, \mathbb{P})$ be a probability space, where Ω is the set of all possible restarts and $\mathbb{P}$ is the probability of using certain restart(s). Given certain restart,

the algorithm becomes deterministic, thus we can assume it returns a deterministic output and spends a deterministic amount of CPU time under the same test environment. We then define a random variable

$$(Y, T) : \Omega \mapsto \mathbb{R} \times \mathbb{R}_{++},$$

where $Y(\omega)$ is the output produced and $T(\omega)$ is the time spent if $\omega \in \Omega$ is employed. The unknown cumulative distribution function of (Y, T) is denoted by

$$G_{Y,T}(y, t).$$

To understand this framework better, we present the following example.

Example 2.1. Consider the following instance of Problem 2.1:

$$\begin{aligned} &\min f(x) : \mathbb{R}^n \mapsto \mathbb{R} \\ &\text{subject to: } x \in [0, 1]^n. \end{aligned} \tag{2.2}$$

Suppose the problem above is approached by the Random Pursuit Algorithm

Algorithm 1: Random Pursuit

Input:
A problem of the form (2.2)
$K \in \mathbb{N}$: number of iterations
x_0 : an initial point
Output: approximate solution x_K to (2.2)
for $k \leftarrow 0$ *to* $K - 1$ **do**
  choose u_k uniformly at random from $\{y \in \mathbb{R}^n : \|y\|_2 = 1\}$
  set $x_{k+1} \leftarrow x_k + LS(x_k, u_k) \cdot u_k$
end

[SMG13] shown in Algorithm 1, where $LS(x_k, u_k)$ is a line search subroutine (e.g., the golden-section search [Kie53]). Depending on the user's strategy for employing restarts, there are a few possible cases of random restarts as discussed below.

Each case is characterized by $(\Omega, \mathbb{P})$. The probability measure $\mathbb{P}$ is sometimes much easier to describe in terms of probability distributions, especially when $\Omega \subset \mathbb{R}$. However, the probability space $(\Omega, \mathbb{P})$ itself is not a random variable. If we define the random variable

$$I(\omega) : \Omega \mapsto \Omega = w$$

as the identity map, then I is a random variable on Ω with probability measure $\mathbb{P}$. In this way, the law of $\mathbb{P}$ can be shown via the probability distribution of I with sufficient technical validity. For technical details of probability measure and random variable, please refer to any measure-theoretic probability theory book, for example [JP03].

Next, we describe three cases of random restarts.

Case 1: The random seed. The random search directions $\{u_k\}_{k=0}^{K-1}$ depend on a sequence of pseudo-random numbers, which depend on the random seed given to the algorithm. Therefore, the random seed affects the result through search directions. If the algorithm accepts 32-bit unsigned binary integer values as random seeds, then $\Omega = \{0, 1, \ldots, 2^{32} - 1\}$. Usually random seeds are sampled uniformly from Ω and $I \sim \mathcal{U}\{0, 2^{32} - 1\}$.

Case 2: Random initial point. Searching from different initial points can also result in different solutions. Without any prior understanding of the objective function, the user can uniformly sample initial points from the feasible region $[0, 1]^n$, then I follows a multi-variate continuous uniform distribution between 0 and 1.

Case 3: Combination of multiple cases. For instance, when Case 1 and Case 2 are combined, the setup can be $\Omega = \{0, 1, \ldots, 2^{32} - 1\} \times [0, 1]^n$ and $I = (I_1, I_2)$, where $I_1 \sim \mathcal{U}\{0, 2^{32} - 1\}$ and I_2 follows a multi-variate continuous uniform distribution between 0 and 1.

□

To compare the performance of different algorithms, we need to understand how the incumbent solution evolves during the optimization process. We model the incumbent solution by the following stochastic process.

Definition 2.8. Given a stochastic optimization algorithm, the *incumbent solution process* is defined as

$$\Gamma(\tau) = \min \left\{ \inf_{k \in \mathbb{N}} \left\{ Y_k : \sum_{i=1}^{k} T_i \leq \tau \right\}, f(x_0) \right\},$$

where $\tau \geq 0$ is the time since the start of the optimization process, the sequence $\{(Y_i, T_i)\}_{i=1}^{\infty}$ are i.i.d. random variables following the distribution of (Y, T) and x_0 is a given point in the feasible set X.

When $\tau < T_1$, the infimum in the above definition is $+\infty$, because the set it corresponds to is $\emptyset$. Thus, the value $f(x_0)$ is used to avoid ∞ when τ is small. The value $f(x_0)$ can be considered as the minimal known value of

f before the optimization process. Clearly, the choice of x_0 in a particular optimization problem should be the same for different algorithms for a fair comparison of the solution quality.

We want to compare the performance of different algorithms by comparing their incumbent solution processes. Since the incumbent solution process $\Gamma(\tau)$ is characterized by the unknown $G_{Y,T}(y,t)$, we cannot fully understand it. However, we can estimate it based on an observed sample of (Y,T) in the test. Next, we will develop an estimator for the incumbent solution process's cumulative distribution function,

$$G_\Gamma(y;\tau) = \mathbb{P}\left(\Gamma(\tau) \leq y\right), \tag{2.3}$$

through bootstrapping from the sample.

2.3 Bootstrap Estimation of the Incumbent Solution Process

We now introduce a theoretical framework for the bootstrap method. As mentioned earlier, $G_{Y,T}(y,t)$ characterizes $\Gamma(\tau)$ and thus $G_\Gamma(y;\tau)$, but it is unknown. In the test, we observe $\{(Y_i,T_i)\}_{i=1}^N$, which is a sample of (Y,T) with size N. It follows that (Y_i,T_i) has the same distribution as (Y,T) for all $1 \leq i \leq N$.

Definition 2.9. Given a sequence $\{(Y_i,T_i)\}_{i=1}^N$ of i.i.d. random variables following the distribution of (Y,T), the *theoretical approximator* for the incumbent solution process is a stochastic process defined as

$$\hat{\Gamma}(\tau;N) = \min\left\{\inf_{k\in\mathbb{N}}\left\{Y_{U_k} : \sum_{i=1}^{k} T_{U_i} \leq \tau\right\}, f(x_0)\right\},$$

where x_0 is a given point in the feasible set X and $\{U_i\}_{i=1}^\infty$ are i.i.d. discrete uniform random variables between 1 and N.

Given the above definition, $\{(Y_{U_i},T_{U_i})\}_{i=1}^\infty$ are i.i.d. random variables and

$$\mathbb{P}\left\{(Y_{U_i},T_{U_i}) = (Y_j,T_j)\right\} = \frac{1}{N}$$

for all $i \geq 1$ and $1 \leq j \leq N$. Resampled with replacement from $\{(Y_i,T_i)\}_{i=1}^N$, the set $\{(Y_{U_i},T_{U_i})\}_{i=1}^\infty$ approximates $\{(Y_i,T_i)\}_{i=1}^\infty$. As a result, $\hat{\Gamma}(\tau;N)$ approximates $\Gamma(\tau)$ based on the observations $\{(Y_i,T_i)\}_{i=1}^N$.

Next, we introduce the implementation of the bootstrap method. Given $\{(Y_i, T_i)\}_{i=1}^N$, the cumulative distribution function of the theoretical approximator is defined by

$$G_{\hat{\Gamma}}(y; \tau, N) = \mathbb{P}\{\hat{\Gamma}(\tau; N) \leq y\}, \tag{2.4}$$

and is deducible analytically. However, it is difficult to compute, therefore it is estimated by its empirical cumulative distribution function based on a finite collection of $\{(Y_{U_i}, T_{U_i})\}_{i=1}^{\infty}$. We simulate B random paths of the theoretical approximator based on $\{(Y_i, T_i)\}_{i=1}^N$ for $0 \leq \tau \leq \tau_{max}$, where τ_{max} is the upper bound of the time horizon we focus on. To ensure data sufficiency for each path, we generate

$$M = \left\lfloor \frac{\tau_{max}}{\min\limits_{1 \leq i \leq N} T_i} \right\rfloor \tag{2.5}$$

sample outputs in each path. In particular, we generate only the first $B \cdot M$ elements of $\{(Y_{U_i}, T_{U_i})\}_{i=1}^{\infty}$ and reindex them as

$$\{(Y^*_{i,j}, T^*_{i,j})\}_{i,j=1}^{i=B,j=M}.$$

Definition 2.10. Given resampled test data $\{(Y^*_{i,j}, T^*_{i,j})\}_{i,j=1}^{i=B,j=M}$, the i^{th} *bootstrap path* is defined as

$$\hat{\Gamma}_i(\tau; N, B) = \min\left\{ \inf_{1 \leq k \leq M} \left\{ Y^*_{i,k} : \sum_{j=1}^{k} T^*_{i,j} \leq \tau \right\}, f(x_0) \right\},$$

where x_0 is a given point in the feasible set X, $1 \leq i \leq B$ and $0 \leq \tau \leq \tau_{max}$.

Finally, our estimator is given by the following definition.

Definition 2.11. The *bootstrap estimator* for the cumulative distribution function of the incumbent solution process is defined as the bootstrap paths' empirical cumulative distribution function

$$\hat{G}_{\hat{\Gamma}}(y; \tau, N, B) = \frac{1}{B} \sum_{i=1}^{B} \mathbb{I}_{(-\infty, y]} \left(\hat{\Gamma}_i(\tau; N, B) \right).$$

2.4 Theoretical Analysis

In this Section, we will first show the bootstrap estimator proposed in Section 2.3 is asymptotically consistent. The following fact provides a foundation for the proof.

Fact 2.12. *[BF81, Corollary 4.1] Let $\{Z_i\}_{i=1}^N$ be a sample from an unknown cumulative distribution function G, which is estimated by $\{Z_i\}_{i=1}^N$'s empirical cumulative distribution function G_N. Given $\{Z_i\}_{i=1}^N$, let $\{Z_i^*\}_{i=1}^B$ be conditionally independent, with common distribution G_N. Let G_{NB} be the empirical cumulative distribution function of $\{Z_i^*\}_{i=1}^B$. Then, as N and $B \to \infty$, $\|G_{NB} - G\| \to 0$ in probability.*

Theorem 2.13. *Given any $\tau_{max} \in \mathbb{R}_+$, $\|\hat{G}_\Gamma(y;\tau,N,B) - G_\Gamma(y;\tau)\| \to 0$ in probability for all $\tau \in [0,\tau_{max}]$ as $N \to \infty$ and $B \to \infty$.*

Proof. We first construct the elements needed for the use of Fact 3.1. Since $\hat{\Gamma}(\tau;N)$ is based on a finite sample $\{(Y_i,T_i)\}_{i=1}^N$, given any $\tau_{max} \in \mathbb{R}^+$, it has finitely many unique paths for $\tau \in [0,\tau_{max}]$. Let this finite number be denoted as $N^*(\tau_{max})$, then the collection of unique paths of $\hat{\Gamma}(\tau;N)$ can be denoted as $\{\hat{\Gamma}_i(\tau;N)\}_{i=1}^{N^*(\tau_{max})}$. Each unique path has the same probability, thus $G_{\hat{\Gamma}}(y;\tau,N)$ is the empirical cumulative distribution function of $\{\hat{\Gamma}_i(\tau;N)\}_{i=1}^{N^*(\tau_{max})}$.

Next, we apply Fact 3.1 to show the result. For any $\tau \in [0,\tau_{max}]$, the set $\{\hat{\Gamma}_i(\tau;N)\}_{i=1}^{N^*(\tau_{max})}$ is a sample from the unknown distribution $G_\Gamma(y;\tau)$, which is estimated by $\{\hat{\Gamma}_i(\tau;N)\}_{i=1}^{N^*(\tau_{max})}$'s empirical cumulative distribution function $G_{\hat{\Gamma}}(y;\tau,N)$. Given the sample $\{\hat{\Gamma}_i(\tau;N)\}_{i=1}^{N^*(\tau_{max})}$, it follows that $\{\hat{\Gamma}_i(\tau;N,B)\}_{i=1}^B$ are conditionally independent with common distribution $G_{\hat{\Gamma}}(y;\tau,N)$. The function $\hat{G}_\Gamma(y;\tau,N,B)$ is the empirical cumulative distribution function of $\{\hat{\Gamma}_i(\tau;N,B)\}_{i=1}^B$. As $N \to \infty$ and $B \to \infty$, we also have $N^*(\tau_{max}) \to \infty$ and $B \to \infty$, then Fact 2.12 implies that $\|\hat{G}_\Gamma(y;\tau,N,B) - G_\Gamma(y;\tau)\| \to 0$ in probability for all $\tau \in [0,\tau_{max}]$.

□

Although the bootstrap estimator is asymptotically consistent by Theorem 2.13, it has a bias when N and B are finite. Next we will show a property of the bias.

The following assumption facilitates a step in the proof of Theorem 2.15. It assumes the bootstrap method introduced in Section 2.3 can give an unbiased estimation for the distribution of a counting process associated with the incumbent solution process.

Assumption 2.1. *Let $C(\tau)$ be a random variable representing the count of outputs generated by the algorithm at time τ, i.e., the maximal index i satisfying the constraint in Definition 2.8. Let $\hat{C}(\tau; N)$ be a random variable representing the count of outputs in the path of $\hat{\Gamma}(\tau; N)$ by time τ, i.e., the maximal index j satisfying the constraint in Definition 2.9. Assume that the expected distribution of $\hat{C}(\tau; N)$ is the same as the distribution of $C(\tau)$, i.e., for all i we have:*

$$\mathbb{E}\left[\mathbb{P}(\hat{C}(\tau; N) = i)\right] = \mathbb{P}(C(\tau) = i).$$

The following fact is a basic property of Order Statistics and is important for the proof of the next theorem.

Fact 2.14. *[DN05, (2.12)] Let $Z_1, Z_2, \ldots, Z_n$ be independent and identically distributed random variables with cumulative distribution function $G(z)$. Then the cumulative distribution function of the smallest element in $Z_1, Z_2, \ldots, Z_n$ is given by*

$$1 - [1 - G(z)]^n.$$

Theorem 2.15. *Given Assumption 2.1, the bootstrap estimator's bias*

$$\mathbb{E}[\hat{G}_{\hat{\Gamma}}(y; \tau, N, B)] - G_{\Gamma}(y; \tau)$$

is non-positive.

Proof. The estimator $\hat{G}_{\hat{\Gamma}}(y; \tau, N, B)$ has two sources of uncertainty: the random sample

$$\{(Y_i, T_i)\}_{i=1}^{N}$$

and the random bootstrap paths

$$\{(Y^*_{i,j}, T^*_{i,j})\}_{i,j=1}^{i=B,j=M}.$$

We first eliminate the latter in the expectation:

$$\begin{aligned}
&\mathbb{E}[\hat{G}_{\hat{\Gamma}}(y; \tau, N, B)] - G_{\Gamma}(y; \tau) && (2.6)\\
=&\mathbb{E}\left\{\mathbb{E}\left[\hat{G}_{\hat{\Gamma}}(y; \tau, N, B) | \{(Y_i, T_i)\}_{i=1}^{N}\right]\right\} - G_{\Gamma}(y; \tau) && (2.7)\\
=&\mathbb{E}\left\{G_{\hat{\Gamma}}(y; \tau, N)\right\} - G_{\Gamma}(y; \tau), && (2.8)
\end{aligned}$$

where the last step is a result of the fact that $\hat{G}_{\hat{\Gamma}}(y; \tau, N, B)$ is defined as an empirical CDF of $\hat{\Gamma}(\tau; N)$ given any $\{(Y_i, T_i)\}_{i=1}^{N}$.

Next, we will analyze

$$\mathbb{E}\left[G_{\hat{\Gamma}}(y;\tau,N)\right] - G_{\Gamma}(y;\tau)$$

as a proxy of $\mathbb{E}[\hat{G}_{\hat{\Gamma}}(y;\tau,N,B)] - G_{\Gamma}(y;\tau)$.

Case 1: $0 \leq \tau < \min\{T_i\}_{i=1}^{N}$. In this case,

$$\Gamma(\tau) \leq f(x_0) = \hat{\Gamma}(\tau;N) \tag{2.9}$$

$$\Rightarrow G_{\Gamma}(y;\tau) \geq \hat{G}_{\hat{\Gamma}}(y;\tau,N) \quad \forall\{(Y_i,T_i)\}_{i=1}^{N} \tag{2.10}$$

$$\Rightarrow G_{\Gamma}(y;\tau) \geq \mathbb{E}\left[\hat{G}_{\hat{\Gamma}}(y;\tau,N)\right] \tag{2.11}$$

$$\Rightarrow \mathbb{E}\left[\hat{G}_{\hat{\Gamma}}(y;\tau,N)\right] - G_{\Gamma}(y;\tau) \leq 0. \tag{2.12}$$

Hence, we proved the non-positivity of the bias for the first case.

Case 2: $\tau \geq \min\{T_i\}_{i=1}^{N}$. Let $G_Y(y)$ denote the marginal cumulative distribution function of Y. Let $\hat{C}^*(\tau;N)$ be the count of unique outputs in the path of $\hat{\Gamma}(\tau;N)$ by time τ, i.e., the number of unique values in $\{Y_i\}_{i=1}^{\hat{C}(\tau;N)}$. We introduce $\hat{C}^*(\tau;N)$ to enable the use of Fact 2.14. Let $C^*(i)$ be the count of unique values in $\{Y_j\}_{j=1}^{i}$. All of $C(\tau)$, $\hat{C}(\tau;N)$, $\hat{C}^*(\tau;N)$ and $C^*(i)$ are random. The process $\hat{C}(\tau;N)$ solely depends on the distribution of $\{T_i\}_{i=1}^{N}$, but $C^*(i)$ is independent of $\{T_i\}_{i=1}^{N}$. Therefore, $\hat{C}(\tau;N)$ and $C^*(i)$ are independent as needed in the proof below. Now, by Fact 2.14,

$$\mathbb{E}\left[G_{\hat{\Gamma}}(y;\tau,N)\right] - G_{\Gamma}(y;\tau) \tag{2.13}$$

$$= \mathbb{E}\left[1-(1-G_Y(y))^{\hat{C}^*(\tau;N)}\right] - \mathbb{E}\left[1-(1-G_Y(y))^{C(\tau)}\right] \tag{2.14}$$

$$= 1 - \mathbb{E}\left[(1-G_Y(y))^{\hat{C}^*(\tau;N)}\right] - 1 + \mathbb{E}\left[(1-G_Y(y))^{C(\tau)}\right] \tag{2.15}$$

$$= \mathbb{E}\left[(1-G_Y(y))^{C(\tau)}\right] - \mathbb{E}\left[(1-G_Y(y))^{\hat{C}^*(\tau;N)}\right] \tag{2.16}$$

$$= \mathbb{E}\left[(1-G_Y(y))^{C(\tau)}\right] - \mathbb{E}\left[\mathbb{E}\left[(1-G_Y(y))^{\hat{C}^*(\tau;N)} | \{(Y_i,T_i)\}_{i=1}^{N}\right]\right] \tag{2.17}$$

$$= \sum_{i=1}^{\infty} \mathbb{P}(C(\tau)=i)(1-G_Y(y))^i - \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{P}(\hat{C}(\tau;N)=i)(1-G_Y(y))^{C^*(i)}\right] \tag{2.18}$$

$$\begin{aligned} &= \sum_{i=1}^{\infty} \mathbb{P}(C(\tau)=i)(1-G_Y(y))^i \\ &\quad - \sum_{i=1}^{\infty} \mathbb{E}\left[\mathbb{P}(\hat{C}(\tau;N)=i)\right] \mathbb{E}\left[(1-G_Y(y))^{C^*(i)}\right] \end{aligned} \tag{2.19}$$

$$= \sum_{i=1}^{\infty} \mathbb{P}(C(\tau) = i)(1 - G_Y(y))^i - \sum_{i=1}^{\infty} \mathbb{P}(C(\tau) = i)\mathbb{E}\left[(1 - G_Y(y))^{C^*(i)}\right] \tag{2.20}$$

$$= \sum_{i=1}^{\infty} \mathbb{P}(C(\tau) = i)\mathbb{E}\left[(1 - G_Y(y))^i - (1 - G_Y(y))^{C^*(i)}\right], \tag{2.21}$$

where (2.20) is due to Assumption 2.1.

Since $C^*(i) \leq i$ is always true and $0 \leq 1 - G_Y(y) \leq 1$, then

$$(1 - G_Y(y))^i - (1 - G_Y(y))^{C^*(i)} \leq 0 \tag{2.22}$$

is always true. Thus,

$$\mathbb{E}\left[(1 - G_Y(y))^i - (1 - G_Y(y))^{C^*(i)}\right] \leq 0. \tag{2.23}$$

As $\mathbb{P}(\cdot) \geq 0$, for all $i \geq 1$ we have

$$\mathbb{P}(C(\tau) = i)\mathbb{E}\left[(1 - G_Y(y))^i - (1 - G_Y(y))^{C^*(i)}\right] \leq 0. \tag{2.24}$$

It therefore follows that

$$\mathbb{E}\left[G_{\hat{\Gamma}}(y; \tau, N)\right] - G_\Gamma(y; \tau) \tag{2.25}$$

$$= \sum_{i=1}^{\infty} \mathbb{P}(C(\tau) = i)\mathbb{E}\left[(1 - G_Y(y))^i - (1 - G_Y(y))^{C^*(i)}\right] \tag{2.26}$$

$$\leq 0, \tag{2.27}$$

which shows the non-positivity of the bias for the second case.

In conclusion, $\mathbb{E}[\hat{G}_{\hat{\Gamma}}(y; \tau, N, B)] - G_\Gamma(y; \tau) = \mathbb{E}\left[G_{\hat{\Gamma}}(y; \tau, N)\right] - G_\Gamma(y; \tau) \leq 0$. □

As a consequence of the above theorem, we have $\mathbb{E}[\hat{G}^{-1}_{\hat{\Gamma}}(y; \tau, N, B)] - G^{-1}_\Gamma(y; \tau) \geq 0$, i.e., the expected estimations of the incumbent solution's quantiles are not less than their true values at any time τ.

Chapter 3

Numerical Validation of the Bootstrap Method

In the previous chapter, we introduced an estimator for the incumbent solution process and analyzed its theoretical properties. In this section, we numerically validate the estimator through an out-of-sample test.

3.1 Test Setup

In the test, we minimize five objective functions. In particular, we selected five non-convex functions from [Ort12] and modified them in order that they have more local minima. The objective functions are as follows:

$$f_1(x) = \exp\left\{\frac{\max\left\{\left|\cos\left(\left|\sin(|x_i|)\right|\right)\right| : i = 1, 2, \ldots, n\right\}}{\eta[|\tan(|\cot(|x_1|)|)|, |\tan(|\cot(|x_2|)|)|, \ldots, |\tan(|\cot(|x_n|)|)|]}\right\},$$

where $\eta(x_1, x_2, \ldots, x_n)$ is the median of $x_1, x_n, \ldots, x_n$,

(3.1)

$$\begin{aligned} f_2(x) = \ln\Bigg(\sum_{i=1}^{m}\sum_{j=i+1}^{m} &\sin^2(3\pi x_i) + (x_i - 1)^2\left[\sin^2(3\pi x_j) + 1\right] \\ &+ (x_j - 1)^2\left[\sin^2(2\pi x_j) + 1\right]\Bigg), \end{aligned} \tag{3.2}$$

$$f_3(x) = \left|100 - \frac{\|x\|}{\pi}\right| + \sum_{i=1}^{n}\log\left(\left|\sin(x_i)\right|\right), \tag{3.3}$$

$$\begin{aligned} f_4(x) = \ln\Bigg(\Bigg|\sum_{i=1}^{m}\sum_{j=i+1}^{m} &(x_j + 47)\sin\left(\sqrt{\left|\frac{x_i}{2} + x_j + 47\right|}\right) \\ &+ x_i\sin\left(\sqrt{|x_i - x_j - 47|}\right)\Bigg|\Bigg), \end{aligned} \tag{3.4}$$

$$f_5(x) = -\sum_{i=1}^{m}\sum_{j=i+1}^{m}\left|\sin(x_i)\cos(x_j)\exp\left(\left|1-\frac{1}{\pi}\sqrt{x_i^2+x_j^2}\right|\right)\right|. \tag{3.5}$$

Each of the 5 objective functions is applied with 5, 10 and 50 variables, resulting in 15 test functions. Setting the feasible set

$$X = \{(x_1, x_2, \ldots, x_n) : -99 \leq x_i \leq 99, i = 1, 2, \ldots, n\}$$

for all test functions, we have 15 test problems of the form (2.1).

The derivative-free optimization algorithm LTMADS [AD06] is used to solve all of the test problems. The LTMADS algorithm is a direct search method that uses random numbers to help determine poll directions. Fixing a random seed can make the algorithm deterministic. We let Ω represent the set of random seeds. For the purpose of numerical testing, we assume that all positive integers not larger than 10^5 are used by the user as random seeds, we can set $\Omega = \{1, 2, \ldots, 10^5\}$. In the test, we solve each test problem with all $\omega \in \Omega$.

The derivative-free optimization software NOMAD 3.7.2 [AAC$^+$] is based on LTMADS and used as the solver in the numerical experiments. All the computer codes are programed in MATLAB 2014b. The MATLAB version of NOMAD is called through OPTI Toolbox [CW12]. All of the numerical experiments in this chapter are performed on an Apple iMac with a 3.5 GHz Intel Core i5 processor, 8 GB of RAM and a 64-bit Windows 8 operating system under Boot Camp.

3.2 Basic Test Results

We use the data set from the numerical experiments to demonstrate the accuracy of the bootstrap estimator by comparing it with the incumbent solution process. Since all the test functions are cheap, we are able to generate the test result for all $\omega \in \Omega$. Therefore, we know the distribution of (Y, T) and can deduce the distribution of $\Gamma(\tau)$ in theory. However, calculating the CDF of $\Gamma(\tau)$ is not computationally tractable. Instead, we approximate the CDF of $\Gamma(\tau)$ by its empirical CDF. In particular, we randomly generate 10^5 independent paths of $\Gamma(\tau)$ and derive their empirical cumulative distribution function, which we call the "truth".

To reflect its behavior when the objective functions are expensive and only limited data is available, we generate the bootstrap estimator based on small samples from the test data and call it the "estimate". We test using $N \in \{50, 100, 200, 400\}$ and $B \in \{10^3, 10^4, 10^5\}$. To explore the variability of

the estimator, for each combination of N and B, we generate 1000 *random samples* of a subset of the test data, each of which has a size of N and corresponds to a particular realization of $\{(Y^*_{i,j}, T^*_{i,j})\}_{i,j=1}^{i=B,j=M}$. In addition, for each test problem we set

$$\tau_{max} = 400\bar{T}, \tag{3.6}$$

where

$$\bar{T} = \frac{1}{|\Omega|}\sum_{\omega\in\Omega} T(\omega) \tag{3.7}$$

normalizes the time unit and provides a convenient reference to the number of restarts. For the rest of this chapter, τ is measured in the unit of $\bar{T}$.

We compare the estimator with the truth through various figures. For all the figures in this section, we only show the test results with f_1, $\mathbb{R}^5$ and $B = 10^5$. The results are similar for different test functions and values of B. The full results are summarized in Figure 3.7 and also available on the Internet[1].

Figure 3.1 gives an intuitive comparison between the estimate and the truth for $N = 100$ and $B = 10^5$. The lines are the medians of the distributions; the lower and upper bounds of the bands are the 10^{th} and 90^{th} percentiles of the distributions. These represent three different samples (one per row) and two different bootstraps for each sample (one per column). It can be observed that the estimates look identical within each row, which indicates the random error from different collections of bootstrap paths is negligable if $B = 10^5$. Hence, for the rest of this chapter we only generate one collection of bootstrap paths for each bootstrap estimate. We also set $B = 10^5$ unless specified otherwise. On the contrary, the accuracy of the estimate varies significantly with the test data sample it is constructed from. Specifically, the instances $(1a)$ and $(1b)$ estimated the truth relatively well. However, the bootstrap estimator may under-estimate or over-estimate the performance of the algorithm when an "unlucky" or "lucky" random sample is drawn as shown by $(2a)/(2b)$ and $(3a)/(3b)$.

Since the behavior of the estimate varies with different random samples, there is a need to show the average case. Instead of showing special instances of the estimate, Figure 3.2 shows the mean of all instances generated in the test. It can be observed that the mean accuracy decays as time increases, but the larger the sample size the slower the decay becomes. Among the

[1] `https://www.dropbox.com/sh/gpldpjg1b7ahr8l/AADzOt5c1w4xC-UU1vGoY_Qua?dl=0`

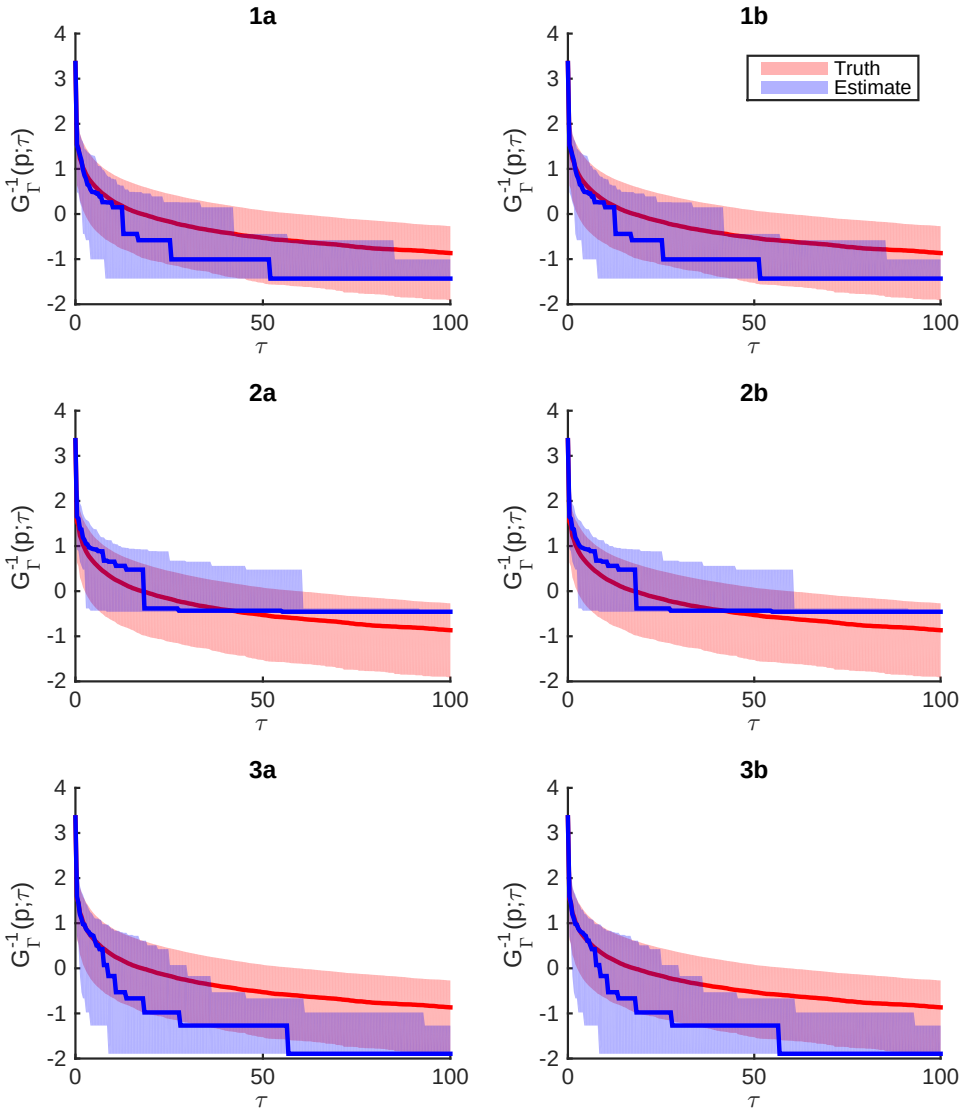


Figure 3.1: Six instances of the estimate versus the truth for f_1 on $\mathbb{R}^5$, $N = 100$ and $B = 10^5$. Three rows of the subplot matrix are based on three different test data samples. In each row, the two subplots are based on the same test data sample, but generated by two different collections of bootstrap paths.

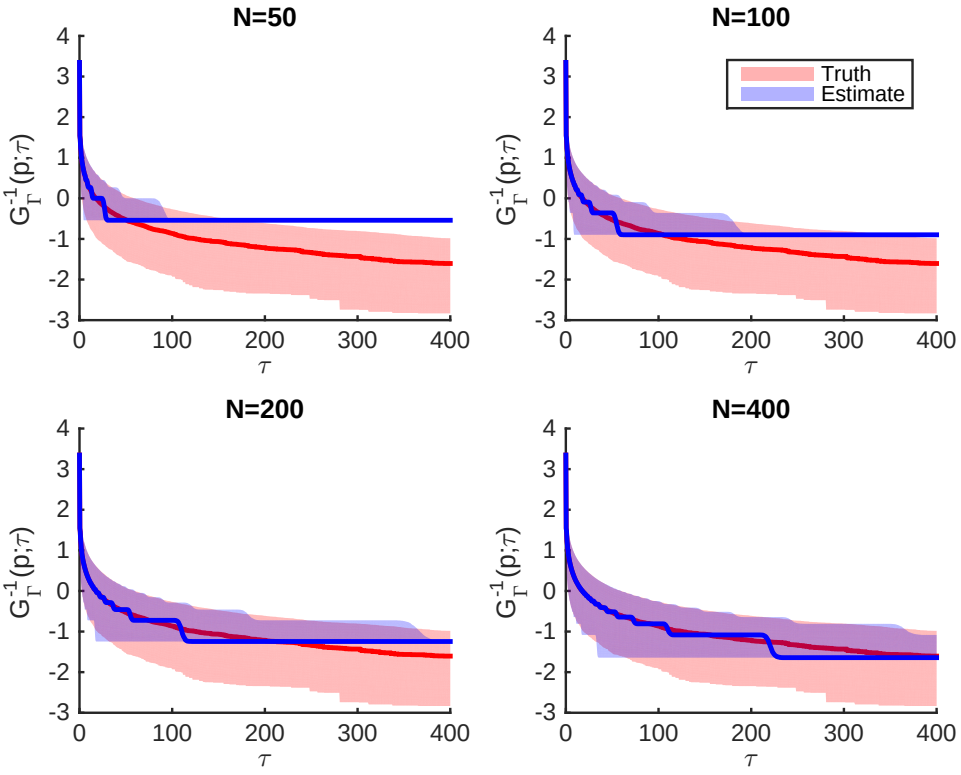


Figure 3.2: Mean of estimates for f_1 on $\mathbb{R}^5$ and $B = 10^5$

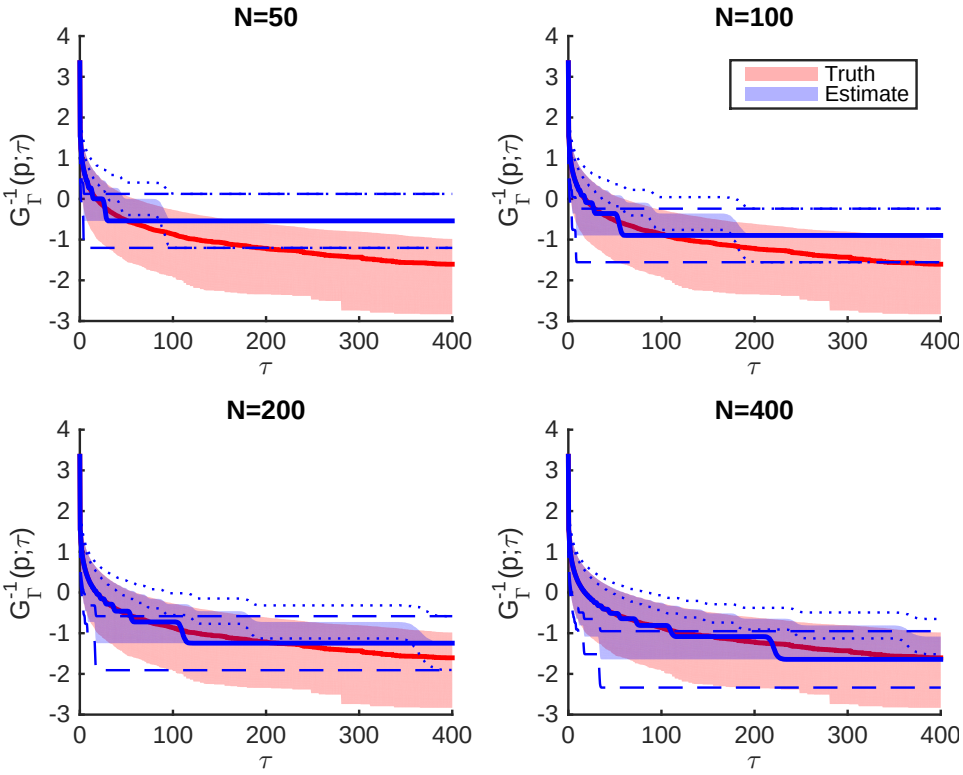


Figure 3.3: Mean and standard deviation of estimates for f_1 on $\mathbb{R}^5$ and $B = 10^5$

three quantiles, the 90^{th} percentile has the best and the 10^{th} percentile has the worst match for the truth. Thus, the mean accuracy of the quantile estimation tends to increase as its corresponding probability increases. Generally, the estimations are not reasonably accurate when $\tau > N$. For example, the estimate stays relatively close to the truth only for $\tau \leq 75$ when $N = 100$ and for $\tau \leq 300$ when $N = 400$.

In addition to Figure 3.2, Figure 3.3 also shows the standard deviation of all the estimates for different τ. In particular, the dotted and dashed lines are one standard deviation away from the mean estimates of the 10^{th} and 90^{th} percentiles. It can be observed that the standard deviation increase as time increases, but the larger the sample size the slower the increase becomes. Moreover, the estimates of the 90^{th} percentile have smaller standard deviations then those of the 10^{th} percentile. Thus, the volatility of the quantile estimation tends to decrease as its corresponding probability

increases.

Even with $B = 10^5$, computing bootstrap estimators is still much cheaper than solving expensive test problem under various random restarts. For instance, in the project demonstrated in Chapter 4, the computation of the bootstrap estimates only took a few minutes for $B = 10^5$, but the samples took a month to generate for $N = 100$. Considering the negligible marginal cost, we would recommend large values of B, e.g., 10^5 or 10^6, to avoid possible unknown flaws. It can be concluded that the bootstrap estimator can accurately predict the incumbent solution process under certain conditions. When used properly, it can save significant computer time.

3.3 Reliable Time Interval and Quantiles

Figures 3.1 to 3.3 intuitively show the accuracy of the bootstrap estimator and it is obvious that the estimator suffers from significant error under some conditions. Next, we will quantify the error and propose a method to avoid excessive error by properly choosing time interval and quantiles.

Definition 3.1. Given a test problem, let $\hat{G}^{-1}_{\hat{\Gamma}}(p;\tau,N,B,i)$ be the estimate based on the i^{th} random sample and N_s be the number of total random samples drawn, then the *relative error* of the bootstrap estimator for probability p (or the p-quantile) and time τ is defined as

$$\frac{\frac{1}{N_s}\sum_{i=1}^{N_s}\left|\hat{G}^{-1}_{\hat{\Gamma}}(p;\tau,N,B,i) - G^{-1}_{\Gamma}(p;\tau)\right|}{f(x_0) - G^{-1}_{\Gamma}(p;\tau)}.$$

The above definition characterizes the deviation of the estimator from the truth. To accurately compare across different test problems, the deviation is normalized relative to the decrease of the incumbent solution.

Figure 3.4 shows the relative error for f_1 on $\mathbb{R}^5$. The observations are consistent with those from previous figures since the relative error increases as τ increases and p decreases. In addition, the increase of N also leads to a decrease of the relative error. For other test problems, the relative errors follow the same pattern and are results are available on the Internet[2].

Figures 3.5 and 3.6 correspondingly show the average-case and worst-case (maximal) relative errors of all test problems when $N = 100$ and $B = 10^5$ are set. It can be observed that the average-case and worst-case relative

[2]The results for other test problems or parameter configurations are available from `https://www.dropbox.com/sh/gpldpjg1b7ahr8l/AADzOt5c1w4xC-UU1vGoY_Qua?dl=0`

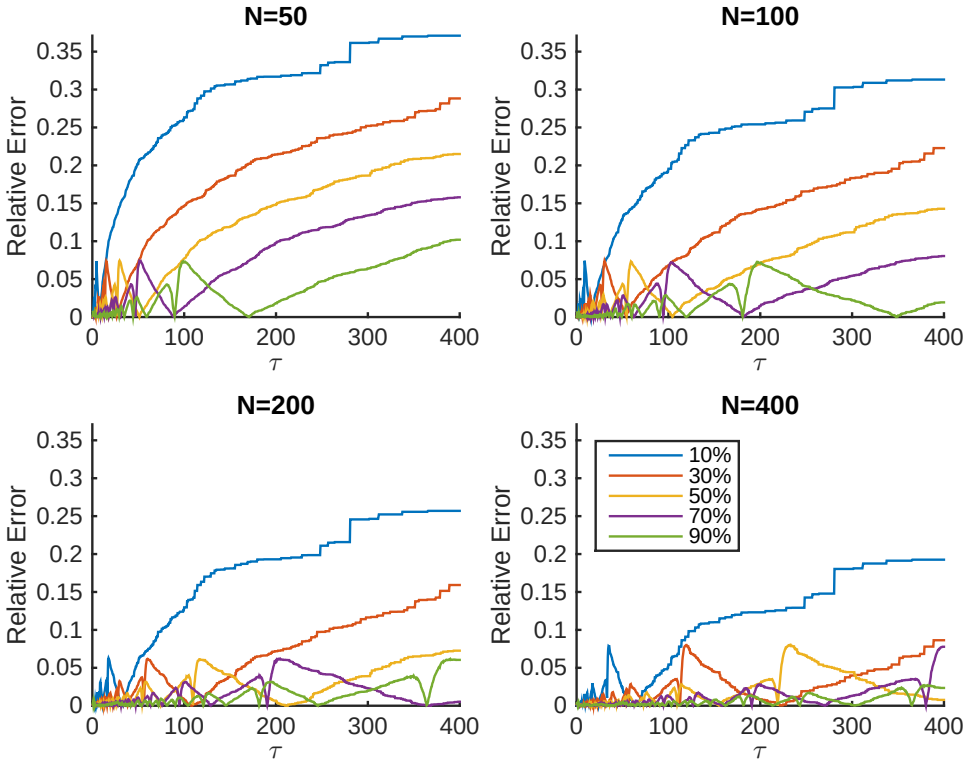


Figure 3.4: Relative error for f_1 on $\mathbb{R}^5$ and $B = 10^5$

errors correspondingly stay below 10% and 20% for most of the probabilities shown and $\tau \in [0, N]$. When N and B are set to other values, similar pattern appears[2].

Next, based on relative error we propose a method to determine time intervals within which the bootstrap estimator remains reliable.

Definition 3.2. Given a test problem, a probability p (or a p-quantile), a threshold δ and τ_{max}, the *reliable time interval* is the largest subinterval of $[0, \tau_{max}]$ within which the corresponding relative error does not exceed δ.

Taking the subplot titled "$N = 400$" in Figure 3.4 for example, if we set $p = 10\%$, $\delta = 0.05$ and $\tau_{max} = 400$, then there are only two groups of subintervals of $[0, \tau_{max}]$ in which the relative error appears to be lower than δ. The first and second group are respectively subsets of $[0, 33.6]$ and $[42.4, 98.8]$. Since $[42.4, 98.8]$ is larger than $[0, 33.6]$, the largest subinterval of $[0, \tau_{max}]$ satisfying the conditions is approximately $[42.4, 98.8]$. Therefore, the reliable time interval in this case is approximately $[42.4, 98.8]$.

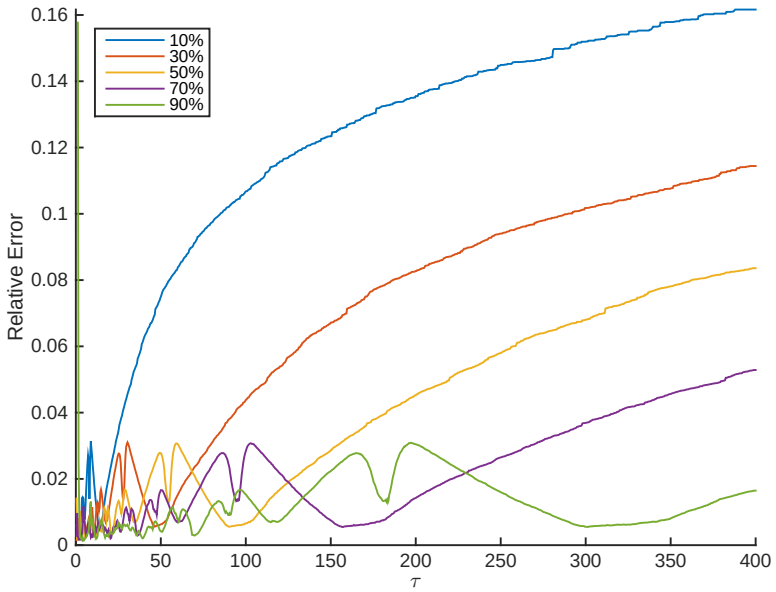


Figure 3.5: Average-case relative error for $N = 100$ and $B = 10^5$

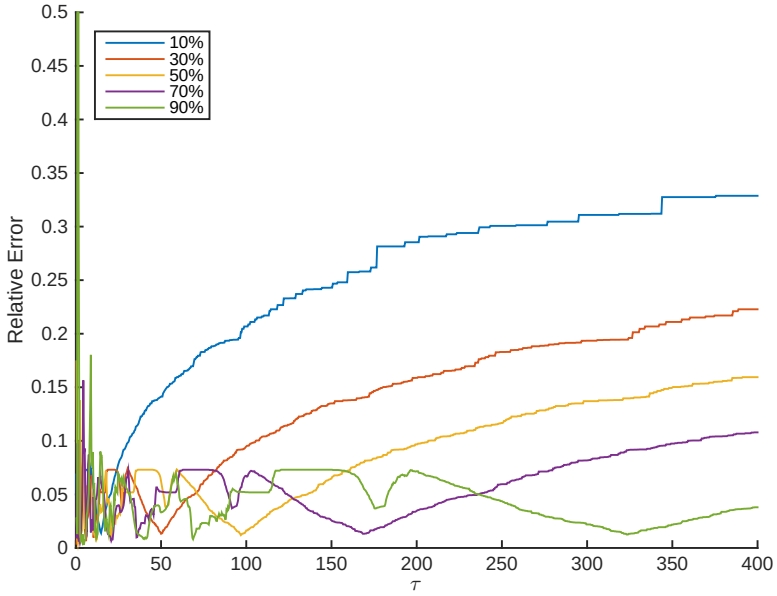


Figure 3.6: Worst-case relative error for $N = 100$ and $B = 10^5$

From Figure 3.4, the reliable time intervals can be determined for only one test problem and five quantiles. On the contrary, Figure 3.7 provides an overview of the reliable time intervals for all test problems, quantiles and values of N and B. Each color corresponds to one of the test problems. Given a test problem and a value p, the interval between the dotted and solid lines represents the reliable time interval for the threshold of 10%. It should be noted that there are multiple vertical lines at $\tau = 0$ and $\tau = 400 = \tau_{max}$, but only one of them is visible. Taking the subplot titled "$N = 50$ $B = 1000$" for example, if we set $p = 0.5$, then the reliable time interval of the yellow represented problem is approximately $[5, 250]$, because the horizontal line at $p = 0.5$ intersects with the dotted and solid yellow curves at about $\tau = 5$ and $\tau = 250$.

The observations reflect those from previous figures because of the following. It can be seen that all the reliable time intervals have a lower bound close to 0, but not all of them have an upper bound close to τ_{max}, indicating the reliability of the bootstrap estimator decays over time. Moreover, except for extremely high quantiles and few cases, the higher the quantile the larger the reliable time interval tends to be, indicating the higher the quantile the higher the estimator's predictability. In addition, the length of the reliable time interval increases as N increases, indicating the larger the sample the more reliable estimate can be generated. However, there is no significant difference when B varies. Figure 3.7 provides a guidance for the choice of the quantiles and the time interval to focus on when comparing different algorithms with the bootstrap estimator. For example, if we set $N = 100$ and want to limit the potential relative error to 10%, we can focus on $\tau \in [0, 75]$ and $p \in [10\%, 95\%]$ because most of the reliable intervals for $p \in [10\%, 95\%]$ cover $[0, 75]$.

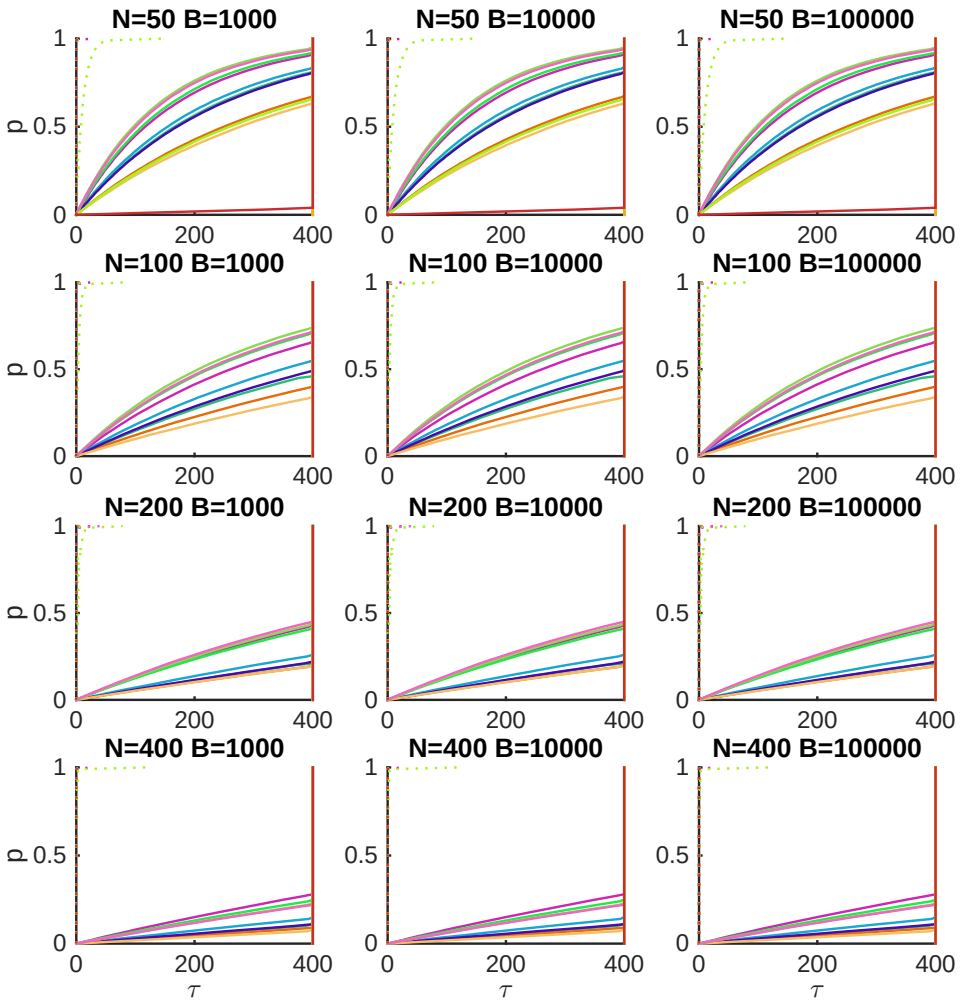


Figure 3.7: Reliable time intervals for $\delta = 10\%$ and $\tau_{max} = 400$ of all test problems

Chapter 4

Methods to Compare Algorithms

In Chapter 2, we introduced a resampling method to estimate the distribution of incumbent solutions over time. Once the estimate is generated, the user can interpret the results in various ways. In this chapter we propose three methods of interpreting the estimators along with numerical results from a real-world case. In this chapter, we use α to index algorithms and $\alpha = 0$ particularly to index the baseline algorithm. Given an algorithm α and fixing parameters N and B, we let

$$G_\alpha(y;\tau) = \hat{G}_{\hat{\Gamma}}(y;\tau,N,B). \tag{4.1}$$

We solved the road alignment optimization problem introduced in Section 1.2 on 10 test problems with 11 algorithms including a baseline algorithm. For each algorithm on each test problem we solved the problem with $N = 100$ subject to the constraint of about one month's computer time for all the algorithms and test problems. Although more test data would increase the accuracy of the results, with a sample of size 100 the bootstrap estimator can already provide considerable prediction for the performance of the algorithms according to the numerical validation in Chapter 3. In addition, we set $B = 10^5$ as justified by the results of Chapter 3. Based on the results of Section 3.3, we also set

$$\tau_{max} = 75\bar{T}',$$

where

$$\bar{T}' = \frac{1}{N}\sum_{i=1}^{N} T_i,$$

and focus on

$$p \in [10\%, 95\%]$$

when comparing algorithms. Throughout this chapter, τ is in the unit of seconds. In Sections 4.2 and 4.3, the figures only show the results of the

first test problem, but the results all follow a similar pattern across all test problems and are available on the Internet[3].

All of the experiments in this chapter were performed on a Dell workstation with an Intel(R) Xenon(R) 2.40 GHz (2 cores) processor, 16 GB of RAM and a 64-bit Windows 7 Enterprise operating system. NOMAD 3.7.3 was used as the solver. The program was coded in C++.

In Section 4.1, we introduce the road design optimization algorithms being compared. Three comparison methods are introduced in the rest of the chapter.

4.1 Road Design Optimization Algorithms Compared

Next, we will describe the road design optimization algorithms being compared. The parentheses after the name of an algorithm contains its abbreviation.

Algorithm 0: the Original Model (**org**)

This is the algorithm based on the model proposed by Mondal, Lucet and Hare [MLH15]. It is considered to be the best existing algorithm, so it is also referred as the "baseline" algorithm. This algorithm only uses true cost function, whereas all the other algorithms introduced below mix the use of the true cost function and certain surrogate cost function, as described in Section 1.2.

Surrogate 1: Constant (**const**)

The constant function

$$f(x) = 0 \tag{4.2}$$

is used as the surrogate. This is designed to test the behavior of the solver with the use of a surrogate that has extremely high speed, but extremely low accuracy.

[3] `https://www.dropbox.com/sh/gpldpjg1b7ahr8l/AADzOt5c1w4xC-UU1vGoY_Qua?dl=0`

Surrogate 2: True Cost Function as Surrogate (=true)

This algorithm uses the true cost function to as the surrogate. This is designed to test the behavior of the solver with the use of a surrogate that has extremely low speed, but extremely high accuracy.

Surrogate 3: Preprocessor Off (preOff)

Preprocessing of a mixed-integer linear program is the process of manipulating variables and constraints in the program before the solver starts to solve the problem. It can make the actual solving process more effective, but itself also consumes extra computational resources. Therefore, there is a trade-off between effectiveness and efficiency in the use of preprocessors [Sav94]. The solver for the vertical alignment problem in the original model applies some preprocessor by default. In this surrogate, the preprocessor is turned off in order to achieve a different balance between effectiveness and efficiency for the problem. Compared to =true, this surrogate should have the same level of accuracy but different speed. It can serve as a supplement to Surrogate 2 for testing the behavior of the solver with the use of a surrogate that has extremely high accuracy.

Surrogate 4 and 5: Skip Sections (skip2 and skip4)

Although terrain is a continuous surface, for computational practicality its shape is sampled at discrete points as the input of the vertical alignment problem. In this surrogate, some of the sample points are skipped to reduce the problem size and therefore the time to solve the problem. In Surrogate 4, one section is skipped for every two consecutive sections. In Surrogate 5, three sections are skipped for every four consecutive sections.

Surrogate 6 and 7: Early Termination (term2 and term4)

When the the vertical alignment problem is being solved, the cost function is evaluated at different points in the feasible set and the solver tracks the incumbent solution. The increase of the solution quality usually slows down as time elapses, therefore the final iterations make relatively less contribution to the overall solution quality. In this surrogate, the vertical alignment solver is terminated early when the number of iterations reaches certain proportion of the expected total number of iterations. We run the original model for 10 times before early termination is applied and use the average

number of iterations as the expected total number of iterations. For Surrogate 6 and 7 we terminate the solver at $\frac{1}{2}$ and $\frac{1}{4}$ of the expected total number of iterations.

Surrogate 8: Delete Earth Movement (noEM)

The earth movement cost accounts only for a small portion of the total cost but makes the vertical alignment problem significantly larger and more time-consuming. In this surrogate, variables and constraints related to earth movement are deleted in order to speedup the algorithm. This surrogate is implemented by deleting all the earth movement related variables, constraints and objective function terms in the original mixed-integer linear program. After deleting the earth movement cost, the costs for non-adjacent sections become independent, which allows for the use of the next two surrogates.

Surrogate 9: Dynamic Programming (DP)

This surrogate is built upon Surrogate 8. In Surrogate 8, the vertical alignment problem is still formulated as a mixed-integer linear program and the cost function is evaluated at different feasible points. However, by exploiting the cost independence between different sections, we could avoid some unnecessary evaluations by using the following dynamic programming formulation.

Assume there are S sections in total and in each section there are L levels. Let $\tilde{f}_i(v_0, v_1)$, where $2 \leq i \leq S$, be the cost of section i if the road elevations are at level v_0 and v_1 in sections $i-1$ and i. Since the start and the end of the road are fixed, only one level is needed for sections 1 and S. Thus, we make all of their levels, except for level 1, infeasible by setting

$$\tilde{f}_1^*(v_1) = \tilde{f}_S(v_0, v_1) = +\infty \text{ for all } v_1 \neq 0. \tag{4.3}$$

Let $\tilde{f}_i^*(v_1)$, where $1 \leq i \leq S$, be the optimal total cost for all the sections 1 through i if the road elevation is at level v_1 in section i. Since cost does not occur in section 1, we set

$$\tilde{f}_1^*(0) = 0. \tag{4.4}$$

Then, for all $2 \leq i \leq S$, the principle of optimality is:

$$\tilde{f}_i^*(v_1) = \min_{1 \leq v_0 \leq L} \tilde{f}_{i-1}^*(v_0) + \tilde{f}_i(v_0, v_1). \tag{4.5}$$

Because of (4.3), the optimal total cost is

$$\tilde{f}_S^*(0) = \min_{1 \le v_0 \le L} \tilde{f}_{S-1}^*(v_0) + \tilde{f}_S(v_0, 0). \tag{4.6}$$

The horizontal alignment problem only needs the minimum but not the minimizer of the vertical alignment problem. Therefore, we simply need to find the minimum by iterating through equation (4.5) without maintaining the solution table and the value table in the standard implementation of dynamic programming.

The level is formulated as a continuous variable in the original model, but needs to be discretized in this surrogate for computational tractability. After discretization, the algorithm will search for the optimum in a discrete subset of X, thus the optimality of the vertical alignment optimization problem is not guaranteed. However, it should approximate the true cost function well as a surrogate. The number of discrete levels is a parameter of this surrogate. More levels can make the surrogate more accurate but slower and vice versa. Therefore there is a trade-off in the choice of the number of levels. For testing, we set it equal to the number of vertical sample points in the input data.

Surrogate 10: Greedy Algorithm (grd)

Similar to Surrogate 9, this surrogate is also built upon Surrogate 8. Unlike Surrogate 9, which guarantees optimality in a discrete subset of X, this surrogate only tries to achive optimality between two adjacent sections. Another difference of this surrogate from Surrogate 9 is that it keeps the level as a continuous variable.

To explain this surrogate, we still use $\tilde{f}$ and $\tilde{f}^*$ to represent the cost of the current section and the total cost up to the current section, but define them differently. Specifically, let $\tilde{f}_i(v_0, v_1)$, where $2 \le i \le S$, be the cost of section i if the elevation of section $i-1$ is v_0 and the elevation of section i is v_1. If v_1 is infeasible given v_0, then $\tilde{f}_i(v_0, v_1) = +\infty$. Let $\tilde{f}_i^*$ be the total cost of sections 1 to i, then

$$\tilde{f}_i^* = \begin{cases} 0 & \text{if } i = 1, \\ \tilde{f}_{i-1}^* + \min\limits_{v_1} \tilde{f}_i({v_0}^*(i-1), v_1) & \text{otherwise,} \end{cases} \tag{4.7}$$

where

$${v_0}^*(i) = \operatorname*{argmin}_{v_1} \tilde{f}_i({v_0}^*(i-1), v_1) \text{ for } i \ge 2 \tag{4.8}$$

and ${v_0}^*(0)$ is the fixed starting point of the road as an input. We calculate the greedy optimal total cost $\widetilde{f}_S^*$ based on both the original terrain data and the reversed terrain data, which starts from the last section and move backwards. Finally, the output returned by this surrogate is the average of the two greedy optimal total costs for better accuracy.

The unconstraint version of (4.8) is a single variable convex problem [HHLR14] and its minimizer is the elevation of the terrain. The minimizer for the constraint version is either at the current terrain elevation or at one bound of the feasible interval. Thus, there are a constant number (3) of objective function evaluations in each section and the time complexity of this algorithm is $\mathbf{O}(S)$, where S is the number of sections. Whereas the time complexity of Surrogate 9 is $\mathbf{O}(SL^2)$, where L is the number of levels, because the algorithm needs to compare the costs of each pair of levels for each section. Therefore, this surrogate should be much faster than Surrogate 9.

Although modeling the level as a continuous parameter may bring some accuracy, this surrogate should still be less accurate than Surrogate 9 as it only tries to achieve optimality between two adjacent sections.

4.2 Prediction Band with Median

Definition 4.1. Given $p \in [0, 1]$, the p *prediction band* for Algorithm α is the set

$$\left\{ (y, \tau) \middle| G_\alpha^{-1}\left(\frac{1-p}{2}; \tau\right) \leq y \leq G_\alpha^{-1}\left(1 - \frac{1-p}{2}; \tau\right) \right\}.$$

Intuitively, the p prediction band is the central region where the incumbent solution is estimated to appear with probability p. The edges of the p prediction band are the quantiles with probability $\frac{1-p}{2}$ and $1 - \frac{1-p}{2}$. It characterizes the range of the incumbent solution distribution to some extent. A straightforward way of comparison is to visualize the prediction band along with the median.

In Figure 4.1, the 80% prediction band and the median are shown for two algorithms on test problem A. This method gives an intuitive comparison between two algorithms. It can be observed that =true underperforms org in median, the 10% percentile and the 90% percentile for most of the time, except for extremely small τ values.

A limitation of this approach is that it can only effectively compare two algorithms as more overlapped prediction bands would be difficult to tell apart.

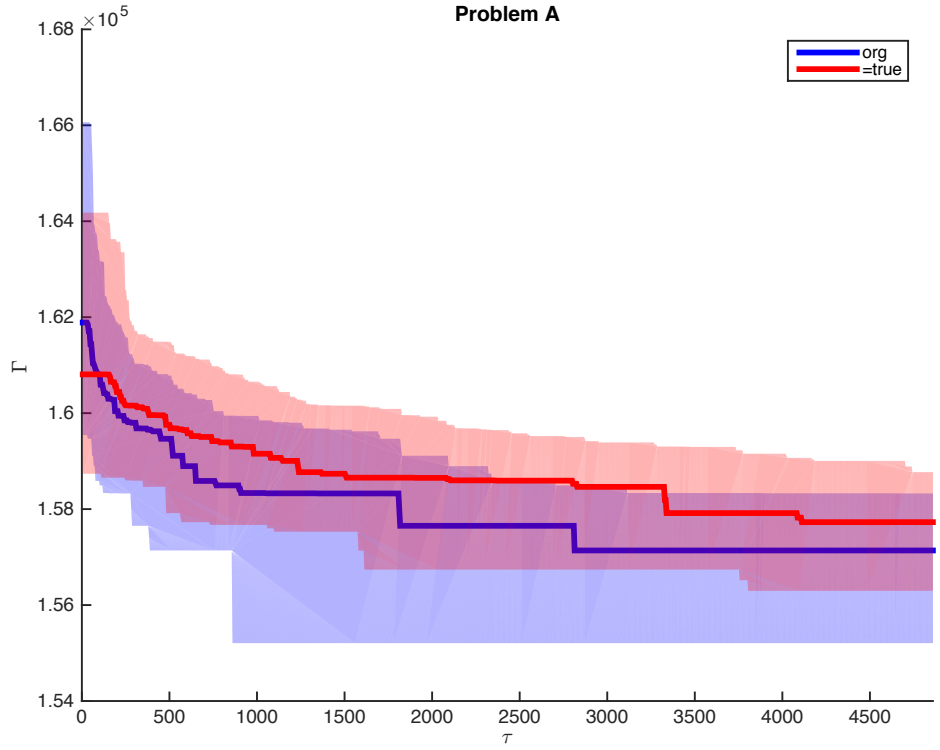


Figure 4.1: The 80% prediction band with median for two algorithms

One way to overcome this limitation is to keep the setup for the baseline algorithm and only show one quantile for all the other algorithms. In Figure 4.2, the 80% prediction band with median of the baseline are compared with the medians of other algorithms on test problem A. In terms of the median, it can be observed that only const outperforms the baseline; noEM and grd performed the worst as they stay outside the prediction band; the rest of the algorithms had similar performance and they all stay within the prediction band.

A limitation of this approach is that it can only show one quantile of all the algorithms except for the baseline. Although it allows for simultaneous comparison of multiple algorithms, limited information can be visualized.

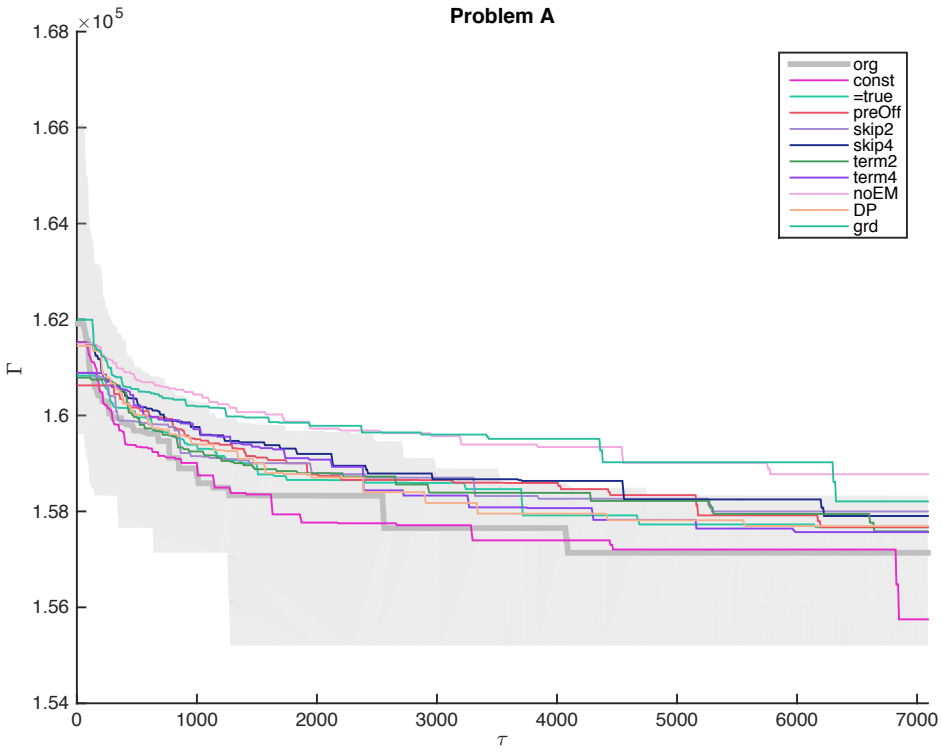


Figure 4.2: The 80% prediction band with median for ten algorithms

4.3 Integrated P-P Plot

Although the prediction band with median method can provide a straightforward comparison, it has limited capability of comparing multiple algorithms simultaneously. The method introduced in this section overcomes this limitation by generalizing the P-P plot method in statistics, which is used to compare two probability distributions by plotting their CDFs against each other.

In our case, we want to compare $G_\alpha(y;\tau)$ of different α over $\tau \in [0, \tau_{max}]$. To generate graphs similar to the P-P plot, we need to eliminate the time dimension by integrating $G_\alpha^{-1}(p;\tau)$ with respect to τ. Since $G_\alpha^{-1}(p;\tau)$ is decreasing in τ, an unweighted integration would over amplify the estimate when τ is small. In addition, different users may prefer different emphasis over $[0, \tau_{max}]$, thus an unweighted integration does not suffice. Let $w(\tau) : [0, \tau_{max}] \mapsto \mathbb{R}^+$ be the weight imposed by the user to $G_\alpha^{-1}(p;\tau)$ at τ for all α, we integrate the quantiles as follows:

$$G_\alpha^{-1}(p) = \int_0^{\tau_{max}} w(\tau) G_\alpha^{-1}(p;\tau) d\tau. \tag{4.9}$$

Although there could be various possible candidates for $w(\tau)$, here we suggest

$$w(\tau) = \frac{1}{G_0^{-1}(0.5;\tau)}. \tag{4.10}$$

This formulation of $w(\tau)$ is designed to impose similar levels of emphasis over $[0, \tau_{max}]$ by balancing the effect of decreasing $G_\alpha^{-1}(p;\tau)$ in τ. Since all the algorithms are solving the same problem, the shape of $G_\alpha^{-1}(p;\tau)$ should be similar for all α and formula (4.10) should present reasonable weighting functions not only for algorithm 0 but also for all other algorithms.

We define the integrated CDF as

$$G_\alpha(y) = \mathbb{P}(G_\alpha^{-1}(p) \leq y). \tag{4.11}$$

By choosing one of the algorithms, which in our case is algorithm 0, as the baseline, we can simultaneously show the performance of different algorithms all against the baseline in an integrated P-P plot. For each alternative algorithm α and for each p of interest we plot

$$G_0(G_\alpha^{-1}(p)), \tag{4.12}$$

which indicates the probability for algorithm 0 to return an output as good as algorithm α's best $100p\%$ output.

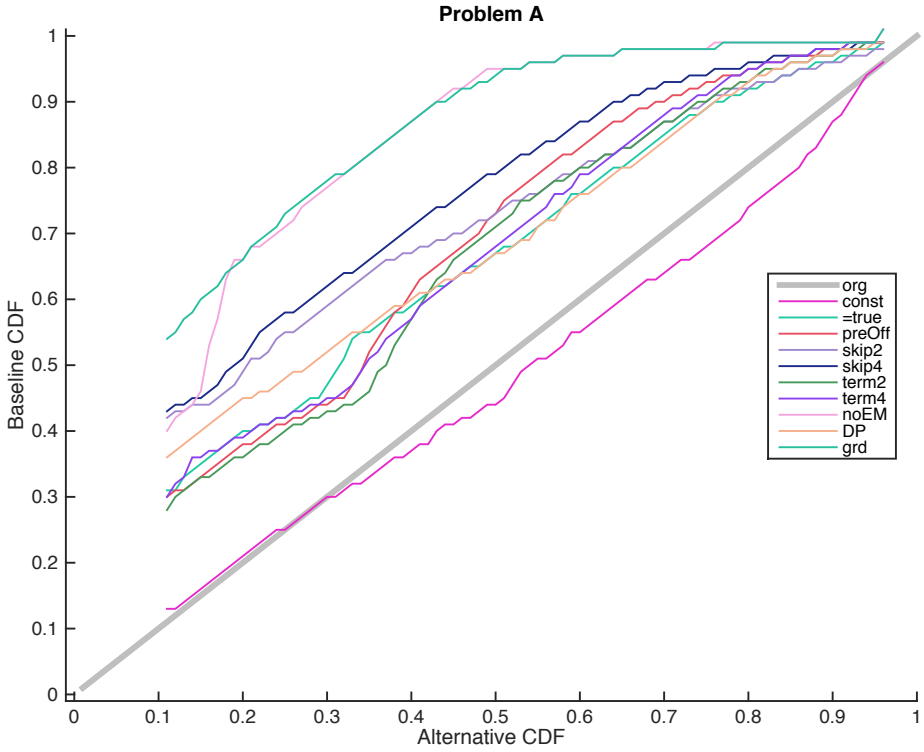


Figure 4.3: Integrated P-P plot

Considering an algorithm $\alpha \neq 0$ for a minimization problem, small values in the P-P plot are favorable to α, because it means the baseline has a low chance of returning outputs as good as the output by α. The baseline's integrated CDF is also plotted against itself as a reference line, which is the diagonal. If the P-P plot of an algorithm stays entirely below the reference line, then its empirical solution distribution stochastically dominates the baseline algorithm, indicating its performance is better than the baseline, and vice versa.

Figure 4.3 is a demonstration of the integrated P-P plot with our test data for test problem A. It can be observed that `const` outperforms the baseline algorithm in most of the probability range, expect for the best cases. All other algorithms are dominated by the baseline algorithm.

4.4 Speed Quantification

The previous two comparison methods are both visualizations, and do not disclose how much speedup or slowdown an algorithm has against another. In this section, we introduce a method to quantify the relative speed of multiple algorithms. When we compare algorithm α versus algorithm 0, the method in general is to find $\lambda \in \mathbb{R}^+$ such that $G_0^{-1}(p; \lambda\tau)$ best matches $G_\alpha^{-1}(p; \tau)$. The value of λ can be interpreted as the relative speed of algorithm α versus algorithm 0. Mathematically, λ is defined as a solution to the following least squares integration problem:

$$\lambda = \underset{\lambda' \in \mathbb{R}^+}{\operatorname{argmin}} \left\{ \frac{\max\{\lambda', 1\}}{\tau_{max}} \int_0^{\frac{\tau_{max}}{\max\{\lambda', 1\}}} \left[G_0^{-1}(p; \lambda'\tau) - G_\alpha^{-1}(p; \tau) \right]^2 d\tau \right\}, \tag{4.13}$$

where p is the probability corresponding to the quantile of interest. The upper bound of the integral is designed in order to ensure that all the data stays within the reliable time interval as λ' changes. Specifically, let m be the upper bound of the integral, then solving (4.13) requires the value of $G_0^{-1}(p; \tau)$ and $G_\alpha^{-1}(p; \tau)$ for $0 \leq \tau \leq \lambda' m$ and $0 \leq \tau \leq m$ respectively. To ensure data reliability, we need

$$\max\{\lambda' m, m\} \leq \tau_{max}. \tag{4.14}$$

Solving this inequality, we have

$$m = \frac{\tau_{max}}{\max\{\lambda', 1\}}. \tag{4.15}$$

The term $\frac{\max\{\lambda', 1\}}{\tau_{max}} = \frac{1}{m}$ before the integral normalizes the size of the integral, hence it is not affected by λ' directly but through the effect of scaling G_0^{-1}.

Since all the algorithms are used to solve the same problem, the incumbent solution paths should have similar shapes. As a result, this method should closely match the scaled incumbent solution process of algorithm 0 with the incumbent solution process of algorithm α.

When a group of algorithms is tested over multiple problems, the harmonic mean for $\lambda_1, \lambda_2, \ldots, \lambda_n$,

$$\bar{\lambda} = \frac{n}{\sum_{i=1}^n \frac{1}{\lambda_i}}, \tag{4.16}$$

is an appropriate method to calculate the average speedup.

To see this, consider the case where there are n problems that require the same amount of computation c and the speed of algorithm α_i is λ_i for all $i = 1, 2, \ldots, n$. The harmonic mean calculates the speed $\bar{\lambda}$ of an algorithm α^* that solves all the problem from the same problem set in the same amount of total time as $\alpha_1, \alpha_2, \ldots, \alpha_n$ do. In particular, to equate the total times, we have

$$n\frac{c}{\bar{\lambda}} = \sum_{i=1}^{n} \frac{c}{\lambda_i}, \tag{4.17}$$

which leads to formula (4.16) by solving for $\bar{\lambda}$.

For demonstration, we set $p = 0.5$ for the calculations of λ, i.e., we match the median of different algorithms to that of the baseline algorithm. The integral in (4.13) is evaluated numerically by equally discretizing $[0, \tau_{max}]$ into 1000 segments. Table 4.1 shows the speed quantifications for this setup. All the values are solved using the fmincon function in MATLAB with initial solution 1. According to the average relative speed, only =true and preOff outperforms the baseline algorithm.

For advanced use of this method, problem (4.13) can be generalized to:

$$\lambda = \underset{\lambda' \in \mathbb{R}^+}{\operatorname{argmin}} \sum_{i=1}^{Q} q_i \frac{\max\{\lambda', 1\}}{\tau_{max}} \int_0^{\frac{\tau_{max}}{\max\{\lambda', 1\}}} \left[G_0^{-1}(p_i; \lambda'\tau) - G_\alpha^{-1}(p_i; \tau)\right]^2 d\tau, \tag{4.18}$$

where Q is the total number of the quantiles to focus on, $p_1, p_2, \ldots, p_Q$ are their corresponding probabilities and $q_1, q_2, \ldots, q_Q$ are their corresponding weights.

There are many ways to specify Q depending on the user's need and the reliability of the quantiles, which is discussed in Section 3.3. For example, assuming all the quantiles with $10\% \leq p \leq 95\%$ are reliable, then by choosing

$$Q = 81,$$

and

$$p_i = (i + 9)\%, q_i = \frac{1}{Q} \text{ for all } 1 \leq i \leq Q$$

the user imposes equal emphasis on each reliable percentile. This weight distribution is more reasonable if the algorithm is used for a large number of times in practice, because the overall outcome is closely related to the average performance by the "Law of Large Numbers". However, if the algorithm is used only a few times, robustness becomes more important. The

setup

$$Q = 1,\ p_1 = 0.95 \text{ and } q_1 = 1$$

can be used when the user only concerns about the 5% value-at-risk. Alternatively,

$$Q = 2,\ p_1 = 0.5,\ p_2 = 0.95 \text{ and } \ q_1 = q_2 = 0.5 \tag{4.19}$$

can be used when the user takes both the median performance and the robustness into consideration with equal emphasis.

For demonstration, we apply the advanced framework based on the configuration in (4.19). Table 4.2 shows the speed quantifications for this setup. The discretization and optimization are conducted the same way as those for Table 4.1. The results in Table 4.2 are similar to those of Table 4.1, indicating there is no significant difference in speedup between the 50^{th} and the 90^{th} percentiles. Although only two of the algorithms outperforms the baseline algorithm, all of them performed robustly when the 5% worst-case scenario is taken into consideration.

Table 4.1: Speed quantification by 50^{th} percentile

Speed		Test Problem										
Algorithm	Average	A	B	C	D	E	F	G	H	I	J
const	0.65	1.04	3.52	0.52	0.45	0.29	2.02	0.48	0.56	0.53	3.10
=true	1.43	0.63	5.33	2.35	1.96	1.28	13.61	0.46	2.80	1.38	5.93
preOff	1.30	0.50	4.73	1.50	1.34	1.00	12.43	1.08	0.88	1.37	6.44
skip2	0.85	0.62	2.28	2.26	0.26	4.93	10.92	0.27	2.69	1.47	3.67
skip4	0.69	0.51	1.76	0.36	0.31	1.32	10.67	0.30	1.70	1.07	4.96
term2	0.52	0.49	2.16	0.23	0.60	0.19	7.98	0.31	3.95	0.89	1.75
term4	0.45	0.54	0.98	0.20	0.28	0.19	8.10	0.30	1.97	0.91	4.06
noEM	0.91	0.33	6.96	2.60	0.31	1.59	11.67	0.42	3.89	1.62	5.11
DP	0.60	0.58	3.05	0.34	0.56	0.18	4.35	0.51	6.84	0.60	3.07
grd	0.73	0.34	4.46	0.52	0.47	0.39	8.47	0.57	1.68	0.90	2.51

Table 4.2: Advanced speed quantification by 50^{th} and 90^{th} percentiles

Algorithm	Speed	Test Problem										
		Average	A	B	C	D	E	F	G	H	I	J
	const	0.62	1.15	3.68	0.52	0.41	0.28	2.01	0.44	0.54	0.51	3.10
	=true	1.42	0.61	5.41	2.33	1.98	1.31	14.84	0.45	2.76	1.37	5.93
	preOff	1.33	0.50	5.22	1.70	1.33	1.00	12.53	1.09	0.95	1.29	6.72
	skip2	0.80	0.61	2.44	2.37	0.23	4.79	10.74	0.25	2.79	1.47	3.63
	skip4	0.65	0.50	1.84	0.31	0.30	1.32	11.13	0.28	1.70	1.05	5.06
	term2	0.50	0.56	2.22	0.21	0.59	0.18	8.57	0.29	3.83	0.87	1.73
	term4	0.43	0.55	0.96	0.18	0.25	0.19	8.18	0.28	1.95	0.90	4.14
	noEM	0.88	0.33	6.86	2.57	0.29	1.69	12.73	0.40	3.81	1.60	5.21
	DP	0.57	0.59	2.87	0.33	0.51	0.17	4.37	0.49	6.90	0.56	3.14
	grd	0.69	0.35	4.62	0.51	0.41	0.38	8.98	0.51	1.74	0.82	2.52

Chapter 5

Conclusion

5.1 Contribution

In this thesis, a methodology is proposed to compare the performance of stochastic optimization algorithms through bootstrapping of test data. The incumbent solution of a problem over multiple restarts is modeled as a stochastic process. A bootstrap estimator of the incumbent solution process given limited amount of test data is designed.

Then, various properties of the estimator are analyzed. From the theoretical aspect, the estimator is shown to be asymptotically consistent. Under some assumptions, the estimator is shown to have a non-positive bias. From the numerical aspect, an out-of-sample test is conducted to demonstrate the reliability of the estimator when different parameters are varied. We find that although the estimator tends to underestimate the probability distribution function of the incumbent solution process, it can provide reliable prediction when the time interval and quantiles to focus on is properly chosen.

Finally, three methods to compare the performance of different stochastic optimization algorithms are proposed and demonstrated with test data from a road design optimization problem. Different methods have different balances between showing more details on a single algorithm and simultaneously comparing more algorithms. The results are consistent across all methods. Ten surrogate cost functions for the road design optimization problem are designed and tested. Although two of the algorithms slightly outperforms the baseline, none of them is considered as a significantly better algorithm. A software package for the use of the comparison methods is available online[4].

5.2 Future Work

Several directions for future work present themselves.

[4] `https://www.dropbox.com/sh/sq4xhz51qrkibwe/AACCtNF3HT9A-hZ_8LQlWXE0a?dl=0`

First, the estimator might be improved through bias correction. Since the incumbent solution is the minimum of a collection of solutions, its distribution highly depends on the left tail of the individual solutions' distribution. However, the empirical CDF has a limited ability to estimate the tails of the true distribution. The bias could be suitably corrected by better estimating the left tail of the individual solutions' distribution. Smoothing techniques [Sim96] or the parametric bootstrap method [Che07] could be explored for bias correction.

Second, it is important to investigate whether Assumption 2.1 can be relaxed. According to Andersen et al. [ABGK93], the bootstrap estimator for the counting process is unbiased for the first and second order moments, but it still remains to verify whether it is also unbiased for the whole distribution.

In addition, the uniqueness of the speed quantification λ as defined in equation (4.13) needs to be investigated. Figure 5.1 shows the normalized objective function values of the least-square integration problem in equation (4.13) for $\lambda \in [0.01, 100]$. For easy comparison, the function values are shifted and scaled to span $[0, 1]$. According to Figure 5.1, it seems that the least squares integration problem regarding λ is convex near 1 and the global minimizer can be found by searching from $\lambda = 1$. A rigorous proof is still required to formally address the issue.

Another direction to which this thesis could be extended is to analyze the solution time distribution over different objective function values. Since both speed and solution quality of algorithms are analyzed, there are two variables: time and function value. By defining the incumbent solution process, we treat time as the independent variable and function value as the dependent variable. It would be interesting to explore the alternative setting, in which the distribution of the time for an algorithm to reach different values of the objective function is studied. However, a challenge is data inconsistency across different values of the objective function. For the methodology in this thesis, any finite time horizon can be covered by a bootstrap path since time accumulates as multiple restarts are drawn from the sample. However, certain range of function values may not be able to be covered by all the algorithms being compared as some of them never reach certain function values.

Researchers could also explore how the resampling method might be generalized to the case of non-independent restarts. One possible case is random sampling without replacement. By using bootstrap, i.e., resampling with replacement, we consequently assumed the user also applies multiple restarts with replacement. However, once a particular restart has been em-

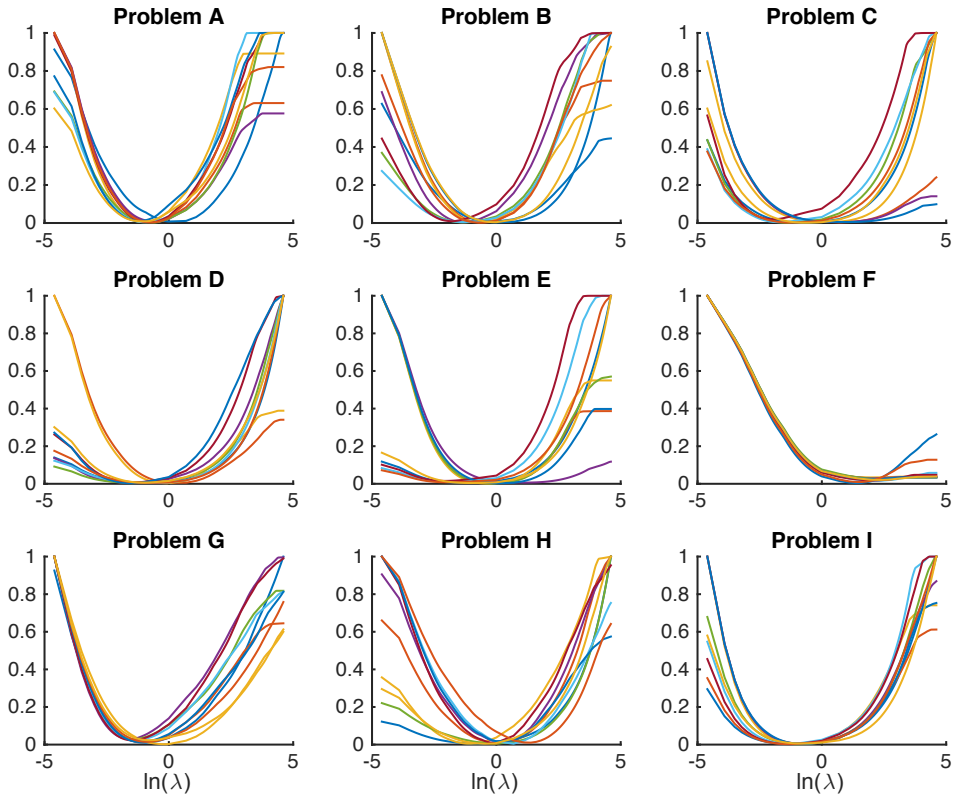


Figure 5.1: Uniqueness of λ

ployed, rerunning it will not increase the solution quality but will decrease the overall speed of the optimization process. Therefore, it would be more reasonable to assume the user resamples without replacement and to model the incumbent solution process accordingly. Another group of cases is when adaptive sampling methods, such as those in the scatter search [Glo98] or the genetic algorithm, are applied in multiple restarts. The incumbent solution process needs to be modified to model the incumbent solution when certain adaptive sampling methods are used.

Moreover, more work can be done to study the use of surrogate cost functions in the road design optimization problem. Based on the test results, none of the surrogates significantly outperforms the original model. The absence of improvement may be due to the way NOMAD uses the surrogate cost functions and the way the surrogates are designed. NOMAD assumes the surrogate cost function is close to free but not necessarily gives an accurate output. As a result, the surrogate cost function is evaluated in numerous points before the true cost function is evaluated. The surrogates designed in this thesis may be too expensive for NOMAD to effectively utilize. One way to overcome this issue is to design surrogates that cooperate better with NOMAD's surrogate usage approach. A more promising way is to develop a new framework for black-box optimization algorithms to use the type of surrogates designed in this thesis. A good starting point can be found in [TY11].

Finally, the framework proposed in this thesis might be extended to analyze the speedup from parallelization of stochastic optimization algorithms. Emphasis should be put on the relationship between solution quality, speed and number of processing units.

Bibliography

[AAC+] M. A. Abramson, C. Audet, G. Couture, J. E. Dennis, Jr., S. Le Digabel, and C. Tribes. The NOMAD project. Software available at `https://www.gerad.ca/nomad/`. → pages 6, 20

[ABGK93] P. K. Andersen, O. Borgan, R. D. Gill, and N. Keiding. *Statistical models based on counting processes (ISBN 0387978720)*. Springer-Verlag Inc, Berlin; New York, 1993. → pages 46

[ABS05] A. Akay, K. Boston, and J. Sessions. The evolution of computer-aided road design systems. *International Journal of Forest Engineering*, 16(2), 2005. → pages 3

[AD06] C. Audet and J.E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006. → pages 20

[ALT09] C. Audet, S. Le Digabel, and C. Tribes. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD, 2009. → pages 7

[Bab79] L. Babai. Monte-Carlo algorithms in graph isomorphism testing. Technical Report, Université de Montréal, D.M.S. No. 79-10, 1979. → pages 2

[BDF+99] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13, 1999. → pages 6, 7

[BF81] P. J. Bickel and D. A. Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6):1196–1217, 11 1981. → pages 15

[BHHR14] A. Butler, R. D. Haynes, T. D. Humphries, and P. Ranjan. Efficient optimization of the likelihood function in gaussian process

modelling. *Computational Statistics & Data Analysis*, 73:40–52, 5 2014. → pages 2

[Can08] Transport Canada. Transportation in Canada: an overview 2008. Technical Report 2009-07612., Minister of Public Works and Government Services, Canada, 2008. → pages 3

[Che07] M. R. Chernick. *Bootstrap Methods: A Guide for Practitioners and Researchers, Second Edition*. John Wiley & Sons, Inc., 2007. → pages 8, 46

[CSV09] A. Conn, K. Scheinberg, and L. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009. → pages 4

[CW12] J. Currie and D.I. Wilson. OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In N. Sahinidis and J. Pinto, editors, *Foundations of Computer-Aided Process Operations*, Savannah, Georgia, USA, 8–11 January 2012. → pages 20

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. → pages 1, 8

[DN05] H. A. David and H. N. Nagaraja. *Order Statistics, Third Edition*. John Wiley & Sons, Inc., 2005. → pages 16

[Don10] D. Donaldson. Railroads of the raj: Estimating the impact of transportation infrastructure. Working Paper 16487, National Bureau of Economic Research, October 2010. → pages 3

[ET93] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993. → pages 8

[FRK+08] K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis Jr., C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray, and T. G. Kolda. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 5 2008. → pages 1

[FRS94] T. A. Feo, M. G. C. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 2017/02/20 1994. → pages 1, 8

[Glo98] F. Glover. *A template for scatter search and path relinking*, pages 1–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. → pages 48

[Gri78] V. A. Grishagin. Operating characteristics of some global search algorithms. *Problems of Stochastic Search*, 7, 1978. → pages 1, 8

[HEAEH98] Y. Hassan, S. Easa, and A. Abd El Halim. Highway alignment: Three-dimensional problem and three-dimensional solution. *Transportation Research Record: Journal of the Transportation Research Board*, 1612:17–25, 1998. → pages 3

[HHLR14] W. Hare, S. Hossain, Y. Lucet, and F. Rahman. Models and strategies for efficiently determining an optimal vertical alignment of roads. *Computers & Operations Research*, 44:161 – 173, 2014. → pages 3, 5, 6, 35

[HJK96] C. R. Houck, J. A. Joines, and M. G. Kay. Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems. *Computers & Operations Research*, 23(6):587–596, 1996. → pages 2

[HS98] H. H. Hoos and T. Stützle. Evaluating las vegas algorithms: Pitfalls and remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 238–245, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. → pages 1, 8

[JP03] J. Jacod and P. E. Protter. *Probability Essentials*. Springer, 2nd edition, 2003. → pages 12

[Kan08] M. W. Kang. An alignment optimization model for a simple highway network, 2008. → pages 5

[KG09] L. Kocsis and A. György. *Efficient Multi-start Strategies for Local Search Algorithms*, pages 705–720. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. → pages 2

[KGB13] S. R. Kuindersma, R. A. Grupen, and A. G. Barto. Variable risk control via stochastic optimization. *The International Journal of Robotics Research*, 32(7):806–825, 2017/03/29 2013. → pages 2

[Kie53] J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953. → pages 11

[MLH15] S. Mondal, Y. Lucet, and W. Hare. Optimizing horizontal alignment of roads in a specified corridor. *Computers & Operations Research*, 64:130 – 138, 2015. → pages 3, 31

[Mon14] S. Mondal. Horizontal alignment optimization in road design. Master's thesis, University of British Columbia, Jul 2014. → pages 4, 5

[MW09] J. Moré and S. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2017/02/20 2009. → pages 1, 8

[Ort12] G. A. Ortiz. Evolution strategies. http://www.mathworks.com/matlabcentral/fileexchange/35801-evolution-strategies–es-, May 2012 [cited Nov. 2012]. → pages 19

[RCS06] J. Rodrigue, C. Comtois, and B. Slack. *The geography of transport systems*. Routledge, 2006. → pages 3

[Sav94] M. W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6(4):445–454, 2017/03/09 1994. → pages 32

[Sim96] J. S. Simonoff. *Smoothing Methods in Statistics*. Springer New York, New York, 1 edition, 1996. → pages 46

[SKM16] Y. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms. *Mathematics and Computers in Simulation*, 2016. → pages 1

[SMG13] S. U. Stich, C. L. Müller, and B. Gärtner. Optimization of convex functions with random pursuit. *SIAM Journal on Optimization*, 23(2):1284–1309, 2013. → pages 11

[Spa03] J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley-Interscience [Imprint], 2003. → pages 2

[TRC12] C. Truchet, F. Richoux, and P. Codognet. Prediction of parallel speed-ups for las vegas algorithms. *CoRR*, abs/1212.4287, 2012. → pages 2

[TY11] J. Takaki and N. Yamashita. A derivative-free trust-region algorithm for unconstrained optimization with controlled error. *Numerical Algebra, Control and Optimization*, 1(1):117–145, 2011. → pages 48

[VDHL17] K. K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti. Surrogateâbased methods for blackâbox optimization. *International Transactions in Operational Research*, 24(3):393–424, 2017. → pages 6

[VWD16] P. Vasant, G. Weber, and V Dieu. Handbook of research on modern optimization algorithms and applications in engineering and economics, 2016. → pages 1