BAYESIAN RANKING AND SELECTION MODELS FOR DISCRETE NETWORK DESIGN PROBLEMS WITH UNCERTAINTIES AND MULTIPLE ENVIRONMENTAL OBJECTIVES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Xun Wang

January 2014

BAYESIAN RANKING AND SELECTION MODELS FOR DISCRETE NETWORK DESIGN PROBLEMS WITH UNCERTAINTIES AND MULTIPLE ENVIRONMENTAL OBJECTIVES

Xun Wang, Ph.D.

Cornell University 2014

In this dissertation we develop a comprehensive Bayesian Ranking and Selection (R&S) modeling framework for single and multi-objective Network Design Problem with Uncertainty (NDPU). NPDU is a classical problem in transportation sciences and engineering. Due to the complex bi-level nature, NDPU is usually solved with heuristic algorithms where the objective value of many candidate solutions are "simulated" for evaluation. As the size of the transportation network can be large, the evaluation of objective values often become the computational bottleneck and should be kept to minimum numbers. On the other hand, most current formulations for (NDPU) characterize uncertainty as a discrete scenario set and tend not to fully explore the inherent correlations among alternatives. Therefore, we feel there is room for improving the efficiency of NDPU solution algorithms with a more rigorous statistical learning model.

In Chapter 2, we formulate the NDPU problem as a Constrained Bayesian Ranking and Selection ($R\&S$) problem with exact correlated beliefs. In this formulation, each solution to the NDPU problem represents an "alternative" and the corresponding objective value represents a "reward" we want to maximize. Uncertainties in the objective values are modeled by normal distributions of the rewards and constraints of the NDPU problem are utilized for pre-eliminating infeasible solutions. At each sampling iteration, we update our belief about the

distribution of all alternative performances and use the cumulative sampling history to make the next sampling decision. We use a customized version of the Knowledge Gradient policy with Correlated Beliefs (KGCB) to account for constraints and unknown variances of the rewards. Case studies are conducted on transportation networks of different sizes, using popular heuristics such as Genetic Algorithm and Simulation Annealing as comparisons. Results show that the Bayesian $R\&S$ model generally provide better accuracy and convergence rate, particularly in scenarios with uncertainty and larger networks.

In Chapter 3, we build upon our model Bayesian R&S model in Chapter 2 to improve its performance under large number of projects/alternatives. The new model features 1) a recursively updated linear approximation of the upper-level objective function using Gaussian-binary basis functions, and 2) A surrogate-assisted knowledge gradient sampling policy which utilizes the optimal solution of the approximated surrogate objective function to constraint the scale of the expensive knowledge gradient calculation. With the two features the computational complexity of our algorithm is reduced to only a low degree (typically ≤ 2) polynomial of the number of projects. Case studies are conducted on the Sioux Fall network and Anaheim network with as many as 20 projects and over 1,000,000 possible network configurations. Results showed that this parametric Bayesian $R\&S$ model is able to identify highly optimal solutions in only around 100 iterations, significantly outpacing our bench-marking Genetic Algorithm and Simulated Annealing Algorithm in both convergence speed and computational cost. Our new method provides a highly scalable framework for discrete NDPU without sacrificing much of the performance advantage of Bayesian R&S models. It also extends the Bayesian R&S model and the knowledge gradient sampling policies to generic large-scale discrete optimization problems, which provides valuable insights for a large class of similar optimization and learning problems.

In Chapter 4, we further extend the Bayesian R&S model to the Multi-Objective discrete Network Design Problem with Uncertainty (MONDPU), an emerging area in transportation planning due to the need for sustainable transportation systems. In this formulation, we put independent parametric beliefs on the expected reward of each objective function like we did in Chapter 3 and update them in parallel through sequential samples. We define a multi-objective version of the Knowledge Gradient policy with Correlated Beliefs which use a crowding distance metric to ensure the diversity of the Pareto optimal front. Case studies are conducted on the Sioux Fall network and Anaheim network. Results showed that our multi-objective Bayesian $R\&S$ model is able to identify a very diverse set of highly optimal solutions under very limited budget, significantly out-performing the bench-marking NSGA-II algorithm in both solution quality and practicality. Our model is also the first to extend the Bayesian R&S model and the knowledge gradient sampling policies to generic multi-objective problems.

In summary, the Bayesian $R\&S$ formulation is well-suited for NDPU and MONDPU due to its uncertainty management capabilities and the sampling efficiency of knowledge-gradient related policies. The models provide an innovative statistical learning perspective to NDPU, which has mainly been studied as an optimization problem. The new formulation is intuitive to understand and easily applicable to similar discrete optimization problems such as the Optimal Sensor Location problem, Uncapcitated Fixed Charge Facility Location problem, etc. The global Bayesian belief structure and the sequential value-of-information sampling policies make the model especially efficient for black-box, gradient free optimization problems where the evaluation of each objective value take up the majority of the computational burden. We believe the models themselves as well as this unique statistical perspective is of great interest and value for transportation network modelers and simulation optimization practitioners.

BIOGRAPHICAL SKETCH

Xun Wang was born in June 5th, 1986 in Nanjing, Jiangsu, China. He attended Nanjing Foreign Language School in high school and obtained his Bachelor of Science degree in Environmental Sciences at Nanjing University in 2008. Xun came to Cornell University in August 2008 to pursue his MS/PhD degree at the School of Civil and Environmental Engineering under the advice of Professor Oliver Gao. After graduation, Xun will work as a quantitative modeler for Sentrana Inc. in Washington D.C.

Xun has a wide range of interests from mathematics, computers, medicine, psychology to music, literature, history, philosophy, politics, religion and so on. In his spare time, Xun enjoys reading books, playing the guitar, playing basketball, jogging and sleeping.

This dissertation is dedicated to the memory of my grandfather who passed away in 2012. Grandpa's firm belief in education culminated in the recipient of my PhD degree. I also dedicate this dissertation to my parents, whose continuing love and support fueled the entire course of my graduate study .

ACKNOWLEDGEMENTS

I'd like to first thank my parents for all the support they gave me in the past 27 years. They worked very hard to provide me with a care-free childhood during which I was able to focus exclusively on self-development. They also served as great role models of honesty, integrity, persistence and commitment, which all became solid foundations for my characters. I also feel very fortunate to have met and spent my PhD years with my girlfriend Michelle Zhu, who has brought so much happiness, courage and compassion into my life.

The person to which I owe the most gratitude throughout my study at Cornell is without any doubt my adviser Prof. Oliver Gao. Oliver is a truly inspiring mentor who always puts the comprehensive development of his students at first priority. He was always patient during my ups and downs in research and was always remarkably open-minded to accommodate whatever skill-set or research topic I am passionate about. All the academic endeavors he laid out for me over the past 5 years have turned into invaluable life lessons that have and will benefit both the academic and the non-academic aspects of my life.

I also like to thank my special committee members: Professor Giles Hooker, Professor Peter Frazier and Professor Mark Turnquist. Giles has the deepest knowledge about statistics and have been providing insightful comments about my research since my master's thesis. Professor Frazier is one of the most versatile operations research scholar and one of the nicest person I have met, whose research work eventually became a major inspiration for my dissertation. Professor Turnquist, with his over 30 years of experience in transportation systems research and academic advisory, is the ultimate reference I can turn to whenever I have any question about transportation modeling, optimization, simulation, teaching and even course selection. Their expertise and guidance has really brought insightful and constructive perspectives into my research.

Next on my list of acknowledgments are all the Cornell professors I interacted with during my graduate study. I feel very fortunate to have been instructed and inspired by some of the brightest minds in the academic field. My special thanks goes to Prof. Linda Nozick, who has provided countless suggestions to my research, Prof. Thomas O'rouke, Prof. Mark Turnquist (again) and Dr. Francis Vanek, who have greatly sharpened my teaching skills, and Prof. Sidney Resnick, from whose self-explanatory lectures I was able to pick up the fundamentals of advanced probability and statistics with almost zero background.

Finally, my thanks go to my friends and colleagues at Cornell: Shiyao Chen, Guozhang Wang, Haden Lee, Timon Stasko, Darrell Sonntag, Binyan Huang, Xi He, Chen Wang and numerous others. I will miss the times we spend together cracking research problems, tackling projects, discussing interesting life questions, playing sports, and having fun. You people made my time at Cornell more colorful and enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this dissertation we develop a Bayesian Ranking and Selection modeling framework for single and multiple objective distrete Network Design Problems with Uncertainty (NDPU). NDPU is a classical problem in combinatorial optimization with numerous applications in transportation sciences and engineering. The generic idea of improving system performance through a few design variables is also widely applied in other areas including water resource management [25], communication networks [56], sensor network design [10], prepositioning of emergency supplies [63], drug discovery [52] and etc. Given a network and its users (demand), NDPU is formulated to choose among a set of “projects” (modifications to the network) to optimize system-wide objective(s) over the network, subject to uncertainty in travel demand, travel cost, etc. Examples of those objectives include total travel time, system capacity [79] and in recent studies environmental impact or total energy consumption [26] of the network. The intricacy of NDP/NDPU lies in the fact that each “project” not only affects the structure of the network, but also alters the way users utilize the network once they are aware of the changes. To capture such interactions, NDP/NDPU are typically formulated as Bi-level programs [23], which are known to be NP-hard even in their simplest forms [87].

1.1 (Single Objective) NDPU

The most common formulation of NDPU involves a single upper level objective function, while the lower level problem is typically modeled as a deterministic

User Equilibrium traffic assignment problem [75]. Research in NDPU started with its deterministic version, the NDP problem. Earlier solution methods for NDP tended to treat or convert the upper and lower level as an integrated optimization problem. Due to the proven computationally complexity, these studies focused mainly on relaxing various aspects of the problem (e.g. objective functions [58], constraints [18], lower-level problems [18], etc.). Exact solutions were only available in special cases with nice properties, e.g. when the volume-delay function is assumed to be constant [7]. A good review of these methods can be found in [93] and [59]. In recent years, iterative meta-heuristic algorithms have been another popular branch of methodology for solving NDP. Applications of such heuristic algorithms include Genetic Algorithms (GA) [94], Simulated Annealing (SA) [32], Tabu Search [9], Ant Systems [57], etc. Meta-heuristic algorithms treat the lower level problem as a computational component of the upper-level objective function. They usually start with a few arbitrary feasible solutions and move towards better ones via local or stochastic search rules inspired by physical, biological, evolutionary processes, etc. As the computational cost of lower level problems reduces dramatically due to advances in both algorithm design and computing power, heuristic methods is becoming increasingly practical for real-world problems. One can usually find reasonably optimal solutions with meta-heuristic methods. However, the convergence rate of those methods are usually very low [48, 28, 38, 17, 69] in finite iterations.

Many NDPU solution algorithms are derived from their NDP counterparts. Most formulations characterize uncertainty as probability distributions over a finite domain/scenario set (e.g. [79]). If the scenario set is small, one can afford to convert the NDPU into a larger NDP with multiple lower level problems [79, 72]. When the scenario set becomes large or infinite (e.g. governed by continuous probability distributions), scenario approximation methods such as Single Point Approximation, Monte Carlo sampling, and Sample Average Approximation are needed. In Single Point Approximation, one tries to construct an input for which the optimal solution closely matches the true optimal solution [84, 82]. In Monte Carlo sampling, one generates repeated samples for a particular candidate solution to estimate its true expected objective value [73]. More sophisticated sampling scheme like Randomized Quasi-Monte Carlo Sampling or Antithetic Sampling can also be used to improve sampling efficiency [73]. In Sample Average Approximation (SAA) [72], the NDPU problem is solved repeatedly under a smaller scenario set sampled from the full scenario set and the best solution generated in this process is used to approximate the true optimal solution [55]. With the exception of single point approximation, all the aforementioned methods require simulating hundreds to thousands of candidate solutions during the computational process. Meanwhile, NDPU in practice can be imposed on large networks with tens of thousands of links, making each evaluation of the objective function computationally expensive. In extreme cases where traffic demand is modeled by activity-based travel demand models such as NYMTC (New York Metropolitan Transportation Council)'s NYBPM [53], even one user equilibrium run can take hours. Therefore, there is strong motivation to reduce the number of samples/simulations as much as possible.

Although the mathematical formulation of NDPU is explicit, the upper level objective function is an implicit function of the lower level problem and thus has no close form in terms of the upper-level decision variables. From this perspective, we can also view NDPU as a simulation optimization problem [44] where the lower level problem is the "simulator" and the upper level problem is a black-box objective function. Simulation optimization methods were rarely applied in NDP/NDPU, but have had numerous applications in other areas such as engineering design, water resources management, modeling calibration, etc. Popular methods in this category include EGO [40], Sequential Kridging [24], Response Surface Modeling [44], Radius basis functions [64, 51, 39], Splines [70], Gaussian Process regression [62], etc. As the exact form of the expensive objective function is usually very difficult to specify, many of the aforementioned methods use a parametric approximation (most commonly a linear regression model) to characterize the shape of the black-box objective function. Unlike meta heuristic methods, simulation optimization algorithms are usually designed with the computational cost of the objective function in mind and aim to exploit as much information as possible in each evaluation of objective value. Indeed, most of the aforementioned algorithms have reported to be able to identify highly optimal solutions with relatively few iterations. However, many of them are designed primarily for deterministic optimization problems and cannot be naturally extended to handle sampling noises or complex uncertainty structures.

1.2 Multi-Objective NDPU

The efficiency of transportation networks used to be measured primarily by congestion-related metrics such as total travel time, demand/capacity ratio, maximum capacity and etc. In recent years, there has been increased emphasis on the externality (e.g. environmental impact, energy consumption) of transportation systems and thus attempts to design sustainable transportation systems [26, 2, 6, 95, 35]. As a result, environmental/energy objectives are now often considered *in parallel* with congestion management goals, making

NDP/NDPU inherently multi-objective. As objectives in MONDP/MONDPU sometimes conflict with each other (e.g. system capacity vs. financial budget), the optimal solution of MONDP/MONDPU is typically not a single point but a set of solutions with non-dominated objective values know as Pareto Optimal set [8]. Due to the non-uniqueness of optimal solutions, population-based evolutionary algorithms especially NSGA-II [22] have been very popular choices for solving MONDP/MONDPU [74, 77, 12, 50]. Meanwhile, MONDPU were also studied in the context of water resources systems [25, 1], petro-chemical sensor networks [11], green supply chain [89], and sales networks [27], where alternative solution algorithms such as ParEGO [45], parallel variable neighborhood search [27] and GOMORS [1], etc. were proposed. Similar to the single-objective case, most aforementioned methods were designed under a deterministic setting with no natural extension to handle complex uncertainty structures. Moreover, evolutionary algorithms typically require simulating the objective values of hundreds to thousands of candidate solutions, which can be computationally intensive for MONDPU problems on large networks. Therefore, we are also motivated to design a MONDPU modeling framework with minimal number of objective value evaluations and efficient management of uncertainty scenarios.

1.3 Limitations of Current NDPU/MONDPU Algorithms

It is worth noticing that we can view most aforementioned heuristic or simulation optimization-based NDPU/MONDPU solution algorithms as "information collection" procedures where we iteratively "learn" about and "search" for the optimal solution by sampling "promising" candidate solutions. Then, there are a few aspects in the structure of NDPU/MONDPU that current methods have not fully exploited. First, due to the combinatorial nature of NDPU/MONDPU, we would expect high correlations among different candidate solutions: for example, solution that share a common subset of projects or solutions with similar numbers of projects. As a result, knowing the performance of one solution implicitly gives us a lot of information about other similar alternatives, which would be worth keeping track of. Second, in existing methods especially heuristic algorithms, the sampling decisions are usually derived from fixed local/stochastic search rules without utilizing the information collected from previous iterations. In reality, it is often possible to identify unattractive solutions at early iterations and explicitly concentrate on more promising alternatives. In summary, if we can design an NDPU model to lever the correlation structure among alternatives and the historical information contained in early samples, we would be able to avoid wasting simulation samples on alternatives with lower or similar performances.

1.4 Overview of Ranking and Selection Methods

In this dissertation, we adapt the Bayesian Ranking and Selection procedure [37] from the statistical learning community to model NDPU and MONDPU. Ranking-and-selection (R&S) procedures [42] compare a finite number of alternatives with stochastic performances through a limited sampling budget. Bayesian (R&S) models places a probabilistic prior belief on the mean performances of all alternatives, update the belief through sequential samples, and select the candidate with the best posterior belief. Unlike many other learning/optimization models, R&S methods are inspired by multiple comparison procedures in the statistics literature (e.g. [5]), and can generally accommodate various uncertainty structures via different families of probability distributions. R&S models, especially their frequentist versions, were traditionally solved by the Indifference Zone (IZ) approach (e.g. [41]). IZ procedures ensure that the best alternative will be correctly selected with probability $1 - \alpha$ when the best solution is at least δ better than all other alternatives in reward. In recent years, value-of-information (VOI) based methods such as [15] and [14] have also been very popular. Most VOI policies consist of a series of "myopic" (i.e. one-step look-ahead) sampling decisions made at each iteration to maximize the expected information gains of the next iteration. The sequential nature of VOI policies is highly analogous to iterative algorithms for optimization problems and is thus ideal for the development of NDPU/MONDPU models.

The sampling policies we adapted to solve NDPU/MONDPU in Bayesian $R\&S$ formulation is a class of VOI policy known as knowledge gradient (KG) policies. Roughly speaking, the knowledge gradient of an alternative represent the expected improvement in the proceeding posterior reward if that particular alternative is sampled next. Many studies, from as early as [36], have developed computational procedures for knowledge-gradient policies where the prior beliefs on alternative performances are assumed to be independent. In recent years, knowledge gradient policies in Bayesian R&S problems with correlated belief structure were also studied [29, 71, 68], which greatly improved the number of alternatives KG polices and Bayesian R&S models can efficiently handle. KG policies have the flexibility to incorporate both discrete [29] and continuous[71] decision variables, as well as parametric [52] or non-parametric [49, 4] approximations of the objective function. Its computational efficiency is comparable or better than many aforementioned simulation optimization algorithms (see [29]), with the additional capability to characterize and utilize the problems' uncertainty structure. Due to the combinatorial nature of NDPU/MONDPU and its non-uniform uncertainty structure, we think Bayesian R&S model and the knowledge gradient policies can be a potentially suitable choice for modeling NDPU/MONDPU.

1.5 Organization of Dissertation

In this section we briefly describe the organization of the dissertation.

Chapter 2 In Chapter 2, we introduce a Bayesian R&S model with correlated beliefs for single-objective NDPU. This model is based on a very generic formulation in [29], where our belief of all alternative performances is a fully specified random vector with explicit means and covariance matrices. We extended the formulation of the Bayesian R&S model to account for constraints and nonuniform alternative variances, and adopted the Knowledge Gradient algorithm with Correlated Belief (KGCB) [29] as our sampling policy. The accuracy of the Bayesian R&S model proved to be much better than a number of popular heuristic algorithms. Nevertheless, the expensive computation of the KGCB policy and update of Bayesian beliefs limit the practical problem size to a few thousand alternatives.

Chapter 3 In Chapter 3, we further improve the computational efficiency of the Bayesian R&S model for single-objective NPDU with two powerful approximations: 1) a linear, parametric representation of our belief on the performance of alternatives as well as their variances and 2) A surrogate-assisted knowledge gradient sampling policy which constructs a surrogate optimization problem with the approximated objective function and uses its solutions to constraint the scale of the knowledge gradient calculation at each iteration. The parametric belief structure allows us to perform learning on the compact parameter space instead of the entire solution space, while the surrogate-assisted sampling policy enables us to neglect the majority of the solution space without losing much insight about the value of information of each sample. These approximations limit the additional computational cost of our model to only a low-degree polynomial of the number of projects, which in practice only adds a fraction of a second per iteration even in our stylized Matlab implementation. The new model was extensively tested for various scenarios on the Sioux Falls and Anaheim network, and was found to be able to save $80\% \sim 90\%$ of the samples compared with heuristic methods. Its performance is comparable or sometimes even better than the exact-belief model we developed in Chapter 2. Bayesian $R\&S$ models with parametric beliefs has been studied for continuous decision variables in the context of linear programming with unknown coefficients [67] and for discrete decision variables with special structures in drug-discovery [52]. We believe our new Bayesian $R\&S$ model for NDPU complements the previous studies with a more generic and practical framework for large-scale problems with discrete and possibly continuous decision variables.

Chapter 4 In Chapter 4, we extend our new formulation in Chapter 3 to MONDP/MONDPU by proposing a multi-objective Bayesian Ranking and Selection (R&S) model. In this model, each objective function has its own independent parametric Bayesian belief, while the sampling decisions are made by jointly considering the value of information of all objective functions. The candidate pool for the sampling alternative at each iteration is generated by solving for the Pareto optimal front of the surrogate problem. The sampling decision is then based on a multi-objective KG policy, which defines the knowledge gradient of an multi-objective optimization/learning problem as the vector of single-objective knowledge gradients. Solutions in the Pareto front of the knowledge gradient vectors are selected or ranked further by their relative crowding distances to match the per-iteration sampling budget. We tested this mutli-objective Bayesian $R\&S$ model again on the Sioux Fall network and the Anaheim network with the NSGAII algorithm serving as a benchmark. Results showed that the multi-objective Bayesian R&S model outperforms NSGAII not only in its convergence speed, but also in the spread, number and diversity of final solutions. The set of solutions provided by the Bayesian R&S model are also much more reflective of the patterns of the true Pareto optimal set, which greatly enhances the practicality of MONDPU models in real-world multi-objective decision making.

Chapter 5 Chapter 5 summarizes our findings and conclusions in the previous chapters. In particular, we will discuss the advantage and limitation of the Bayesian R&S model in modeling NDP/NDPU problems, future improvements for the models, and possible extension of the model to other problem classes.

CHAPTER 2

A SEQUENTIAL BAYESIAN MODEL FOR THE DISCRETE NETWORK DESIGN PROBLEM WITH UNCERTAINTIES

2.1 Introduction

The Network Design Problem (NDP) is a classical problem in combinatorial optimization with numerous applications in transportation sciences and engineering. Given a network and its users (demand), NDP is formulated to choose among a set of "projects"(modifications to the network) to optimize certain system-wide objective(s) over the network. Examples of those objectives include total travel time on the network, system capacity of the network and in recent studies its environmental impact or energy consumption [26]. If we allow the NDP formulation to include randomness in travel demand, travel cost, etc., we obtain the Network Design Problem with Uncertainties (NDPU). The intricacy of the NDP/NDPU problem lies in the fact that each "project" not only affects the structure of the network, but also alters the way users utilize the network once they are aware of the changes. Indeed, to capture such interactions, NDP/NDPU are typically formulated as Bi-level programs [23], which are known to be NP-hard even in their simplest forms [87].

Due to its proven complexity, earlier methods for solving the deterministic NDP problem focused on relaxing various aspects of the problem (e.g. objective functions [58], constraints [18], lower-level problems [18], etc.). Exact solutions were only available in special cases with nice properties, e.g. when the volume-delay function is assumed to be constant [7]. A good review of these methods can be found in [93] and [59]. In recent years, iterative meta-heuristic algorithms have been another popular branch of methodology for solving the NDP. Applications of such heuristic algorithms include Genetic Algorithms (GA) [94], Simulated Annealing (SA) [32], Tabu Search [9], Ant Systems [57], etc. These algorithms usually start with a few arbitrary feasible solutions and move towards better ones via local and/or stochastic search rules. The performances of those algorithms were usually satisfactory although their convergence speeds are known to be rather slow [48, 28, 38, 17, 69].

The NDPU problems were first solved also by heuristic methods [79], in which uncertainty was characterized by probability distributions over a finite scenario set. The objective function of the NDPU problem then becomes the *expected* system performance under that probability distribution. If the scenario set is small, one can afford to convert the NDPU into a larger NDP problem with multiple lower level problems and solve it with NDP solution algorithms [79, 72]. When the scenario set becomes large or infinite (e.g. the governing probability distribution is continuous), approximation methods such as Single Point Approximation, Monte Carlo sampling, and Sample Average Approximation are needed. In Single Point Approximation, one tries to construct an input for which the optimal solution closely matches the true optimal solution [84, 82]. In Monte Carlo sampling, one generates repeated samples for a particular candidate solution to estimate its true expected objective value [73]. More sophisticated sampling scheme like Randomized Quasi−Monte Carlo Sampling or Antithetic Sampling can also be used to improve sampling efficiency [73]. In Sample Average Approximation (SAA) [72], the NDPU problem is solved repeatedly under a smaller scenario set sampled from the full scenario set and the best solution generated in this process is then used to approximate the true optimal solution [55].

With the exception of single point approximation, all the aforementioned procedures require simulating hundreds to thousands of candidate solutions during the computational process. Meanwhile, network design problems in practice can be imposed on large networks with tens of thousands of links, making each evaluation of the objective value computationally expensive. In extreme cases where traffic demand is modeled by activity-based travel demand model such as NYMTC (New York Metropolitan Transportation Council)'s NYBPM[53], even one simulation run can take hours or days. Therefore, there is strong motivation to reduce the number of samples/simulations as much as possible.

In order to reduce the simulation burden of the NDP/NDPU, it is worth noticing that we can view all aforementioned heuristic algorithms as "information collection" procedures, where we iteratively "learn" about and "search" for the optimal solution by sampling "promising" candidate solutions. From this perspective, there are a few aspects in the structure of NDP/NDPU that current methods have not fully exploited. First, due to the combinatorial nature of NDP/NDPU, we would expect high correlations among different candidate solutions: for example, solution that share a common subset of projects or solutions with similar numbers of projects. As a result, knowing the performance of one solution implicitly gives us a lot of information about other similar alternatives. Second, in existing methods especially heuristic algorithms, the alternatives to be simulated at each iteration are usually derived from fixed local/stochastic search rules without utilizing the information in the sampling history. In reality, it is often possible to identify unattractive solutions at early iterations and explicitly concentrate on more promising alternatives. In summary, if we can design a sequential sampling policy that efficiently leverages the correlation structure among alternatives, we would be able to avoid wasting simulation samples on alternatives with lower or similar performances.

In this paper, we propose a Bayesian Ranking and Selection formulation to the NDP/NDPU problem. Ranking-and-selection (R&S) procedures [42] attempt to compare a finite number of alternatives with stochastic performances through a finite sampling budget. Under the Bayesian R&S setting, we view each candidate solution to the NDPU problem as an alternative and place prior beliefs on the expected objective value of each alternative. The belief of each alternative is then updated by taking "samples" (simulations) from selected alternatives. Solving the NDPU problem is then equivalent to designing a sampling policy that will quickly identify the alternative with the largest expected reward. Early research on $R\&S$ problems have focused on situations where the prior beliefs on alternative performances are assumed to be independent. In recent years, $R\&S$ problems with correlated prior belief structure were studied [29, 71, 68] and the corresponding solution algorithms exhibited close similarities to iterative optimization algorithms. To solve the NDP/NDPU in the new Bayesian R&S formulation, we adopt the Knowledge Gradient algorithm with Correlated Belief (KGCB) [29], which has great computational performance and rigorous mathematical properties. We believe such a new formulation can in general provide alternative insights to the construction and solution of NDP/NDPU problems or even other stochastic optimization problems with combinatorial structures.

In Section 2, we first describe the traditional formulation of the NDP/NDPU problem and show how they can be viewed as a Bayesian Ranking and Selection problem. Then we describe the KGCB sampling policy and customize it for

NDP/NDPU. Section 3 provides computational examples and Section 4 provides conclusions and discusses directions for future research.

2.2 Methodology

2.2.1 The Network Design Problem with Uncertainty

Assume a generic network (graph) $G = (V, E)$, with V and E being the vertex set and the edge set respectively. For each link in the edge set E, we have information about its capacity, length and other attributes. A network design problem (NDP) is formulated when the decision maker wants to optimize certain objectives (e.g. maximizing throughput, minimizing congestion, etc.) over the network by modifying the edge set E with k improvement "projects". The projects can come in the form of adding new links to the network (Discrete NDP), modifying the attributes of existing links (Continuous NDP) or both (Mixed NDP). In this paper we focus on Discrete NDP for demonstration purposes. Let the vector **a** $= (a_1, a_2, ...a_k)$ be an overall decision (solution) with each element a_i representing the binary decision on project i, and let **c** $= (c_1, c_2, ...c_k)$ be the corresponding project costs. In the discrete case, each element a_i is a binary number indicating whether project i is implemented or not. The NDP then seeks a decision **a*** to optimize the pre-defined system performance measure, subject to a total budget cap B.

Uncertainty in the NDPU problem is usually characterized by a countable scenario space Ω, with the probability of each scenario $\{p(\omega) : \omega \in \Omega\}$ well defined. Parameters in the NDPU problem can take random values based on ω, and the objective of a NDPU is to choose the decision $\mathbf{a}^*$ that will optimize the expected value of the objective function (or in some cases a convex combination of the higher moments of the objective function [83]).

The NDPU problem is usually formulated as a Bi-level mathematical program. The upper level problem models the decision makers' objectives while the lower level models the network users' responsive behavior under changes to the network. Without loss of generality, in this paper we focus on a typical formulation of the NDPU problem to maximize reduction in total travel time over the network, assuming that uncertainty only exists in travel demands. The corresponding formulation is then given as below:

$$\text{(Upper Level)} \max_{\mathbf{a}\in A} \quad \mathbb{E}(T_{\mathbf{a}}) = T_0 - \sum_{\omega\in\Omega} p(\omega) \cdot \sum_{(i,j)\in E\cup E'_{\mathbf{a}}} x^*_{ij}(\omega) t_{ij}(x^*_{ij}(\omega)) \tag{2.1}$$

$$\mathbf{c}^T \cdot \mathbf{a} = \sum_{i=1}^{k} c_i \cdot a_i \leq B \tag{2.2}$$

$$a_i \in \{0,1\}, \forall i \in \{1, 2...k\} \tag{2.3}$$

$$\{x_{ij}(\omega)^* : (i,j) \in E\} \tag{2.4}$$

is the optimal solution for the lower level problem:

(2.5)

$$\text{(Lower Level)} \quad \min_{x} \quad \sum_{(i,j)\in E\cup E'_{\mathbf{a}}} \int_0^{x_{ij}} t_{ij}(u)du \tag{2.6}$$

$$\text{s.t.:} \quad \sum_{p\in P_{rs}} f_p^{rs} = d(\omega)_{rs}, \forall (r,s) \in D \tag{2.7}$$

$$x_{ij} = \sum_{(r,s)\in D} \sum_{p\in P_{rs}} f_p^{rs} \cdot \delta_{ij,p}^{rs} \tag{2.8}$$

$$t_{ij} = t_{ij}^0 \cdot (1 + \alpha \cdot (\frac{x_{ij}}{C_{ij}})^{\beta}) \tag{2.9}$$

$$f_p^{rs} \geq 0, \forall p \in P_{rs}, \forall (r,s) \in D \tag{2.10}$$

where

A : the set of all network configurations, i.e. $\{0,1\}^k$

T_0 : expected total travel time on the base network

$E'_{\mathbf{a}}$: the set of additional links selected by decision **a**.

x_{ij} : the equilibrium traffic flow on link (i,j).

$t_{ij}(x)$: travel time on link (i,j) when flow is x

t_{ij}^0 : free-flow travel time on link (i,j)

C_{ij} : capacity of link (i,j)

α, β : parameters for calculating travel time

D : the set of traffic demands indexed by Origin(r) and Destination(s).

P_{rs} : the set of paths (contiguous links) which starts in node r and ends in node s

f_p^{rs} : the flow between origin r and destination s on path p

$\delta_{ij,p}$: equals 1 if link (i,j) belongs to path p and 0 otherwise

Equations 2.1-2.5 define the Upper level problem. Equation 2.1 is the objective function to maximize improvement in total travel time and Equation 2.2 is the budget constraint. Equations 2.6-2.10 specify the lower level problem which is assumed to be the deterministic User Equilibrium Problem [75] in this example. In particular, Equation 2.9 assumes the volume-delay function follows the traditional BPR formula, which can be replaced by other volume-delay functions. We can also replace the entire lower level problem with other models such as stochastic user equilibrium[75] or system capacity maximization [79].

In practice, many lower level problems of NDPU have efficient solution methods that are too sophisticated to be integrated into the bi-level structure of NDPU. Therefore, an alternative approach to the NDPU problem is to treat the Lower Level problem as part of an expensive evaluation of the objective value in the Upper level problem. Then, the NDPU problem becomes a discrete optimization problem whose objective function has no closed-form with respect to the decision variables. This viewpoint is at the heart of the Bayesian Ranking & Selection model we develop in next section.

2.2.2 NDPU as a Bayesian Ranking and Selection Problem

The Bayesian Ranking and Selection Problem

In a Bayesian Ranking and Selection (R&S) problem, we have M alternatives whose rewards are random with mean $\boldsymbol{\theta} = \{\theta_1 ... \theta_M\}$. Our objective is to identify the alternative with the maximum expected reward through sample measurements. We have a prior belief about $\boldsymbol{\theta}$, denoted as $\boldsymbol{\mu}^0 = \{\mu_1^0 ... \mu_M^0\}$, and possibly a prior belief about the correlation structure of $\boldsymbol{\theta}$, characterized by an $M \times M$ positive semi-definite matrix Σ^0. We are given the opportunity to make N sample decisions, $\mathbf{X} = \{x_1 ... x_N\}$, and obtain the corresponding sample observations $\mathbf{Y} = \{\hat{y_1} ... \hat{y_N}\}$. Let $\mathcal{F}^n$ be the sigma-algebra generated by $\{x_1 ... x_n\}$

and $\{\hat{y_1}...\hat{y_n}\}$. After taking all the samples, our belief about the means and the covariance matrix is updated to $\boldsymbol{\mu}^N := \mathbb{E}(\boldsymbol{\theta}|\mathcal{F}^{\mathbb{N}})$ and $\Sigma^N := Cov(\boldsymbol{\theta}|\mathcal{F}^N)$ respectively via Bayes' Rule [33]. When sampling is completed, we select the alternative with the largest posterior mean (i.e., $x_N^* = \arg\max_x \mu_x^N$) as the optimal solution. The goal for a Bayesian Ranking and Selection problem is to maximize the posterior expected reward after N measurements through a sampling policy $\pi = \{x_1^\pi...x_N^\pi\}$. This objective can be written as:

$$\sup_{\pi \in \Pi} \mathbb{E}^\pi[\max_x \mu_x^N] \tag{2.11}$$

Where Π is the set of all possible sampling policies with sampling budget N, and $\mathbb{E}^\pi$ denotes the conditional expectation under policy π. For R&S problems we are usually interested in *sequential* sampling policies under which x_n^π is measurable with respect to $\mathcal{F}^{n-1}$ for all $n = 1...N$. In other words, our measurement decision x_n at time n depends only on the sampling history $x_1...x_{n-1}$ and $y_1...y_{n-1}$.

The Bayesian Ranking and Selection Model for NDPU

Prior and Likelihood We can formulate the NDP/NDPU problem in Section 2.1 as a Bayesian Ranking and Selection problem by viewing each network configuration (decision) **a** as an alternative. More specifically, we can think of **a** as a k-digit binary variable, the i_{th} digit of which indicates whether project i will be implemented. The decimal value of **a** can be viewed as an index for the alternative set A and we can assume without loss of generality that alternatives in A are ordered by the decimal value of **a**. As we simulate alternatives $\mathbf{a}_1, \mathbf{a}_2...\mathbf{a}_N$, we obtain (perfect) measurements of the random "reward" of those "alternatives', denoted as $T_{\mathbf{a}_1}, T_{\mathbf{a}_2}, ...T_{\mathbf{a}_N}$.

For the NDP/NDPU problem, we further assume our prior belief about the *expected* "reward" of all alternatives follows a multi-variate normal distribution and that the sequence of measurements follow normal distributions with known variances conditioned on the parameter values:

$$\boldsymbol{\theta} := (\mathbb{E}(T_1), \mathbb{E}(T_2)...\mathbb{E}(T_{|A|})) \sim \mathcal{N}(\mu^0, \Sigma^0) \tag{2.12}$$

$$T_{\mathbf{a}_i}|\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}_{\mathbf{a}_i}, \lambda_{\mathbf{a}_i}), \forall i \in 1...N \tag{2.13}$$

It is easy to see that the deterministic NDP problem is a special case of the formulation where $\lambda_{\mathbf{a}_i} = 0, \;\; \forall i \in 1...N$. For most stochastic problems, $\{\lambda_{\mathbf{a}_i}\}_{i=1}^N$ is usually unknown and needs to be estimated either from samples or from our prior understanding of the problem (to be discussed in Section 3). In addition, it is worth noticing that although the *actual* objective value for each alternative can be correlated, the sequence of measurements where we place our probabilistic beliefs are drawn from independent simulations. Therefore, the likelihood function of those measurements in Equation 2.13 are independent.

Posterior Distribution It is a well-known result [33] that under Equation 2.12 and 2.13 the posterior distribution of $\boldsymbol{\theta}$ will also be normal, i.e.:

$$\boldsymbol{\theta}|\mathcal{F}^n \sim \mathcal{N}(\mu^n, \Sigma^n), \; \forall n \in 1...N \tag{2.14}$$

where μ^n and Σ^n can be updated through the recursive equations:

$$\mu^n = \mu^{n-1} + \frac{\hat{T}_{\mathbf{a}_n} - \mu^{n-1}_{\mathbf{a}_n}}{\lambda_{\mathbf{a}_n} + \sigma^{n-1}_{\mathbf{a}_n\mathbf{a}_n}} \Sigma^{n-1} e_{\mathbf{a}_n} \tag{2.15}$$

$$\Sigma^n = \Sigma^{n-1} - \frac{\Sigma^{n-1} e_{\mathbf{a}_n} e'_{\mathbf{a}_n} \Sigma^{n-1}}{\lambda_{\mathbf{a}_n} + \sigma^{n-1}_{\mathbf{a}_n\mathbf{a}_n}} \tag{2.16}$$

Here $\mathbf{a}_n$ is the alternative sampled at iteration n, $e_{\mathbf{a}_n}$ is the column vector with the $\mathbf{a}_n$th (in terms of $\mathbf{a}_n$'s decimal value) element set to 1 and all others set to 0, and $\sigma_{\mathbf{a}_n\mathbf{a}_n}$ is the $\mathbf{a}_n$th diagonal element of Σ^{n-1}. Consequently, it is sufficient to summarize our progressive belief about $\boldsymbol{\theta}$ in the sequence $(\mu_i, \Sigma_i)_{i=1}^N$. As a large class of distributions can be approximated or transformed into normal distributions, maintaining $(\mu_i, \Sigma_i)_{i=1}^N$ can usually be much more efficient and informative than the traditional scenario-based, discretized uncertainty formulation (even if normality is strongly violated, we still have the option to use Equations 2.15 and 2.16 to learn about each discrete scenario with λ set to 0). Moreover, if Σ^0 is non-diagonal, then at each iteration Equations 2.15 and 2.16 enable us to update our estimate of the *entire* (μ^n, Σ^n) from a single observation.

Decision After we have exhausted N simulation samples, we select the network configuration with the best posterior reward as the proposed optimal solution:

$$\mathbf{a}_N^* \in \arg\max_{\mathbf{a}\in A^f} \mathbb{E}(T_{\mathbf{a}}|\mathcal{F}^N) \tag{2.17}$$

Here A^f denotes the set of feasible alternatives. Note that $\mathbf{a}_N^*$, the best solution conditioned on the knowledge collected by the sampling policy at time N, is often different from $\mathbf{a}^* \in \arg\max_{\mathbf{a}} \theta_a := \arg\max_{\mathbf{a}} \mathbb{E}(T_{\mathbf{a}})$, the best solution given *perfect* knowledge. This difference is analogous to the optimality "gap" of an iterative algorithm executed in finite iterations.

Constraint Handling in NDPU with Bayesian R&S Formulation

In traditional Bayesian Ranking and Selection problems, every alternative is assumed to be feasible, which does not apply to NDP/NDPU with the budget constraint (Equation 2.2). Instead, we can utilize the coefficients of the budget constraint (i.,e. project costs) to reduce $|A|$. A reduced alternative set, as we will see in Section 3, will greatly improve the performance of our sampling policy.

Partial Elimination of Infeasible Alternatives The problem of approximating the size of the feasible solution set constrained by Equation 2.2 is sometimes referred to as Knapsack Counting [34]. For the NDPU problem with Bayesian R&S formulation, we apply a simple bound on the size of the feasible solution set by limiting the maximum number of active projects in any feasible alternative. The procedure is described in Algorithm 2.1, which uses the subset of least expensive projects to bound the number of active projects. Admittedly, the actual effectiveness of Algorithm 2.1 depends on specific values of $\{c_1...c_k\}$, but it will be generally effective as long as the variability within $\{c_1...c_k\}$ is not dramatic.

Algorithm 2.1: Find the maximum number of projects in any alternative

```
Require: Project costs $\{c_1...c_k\}$ and budget cap B
1: Sort $\{c_1...c_k\}$ in ascending order to $\{c_{(1)}...c_{(k)}\}$.
2: for i= 1 to k do
3:     if $\sum_{j=1}^{i}(c_{(j)}) > B$ then
4:         return $i$ as $i_{max}$.
5:     end if
6: end for
```

i_{max} defines a new set $A' := \{\mathbf{a} : \sum_{i=1}^{k} a_i \leq i_{max}\} \subset A$. We can quickly verify that the true feasible set $A^f \subset A'$, since any alternative in $A \setminus A'$ (i.e. with greater than i_{max} active projects) must have already violated the budget constraint and is thus infeasible. We may also have $A^f = A'$ in some special cases, e.g. when the cost of projects are the same. The time complexity of Algorithm 2.1 is $O(k \log k)$. The size of A′ is $|A'| = \sum_{i=1}^{i_{max}} \binom{k}{i}$ by construction.

Re-indexing of the Reduced Alternative Set The natural indices of elements in $|A'|$ no longer corresponds to the configuration of the network in their binary forms as they do in the original alternative set A. To efficiently (that is, without a "brute-force" mapping of $O(|A'|)$ space complexity) keep track of the correspondence between each alternative's network configuration and its new index in $|A|'$, we can arrange the alternatives (in ascending order) first by the number of active projects then by the decimal value of the network configuration. The arrangement to index any arbitrary n-choose-k combination set is also known as the Combinatorial Number System [46]. There exists an efficient procedure (Algorithm 2.2) to "decode" the configuration of the network from the natural index of the elements in $|A'|$ [76].

```
Algorithm 2.2: Retrieve the network configuration of an alternative from
               its index in A′
Require: Natural index i and maximum number of projects k
 1: Step 1: Determine the number of active projects r in alternative i and its
    position $n^r$ in the k choose r combination set.
 2: $r \leftarrow 1,\ n^r \leftarrow i$
 3: while $n^r > \binom{k}{r}$ do
 4:     $n^r \leftarrow n^r - \binom{k}{r},\ r \leftarrow r + 1$
 5: end while
 6: Step 2: Retrieve the corresponding network configuration $\mathbf{a}(i) = (a_1, a_2 ... a_k)$
 7: for j= r to 1 do
 8:     if $j > r$ and $r \geq 0$
 9:         $y \leftarrow \binom{j-1}{r}$
10:     else $y \leftarrow 0$
11:     if $n^r >= y$
12:         $n^r = n^r - y,\ a_j \leftarrow 1,\ r \leftarrow r - 1$
13:     else $a_j \leftarrow 0$
14: end for
15: return $\mathbf{a}(i)$
```

The time complexity of Algorithm 2.2 is $O(k)$. Before we start the sampling process, we use Algorithm 2.1 to determine the size of the reduced A'. In each sampling iteration, we apply Algorithm 2.2 to retrieve the configuration of the network.

2.2.3 Solving NDP/NDPU with Bayesian R&S Formulation

Overview of Solution Procedure

The solution procedure for NDP/NDPU with Bayesian Ranking and Selection formulation (Equations 2.11-2.13) can be summarized in Algorithm 2.3 below. In short, we iteratively update our Bayesian belief about the alternative rewards through adaptive sequential simulation decisions. At the end of the last iteration, we select a *feasible* solution with the best posterior mean. In the next two sections we describe the KGCB sampling policy we use for Line 3 and the variance update procedure in Line 7 of the algorithm.

Algorithm 2.3: Solving NDPU with Bayesian $R\&S$ formulation

```
Require: Inputs $\mu^0$, $\Sigma^0$, budget cap B, Project costs c
1: Pre-eliminate infeasible solutions using Algorithm 2.1.
2: for $n = 1 \to N$ do
3:     Determine $i_n$, the index of the next alternative to sample.
4:     Apply Algorithm 2.2 to decode the network configuration $\mathbf{a}_n$ from $i_n$
5:     Simulate network configuration $\mathbf{a}_n$ and obtain objective value $\hat{T}_{\mathbf{a}_n}$
6:     Update alternative variance $\hat{\lambda^n_{\mathbf{a}_n}}$
7:     Given $\mathbf{a}_n$, $\hat{\lambda^n_{\mathbf{a}_n}}$ and $\hat{T}_{\mathbf{a}_n}$, update posterior belief $(\mu^{n-1}, \Sigma^{n-1})$ to $(\mu^n, \Sigma^n)$ using Equations 2.15 and 2.16.
8: end for
9: return $\arg\max_{\mathbf{a} \in A^f} \mu^N_{\mathbf{a}}$
```

The KGCB Sampling Policy for NDPU

In this section we introduce the KGCB sampling policy [29] for Bayesian $R\&S$ problem with correlated beliefs. The core idea of the KGCB policy is that at each iteration one should choose to sample the alternative from which the "Knowledge Gradient" is maximized. That is, we should choose

$$\mathbf{a}^{KG,n} \in \arg\max_{\mathbf{a}\in A} v^{KG,n}(\mathbf{a}) = \arg\max_{\mathbf{a}\in A} \left(\mathbb{E}^n(\max_{i\in A} \mu_i^{n+1} | \mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}) - \max_{i\in A} \mu_i^n\right) \tag{2.18}$$

Note that $max_i \mu_i^n$ is the optimal value we would receive if we stop at time n and $max_i \mu_i^{n+1} | x_n$ is the optimal value we would obtain if we take an additional sample x_n. The knowledge gradient represents the expected improvement in posterior optimal value obtained from measuring a particular alternative, which we want to maximize at each iteration. By calculating the conditional *predictive* expectation of $max_i \mu_i^{n+1}$, we can forecast the value of information of all alternatives without taking actual samples of them. It is presented in [29] that the KGCB policy is almost-surely optimal for N=1 or N$\to\infty$ (i.e. both myopically and asymptotically optimal) and has sub-optimality bounds when N is finite.

With the presence of infeasible solutions in NDP/NDPU, we need to slightly modify the definition of the KGCB sampling policy to:

$$\mathbf{a}_f^{KG,n} \in \arg\max_{\mathbf{a}\in A'} v_f^{KG,n}(\mathbf{a}) = \arg\max_{\mathbf{a}\in A'} \left(\mathbb{E}^n(\max_{i\in A^f} \mu_i^{n+1} | \mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}) - \max_{i\in A^f} \mu_i^n\right) \tag{2.19}$$

In short, we now select the alternative which maximizes the expected improvement of the *constrained* optimal value instead. Note that in the $\arg\max$ operator we do include infeasible solutions in A', as they could potentially have higher $v_f^{KG}(.)$ value (e.g. due to the common active projects they share with some highly optimal feasible solutions). Fortunately, the computational procedure to calculate $\mathbf{a}^{KG,n}$ in [29] still applies to $\mathbf{a}_f^{KG,n}$ after minor adjustments. In specific, $\mathbf{a}_n$ in Algorithm 2.3 can be determined via the following Algorithm 2.4 at each iteration. The time complexity of Algorithm 2.4 is $O(\log(|A^f|)|A^f||A'|)$.

Algorithm 2.4: The KGCB sampling policy for NDPU

```
Require: Inputs $\mu^n$ and $\Sigma^n$.
 1: for each $\mathbf{a} \in A'$ do
 2:     $p \leftarrow \mu^n_{A^f}$, $q \leftarrow (\Sigma^n \cdot e_{\mathbf{a}})_{A^f} / \sqrt{\lambda_{\mathbf{a}} + \sigma^n_{\mathbf{aa}}}$
 3:     ($\mathrm{v}_{A^f}$ returns the feasible sub-vector of the vector v)
 4:     Sort the sequence of pairs $(p_i, q_i)_{i=1}^{|A^f|}$ so that the $q_i$ are in non-decreasing
        order and ties in q are broken so that $p_i \leq p_{i+1}$ if $q_i = q_{i+1}$.
 5:     Remove all entry i in $(p_i, q_i)_{i=1}^{|A^f|}$ where $q_i = q_{i+1}$
 6:     $c_0 \leftarrow -\infty$, $c_{|A'|} \leftarrow +\infty$, $c_i \leftarrow -(p_{i+1} - p_i)/(q_{i+1} - q_i)$, $\forall i \in 1...|A^f| - 1$
 7:     Remove all entry i in $(p_i, q_i, c_i)_{i=1}^{|A^f|}$ where $c_i \geq c_{i+1}$
 8:     $v_f^{KG,n}(\mathbf{a}) \leftarrow \log(\sum_{i=1}^{|A^f|-1} (q_{i+1} - q_i)(\varphi(-|c_i|) - |c_i|\Phi(-|c_i|)))$
 9:     ($\varphi(.)$ and $\Phi(.)$ is the pdf and the cdf of a standard normal variable)
10: end for
11: return $\mathbf{a}_n = \mathbf{a}_f^{KG,n} \in \arg\max_{\mathbf{a} \in A'} v_f^{KG,n}(\mathbf{a})$
```

Approximation and Update of Unknown Alternative Variances

In Algorithm 2.4 and the Bayesian R&S model described in Equations 2.12 and 2.13, we assume that the alternative variances λ_i are known for all $i \in \{1...|A'|\}$. In reality, $(\lambda_i)_{i=1}^{|A'|}$ are usually unknown and non-uniform across alternatives. While there exist Bayesian models that jointly consider unknown multi-variate alternative means and variances, few (except the very recent work of [61]) can possibly update beliefs through single scalar observations as required by the formulation of NDPU. For NDPU, we develop a separate light-weighted approximation/update procedure inspired by the Bayesian normal model with known mean and unknown variance [33] to estimate the variances of the "rewards" of all alternatives.

We estimate $(\lambda_i)_{i=1}^{|A'|}$ first by a hierarchical "prior" belief that λ_i depends only on the number of projects in alternative i and a deviation term, i.e.

$$\hat{\lambda}_{\mathbf{a}}^{0} = \hat{\lambda}^{pCount(\mathbf{a})} + \epsilon_{pCount(\mathbf{a})}, \forall \mathbf{a} \in A' \tag{2.20}$$

Where $pCount(\mathbf{a})$ returns the number of active projects in alternative $\mathbf{a}$. $\epsilon_{pCount(\mathbf{a})}$ is usually set to a random fraction of $\hat{\lambda}^{pCount(\mathbf{a})}$ for each $\mathbf{a}$.

As the algorithm proceeds, we can collect more samples for a particular alternative $\mathbf{a}$ and update our estimate of $\lambda_{\mathbf{a}}$. For an iteration $n \in 1...N$ in which we sampled alternative $\mathbf{a}$, we have:

$$\hat{\lambda}_{\mathbf{a}}^{n} = \frac{\hat{\lambda}_{\mathbf{a}}^{n-1} \cdot (v_0 + N_{\mathbf{a}} - 3) + (y_{\mathbf{a}}^{n} - \mu_{\mathbf{a}}^{n})^2}{v_0 + N_{\mathbf{a}} - 2} \tag{2.21}$$

$$= \frac{v_0 \hat{\lambda}_{\mathbf{a}}^{0} + \sum_{i=1}^{N_{\mathbf{a}}} (y_{\mathbf{a}}^{n(i)} - \mu_{\mathbf{a}}^{n(i)})^2}{v_0 + N_{\mathbf{a}} - 2} \tag{2.22}$$

$$\approx \frac{v_0 \hat{\lambda}_{\mathbf{a}}^{0} + \sum_{i=1}^{N_{\mathbf{a}}} (y_{\mathbf{a}}^{n(i)} - \theta_{\mathbf{a}})^2}{v_0 + N_{\mathbf{a}} - 2} \quad \forall \mathbf{a} \in \{\mathbf{a}_1..\mathbf{a}_N\} \tag{2.23}$$

where $N_{\mathbf{a}}$ is the number of samples collected for alternative $\mathbf{a}$ at iteration n, $n(i)$ is the iteration at which the i_{th} sample of alternative $\mathbf{a}$ is collected, $\mu_{\mathbf{a}}^{n(i)}$ is the posterior mean for alternative $\mathbf{a}$ at iteration $n(i)$ and v_0 is a non-negative weight parameter. Equation 2.21 is mathematically equivalent to the "posterior" mean of $\lambda_{\mathbf{a}}$ (i.e. Equation 2.23) if we put a scaled-inverse chi-square prior on $\lambda_{\mathbf{a}}$ (i.e. $\lambda_{\mathbf{a}} \sim$ *Scale-inv-*$\chi^2(v_0, \hat{\lambda}_{\mathbf{a}}^{0})$), except that the true mean $\theta_{\mathbf{a}}$ is approximated by the sequence of posterior means $\mu_{\mathbf{a}}^{n(1)} ... \mu_{\mathbf{a}}^{n(N_{\mathbf{a}})}$. To marginalize the inaccuracy from the posterior means, we require $n_{\mathbf{a}}$ to be at least larger than a threshold n_{min} (typically 3-5) before Equation 2.21 can be applied.

Equations 2.20 and 2.21 reduce the number of variance parameters from $|A'|$ to k. In practice, we sample one alternative from each $pCount \in 1...k$ before running Algorithm 2.3 and use these variance estimates as $(\hat{\lambda}^{pCount})_{pCount=1}^{k}$. Then we plug in the corresponding $\hat{\lambda}_{\mathbf{a}_n}$ into Algorithm 2.4 in each sampling iteration. Inaccurate variance estimation will likely affect the performance of the KGCB sampling policy and the posterior update in earlier iterations. However, as the number of samples accumulates, it is easy to see that the variance estimates will converge to the true variances $\{\lambda_{\mathbf{a}} : \mathbf{a} \in A'\}$, and consequently the asymptotic accuracy of the entire Algorithm 2.3 will be guaranteed.

2.3 Computational Examples

2.3.1 Data Source and Simulation Design

We conducted three sets of computational experiments: deterministic tests, stochastic tests and large network tests. For the deterministic and stochastic tests, we use the famous Sioux Fall network [3] as the underlying transportation network. On top of the base network, we define 10 candidate projects, whose (fictitious) costs are summarized in Table 2.1. According to Table 2.1, the maximum number of feasible network configurations/alternatives is 1023, which makes the problem non-trivial but still manageable for experimental purposes. To test the constraint-handling techniques, we run the problem over two budget levels: \$6000, which represents a scenario with less (398/1023) feasible solutions, and \$10000, which represents a scenario with more (950/1023) feasible solutions. Algorithm 2.1 calculates the maximum number of active projects to be 6 and 8 for the two budget levels respectively.

Table 2.1: List of Projects

ProjectID	**Budget (\$)**	**ProjectID**	**Budget (\$)**
1	1800	2	1500
3	1000	4	1950
5	1650	6	2100
7	1200	8	625
9	650	10	850

For the large-scale network tests, we use two additional larger testing networks of Anaheim and Chicago (sketched version), with the same project costs and budget selection as in Table 2.1. The absolute size of the network is moderate, but they are "large" in the sense that we need to run the lower level User Equilibrium problem on them for hundreds of iterations and tens of repetitions. Projects are typically constructed as bi-directional links that connect OD paths with high volumes. Information about the two networks along with the Sioux Fall network is presented in Table2.2. Data for all three networks are publicly available at [3].

Table 2.2: Size of Networks

Network Name	**# of Nodes**	**# of Links**	**# of OD Pairs**
Sioux Fall	24	76	576
Anaheim	416	914	1444
Chicago "Sketch"	933	2950	$\sim$ 140,000

For all tests, the lower level User Equilibrium problem (Equation 2.6-2.10) is solved by an open-source solver [78] with the relative gap of convergence fixed at 10^{-6}. The implementation of the KGCB sample policy is based on the MatlabKG library available at [30]. For the KGCB algorithm, a non- informative prior is set with $\mu^0 = \mathbf{0}$ and $\Sigma^0 = (0.5I + 0.5 \cdot J_{|A'|}) \cdot 10^{12}$, where $J_{|A'|}$ is the $|A'|$ by $|A'|$ matrix of ones and $\mathbf{0}$ is the $|A'|$-dimensional zero vector.

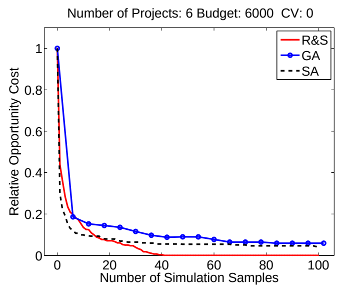
For bench-marking, we solve the same problems using Genetic Algorithm (GA) and Simulated Annealing [43], both of which are very popular for solving NDP/NDPU and other similar discrete optimization problems. The Genetic Algorithm uses evolutionary strategies to evolve a population of alternatives to the optimal solution while Simulated Annealing uses a random walk in the solution space to generate a stream of candidate solutions converging to the optimal solution. For the GA algorithm we use the package from [21]. This package applies an Elitist policy where the best alternative in each generation is always preserved to the next generation. The population size of the GA solver is set to the number of projects in the test, the crossover probability is set to 0.9, and the mutation probability is set to 0.02. Our custom-made SA algorithm is similar to the original version presented in [43], with the additional feature that at each iteration the algorithm will keep searching the neighborhood of the previous solution for a feasible solution to move to. The initial "temperature" of the SA algorithm is set to 100, and decreases by 5% after each simulation evaluation. The "neighborhood" of each solution in the SA algorithm is defined by the set with the active/inactive status of 3 projects flipped. Infeasible solutions are penalized in their objective value in the GA solver and rejected during state transition in the SA algorithm.

The primary performance measure of our experiments is the Relative Opportunity Cost (RelOC) of each algorithm at given iterations. RelOC is defined as the percentage difference between the expected optimal value of the true optimal solution and the expected objective value evaluated at the "best" alternative proposed by the algorithm when it terminates. Thus, once the RelOC is zero, the algorithm has found the best solution possible. We also define the RelOC of all infeasible solutions to be 1, as recommending an infeasible solution does not help our decision-making at all. RelOC is a more reliable performance measure for our Bayesian R&S model than the commonly used difference of objective values, since each of our random samples has the potential to randomly generate large optimal values. In our experiments, the "true" objective value for each alternative is approximated by the mean of 100 Monte Carlo samples.

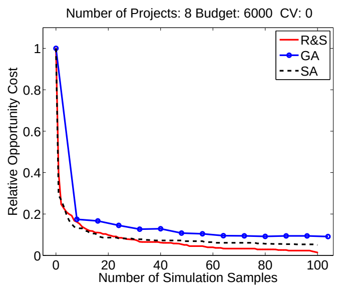
2.3.2 Results for the NDP (Deterministic) Tests

We first test the performance of the Bayesian R&S model for the deterministic NDP problem. This is the case where $\lambda_i = 0, \forall i \in \{1...|A'|\}$ in Equations 2.13 and thus requires no variance approximation. This test serves as a bottom line for performance of the Bayesian R&S model without its advantage in managing uncertainties yet, and can also gauge the model's potential to solve discrete scenario-based NDPU formulations. We assume the NDP problem is formulated as in Equations 2.1-2.10 with the uncertainty in demand removed. We then run the Bayesian $R\&S$ model, the GA solver and the SA solver on the same problem for 30 repetitions and 100 iterations each (note that 100 iterations in KGCB and SA means $\lceil$ 100/populationsize $\rceil$ generations in the GA solver). Comparison of the mean RelOC of the three algorithms are presented below in

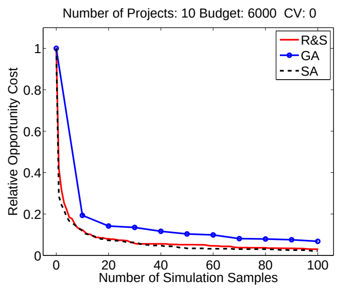
Figure 2.1.



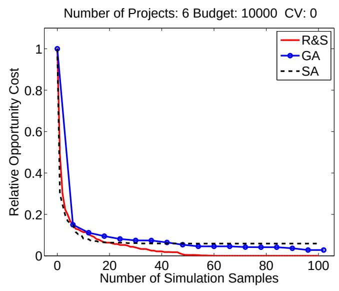
(a) Projects=6, Budget=6000



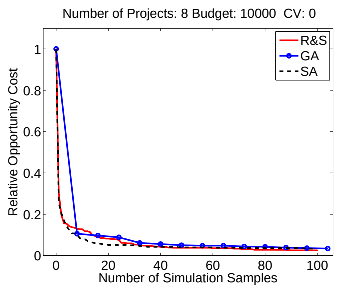
(b) Projects=8, Budget=6000



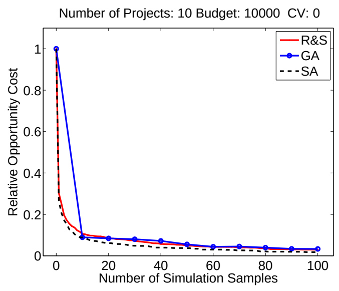
(c) Projects=10, Budget=6000



(d) Projects=6, Budget=10000



(e) Projects=8, Budget=10000



(f) Projects=10, Budget=10000

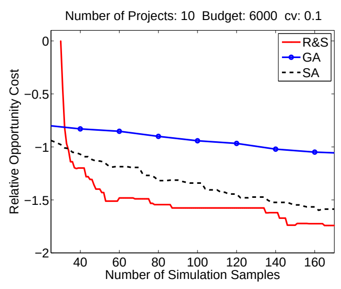
Figure 2.1: Performance comparison for NDP: 100 simulation budget.

Figure 2.1 shows clearly that while all three algorithms performed reasonably well under each scenario, the Bayesian $R\&S$ model out-performs GA and SA in all combinations of project numbers and budget levels. It is more effective in discovering solutions with lower relative opportunity costs (e.g. 0.02. 0,01 or < 0.01) while the SA and GA often converge to locally optimal solutions with RelOC values between 0.05 and 0.10. The superior performance of the Bayesian $R\&S$ model is mainly due to the fact that it places beliefs on the potential rewards of *all* alternatives and fully utilizes the sample history to *direct* future samples via the KGCB sampling policy. The advantage of the Bayesian $R\&S$ model is also noticeably larger in cases with more infeasible solutions (lower budget) and smaller project numbers. This is very indicative of the trade-offs the Bayesian $R\&S$ model make between accuracy and efficiency: by maintaining, updating and inferring about the objective values for all alternatives, the Bayesian $R\&S$ model can avoid local optima and identify globally optimal solutions much more easily. However, under a deterministic setting, as the number of possible alternatives increase, the relative significance of the additional information collected in limited samples is reduced, and the performance of the Bayesian $R\&S$ becomes only marginally better than local-search-based heuristic algorithms for NDP.

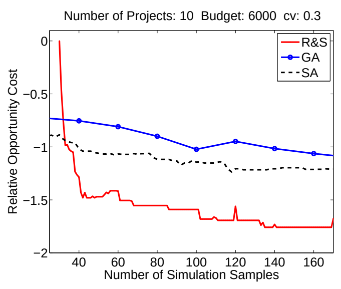
2.3.3 Results for the NDPU (Stochastic) Tests

We then test the performance of the Bayesian R&S formulation on the NDPU problem described in Section 2.1. More specifically, we assume that $d_{rs} : (r, s) \in D$, the demand for each Origin-Destination pair in Equation 2.7, is subject to a $p\%$ perturbation. p follows a normal distribution with mean 0 and standard deviation cv (coefficient of variation), where $cv > 0$ controls the magnitude of uncertainty. p is generated prior to each simulation and applied to all OD pairs in the testing network. If any d_{rs} becomes negative after the perturbation, it is truncated to 0. Other settings for Bayesian $R\&S$ model are the same as in the deterministic case, except that we applied the variance approximation technique described in Equations 2.21 and 2.20. The $\epsilon_{pCount(\mathbf{a})}$ deviation terms in Equation 2.20 were sampled from a normal random variable with mean 0 and a standard deviation of 5% of the corresponding $\hat{\lambda}^{pCount(\mathbf{a})}$.

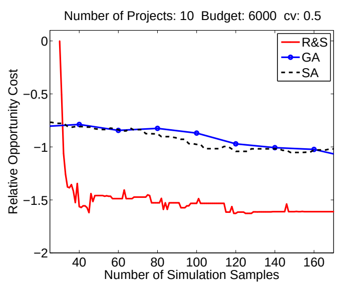
To adjust the GA and SA solver for NDPU, we instead use the mean of a few Monte Carlo samples to approximate the expected objective value for a particular alternative. While probably not the most efficient strategy, this method is representative of the large class of methods (e.g.[79],[82],[73],[72] etc.) which all in some sense used multiple random samples to account for uncertainty. For the experiments we set the number of Monte Carlo samples for the GA algorithm to be 2. We test the performance of the three algorithms through two budgets levels (6000, 1000) and three cv values (0.1, 0.3 and 0.5). The number of projects is set to be 10. Each algorithm is run for 20 repetitions with a budget of 200 evaluations of objective value for each run. The Bayesian $R\&S$ model will use 30 and 40 samples for the 6000 and 10000 budget levels in the first-stage variance estimates. The GA and SA algorithm will use all 200 samples in search of the optimal solution.



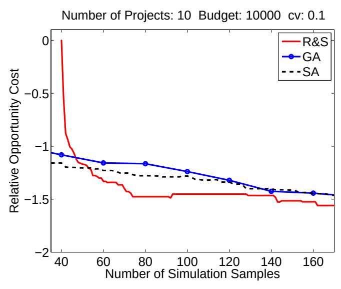
(a) cv=0.1, Budget=6000



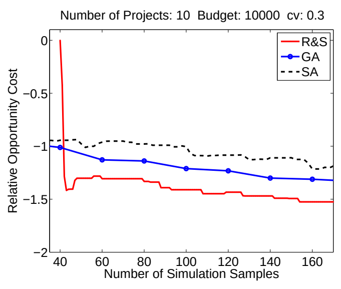
(b) cv=0.3, Budget=6000



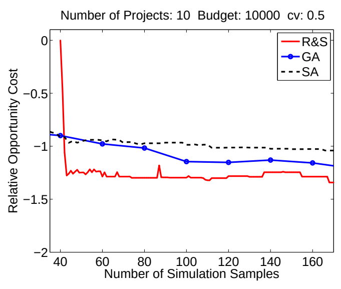
(c) cv=0.5, Budget=6000



(d) cv=0.1, Budget=10000



(e) cv=0.3, Budget=10000



(f) cv=0.5, Budget=10000

Figure 2.2: Performance comparison on NDPU: 150 simulation budget.

Performance comparisons of the GA, SA and the Bayesian $R\&S$ model are presented in Figure 2.2. To better illustrate the difference we use the log (base 10) of the RelOC on the y-axis. Note also that the x-axis of Figure 2.2 starts at 30/40 due to the first stage variance estimation of the Bayesian R&S model. Despite the additional budget spent on variance estimation, the Bayesian $R\&S$ model still catches up very quickly and out-performs both the SA and GA in all scenarios. Across different levels of uncertainty, the performance of the Bayesian R& S model is also more stable. On the other hand, the SA algorithm, which performed rather competitively in the deterministic tests, suffered a significant performance drop when the level of uncertainty increases. This is because the SA algorithm tracks only one solution at any given iteration, and can therefore be easily misguided by randomly large objective value from sub-optimal alternatives. Meanwhile, GA stores a population of alternatives at each iteration, which greatly reduces the impact of random variation from individual solutions. However, the population-based approach adds significant simulation budget in each generation and greatly slows down the progress of the algorithm under a given simulation budget. In the Bayesian R&S model, each sample not only updates the mean estimate of *all* alternatives, but also their *distributions*. By utilizing information about the variances of all alternatives, the Bayesian R&S model properly acknowledges the level of randomness in the objective values and is able to separate alternatives with high variability from the "true" optimal alternatives with high expected objective value.

The benefit of the Bayesian $R\&S$ model is better illustrated in Table 2.3, where we run all three algorithm for a simulation budget of 500 and compare the number of simulations required for the mean RelOC to reach practical levels of optimality (For the Bayesian $R\&S$ model the samples for variance estimation is included in the count). For both budget levels, the Bayesian $R\&S$ model exhibits satisfactory accuracy and greater performance stability under various levels of uncertainty. For a given RelOC, the Bayesian $R\&S$ model can generally reduce the number of necessary simulations by up to $50\% \sim 80\%$ and is often

Table 2.3: Average Sample Number Needed for Different RelOC

B= 6000	**cv: 0.1**			**cv: 0.3**			**cv: 0.5**		
RelOC	**10%**	**5%**	**2.5%**	**10%**	**5%**	**2.5%**	**10%**	**5%**	**2.5%**
GA	140	> 500	> 500	100	420	> 500	140	380	> 500
SA	32	80	178	40	> 500	> 500	104	> 500	> 500
$R\&S$	34	45	135	35	41	102	32	34	116
B= 10000	**cv: 0.1**			**cv: 0.3**			**cv: 0.5**		
RelOC	**10%**	**5%**	**2.5%**	**10%**	**5%**	**2.5%**	**10%**	**5%**	**2.5%**
GA	20	120	200	40	140	> 500	80	200	> 500
SA	18	104	332	52	> 500	> 500	112	> 500	> 500
$R\&S$	44	60	191	42	60	190	44	190	> 500

able to discover low RelOC (i.e. highly optimal) solutions when the other two algorithms failed to make further progress.

2.3.4 Large Network Tests

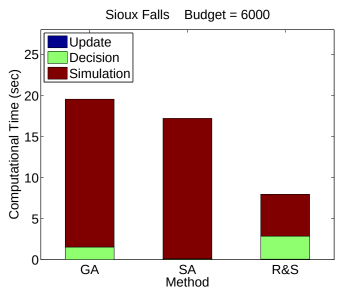
It is evident that the additional book-keeping of the Bayesian $R\&S$ model and KGCB sampling policy can significantly increase the per-iteration computational time. Naturally, if the underlying network is larger and each evaluation of the lower-level problem takes longer, the additional cost of the Bayesian $R\&S$ model will be much more justified. Although of great practical concern, computing times were rarely presented or discussed in previous studies of Bayesian R&S models. In this last experiment we aim to produce some quantitative result about the trade-off between computational time and accuracy for using the Bayesian R&S model. Similar to the stochastic tests, we run the three compet-

Table 2.4: Average Processing and Simulation Times per Iteration

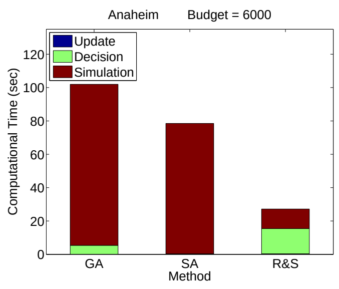
Time (sec)	**Update**	**Decision**	**Network**	**SimTime (sec)**
GA	< 0.0001	0.0127	Sioux Falls	0.15
SA	< 0.0001	0.00079	Anaheim	0.23
R&S (6000/10000)	0.016/0.018	0.72/0.80	Chicago Sketch	3.20

ing algorithms over three networks at the 6000 and 10000 budget levels for 30 replications each. cv is set to 0.1 for the Sioux Falls and Anaheim Networks and (for faster convergence) 0.05 for the Chicago Sketch network. We use $\tau_{0.1}$, the time for each algorithm to reach a mean RelOC of 0.1 as the measure of time efficiency. Table 2.4 summarizes the per-iteration computational time for each algorithm and per iteration "simulation" time needed to evaluate each network. The "update" step includes Equations 2.15 and 2.16 for the Bayesian R&S model and in GA and SA only involves updating the best solution so far. The "decision" step for the Bayesian R&S model is mainly Algorithm 2.4. For GA, it includes the crossover and mutation operations and for SA includes the selection of the next candidate solution and the calculation of the transition probability. All computations were conducted on a workstation with two six-core Intel Xeon processor @2.40Ghz and 32 GB of ram.

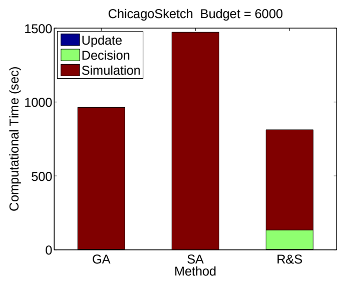
In Figure 2.3 we present the comparison of computational times over the three networks and two budget levels. In each plot we stack the total time spent in sampling(simulation), update and sampling decision for each algorithm. In general, the "simulation" portions of the Bayesian R&S model are much shorter than the other two algorithms, demonstrating a consistent saving of simulation budgets. The Bayesian $R\&S$ model delivered much more competitive computational times in the 6000-budget scenarios for all three networks, as both its



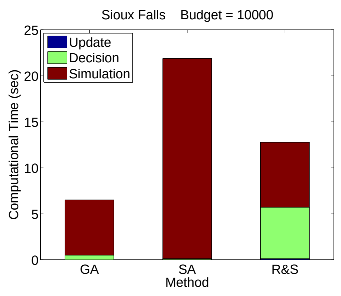
(a) Sioux Falls, Budget=6000



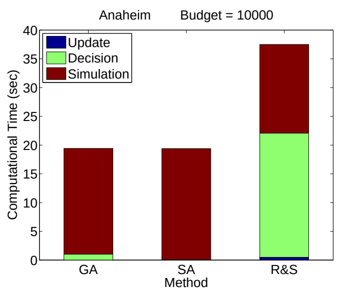
(b) Anaheim, Budget=6000



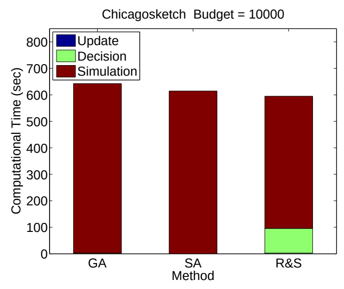
(c) Chicago, Budget=6000



(d) Sioux Falls, Budget=10000



(e) Anaheim, Budget=10000



(f) Chicago, Budget=10000

Figure 2.3: Comparison of time needed to achieve 0.1 RelOC

updating equations sampling policy were computed over fewer alternatives. The performance of the 1000-budget scenarios were less ideal for the Sioux Falls and Anaheim network due to their relatively low computational cost compared with the overhead of Bayesian updates and sampling policies. Nevertheless, the time saved from fewer samples becomes more significant as the computational cost of each simulation exceeds the cost of the Bayesian $R\&S$ model, as is the case for the ChicagoSketch network. Therefore, we can confidently project a substantial time saving when the underlying traffic network requires minutes or hours for each user-equilibrium (i.e. "simulation") evaluation.

2.4 Conclusion

We have developed a Bayesian Ranking and Selection model for the Network Design Problem with Uncertainty (NDPU). With this formulation, each solution to NDPU represents an "alternative" with a prior belief on its expected objective value (i.e. "reward"). Uncertainties in the objective functions are modeled by probabilistic (normal) distributions whose non-uniform variances are estimated and updated from samples. At each iteration, we make sequential decisions about the next alternative to simulate by utilizing reward and correlation information collected in previous iterations. We used constraints in NDPU to pre-eliminate infeasible alternatives and also to defined a constrained version of the KGCB policy. With the Bayesian R&S formulation, we are able to view NDP/NDPU problems from a statistical learning/simulation optimization perspective and focus our effort on directly exploring the inherent relationships between the performances of different network configurations.

The advantages of this new Bayesian $R\&S$ formulation of the NDP/NDPU problem are mainly: 1) It explicitly records and updates the correlations among all candidate solutions to efficiently leverage the information collected from each sample; 2) It enables us to use sequential sampling policies like KGCB, in which later simulation decisions can directly benefit from earlier samples; 3) It models uncertainty with continuous probability distributions, which are more efficient and flexible than discrete scenario sets. Indeed, our computational experiments on the studied networks suggest that the Bayesian R&S model performed better than both the Genetic Algorithm and the Simulated Annealing Algorithm in both deterministic and stochastic settings. The superior performance of the Bayesian $R\&S$ model also implies its potential in solving similar discrete optimization problems such as the Optimal Sensor Location problem [10], Uncapcitated Fixed Charge Facility Location problem [19], etc.

The major limitation of our model is its computational burden in both time and space, due to the fact that it is committed to keeping track of all feasible alternatives: for NDP/NDPU problems, computing time and space can grow exponentially in the number of projects in the worst case. Nevertheless, if the cost of each network simulation is large enough or the budget constraint is tight enough, then the trade-off between simulation time and decision time will generally be worthwhile. As illustrated in our large-scale network tests, if each lower-level user-equilibrium problems takes as little as a few seconds to evaluate, we can expect the the Bayesian R&S formulation to help reduce the total computational burden significantly. For problems with smaller sampling budget, we could also apply a more efficient approximation like [71], in which Bayesian beliefs were maintained only on sampled alternatives.

In conclusion, we believe our Bayesian R&S model will provide valuable insights to the formulation and computation of NDP/NDPU or even other similar discrete optimization problems with expensive objective functions. For future work, we are currently working on a Bayesian $R\&S$ model with parametric beliefs and approximated KGCB policy, which will reduce the computational complexity of both the update and decision step and potentially still retains the model's competitiveness. Meanwhile, we will also seek to expand our methodology to multi-objective NDPU problems.

CHAPTER 3

A BAYESIAN RANKING AND SELECTION MODEL WITH PARAMETRIC BELIEFS AND SURROGATE-ASSISTED KNOWLEDGE GRADIENT POLICY FOR THE DISCRETE NETWORK DESIGN PROBLEM WITH UNCERTAINTIES

3.1 Introduction

The Network Design Problem with Uncertainty (NDPU) is a classical problem in transportation sciences and engineering. Given a network and its users (demand), NDPU optimizes network-wide objective(s) (e.g. total travel time, system capacity, environmental impact, etc.) through a set of candidate "projects" (modifications to the network), subject to randomness in traffic demand, travel cost, etc. The intricacy of NDPU lies in the fact that each "project" not only affects the structure of the network, but also alters the way users utilize the network once they are informed of the changes. To capture such interactions, NDPU problems are typically formulated as Bi-level programs [23], which are known to be NP-hard even in their simplest forms [87].

Earlier studies of NDPU focused on its deterministic version, NDP. Due to its proven complexity, most of these methods adapted alternative formulations which relaxed various aspects of the problem (e.g. objective functions [58], constraints[18], lower-level problems [18], etc.). In recent years, iterative meta-heuristic algorithms such as Genetic Algorithms (GA) [94], Simulated Annealing (SA) [32], Tabu Search[9], Ant Systems[57], etc. have been another popular branch of methodology. A good review of both category of methods can be found in [93] and [59]. Many NDPU solutions methods were derived from their

NDP counterparts plus a mechanism to manage uncertainty. The m ost popular formulation characterizes uncertainty as a probability distribution over a finite scenario set (e.g. [79, 72]). When the size of the scenario set becomes large or infinite (e.g. when the governing probability distribution is continuous), scenario approximation methods such as Single Point Approximation [84, 82], Monte Carlo sampling ([73]) and Sample Average Approximation [72] are applied. Most aforementioned procedures require simulating hundreds to thousands of candidate solutions during the computational process. As NDPU in practice can be imposed on very large networks, there is strong motivation to improve the computational efficiency of NDPU solution algorithms by reducing the number of evaluated solutions.

In a previous study [90], we introduced a Bayesian Ranking and Selection (R&S) model with exact correlated beliefs. In the Bayesian R&S setting, we place a probabilistic prior belief on the expected performance of each NDPU solution (aka. "alternative"). The beliefs are then recursively updated by evaluating alternatives selected by KGCB [29], a value-of-information-based sequential sampling policy [29]. The accuracy of our model proved to be much better than a number of popular heuristic algorithms. However, the expensive computational cost of the update of Bayesian beliefs and the KGCB policy limits the practical problem size to a few thousand alternatives. Although there exists a method ([71]) that maintains beliefs on only the sampled alternatives, its efficiency can only be achieved for relatively small sampling budgets. Moreover, the accuracy of the approximation in [71] would likely never exceed our old study in [90].

In this paper we further improve the computational efficiency of the

Bayesian R&S model with two powerful approximations: 1) a parametric representation of our belief on the expected objective value of each alternative, and 2) A surrogate-assisted knowledge gradient sampling policy which constructs a surrogate optimization problem with the approximated objective function and uses its solutions to constraint the scale of the knowledge gradient calculation. The parametric belief structure allows us to perform learning on the compact parameter space instead of the entire solution space, while the surrogate-assisted sampling policy enables us to neglect the majority of the solution space without losing much insight about the value of information of each sample. Bayesian $R\&S$ models with parametric beliefs has been studied for continuous decision variables in the context of linear programming with unknown coefficients [67] and for special cases with discrete decision variables in drug-discovery [52]. We believe our parametric Bayesian $R\&S$ model and the surrogate-assisted sampling policy complements the previous studies with a more generic and practical framework for large-scale problems with discrete decision variables. Our new method can also be classified as a stochastic Bayesian version of a large class of black-box optimization/simulation optimization methods such as EGO [40], Sequential Kridging [24], Radius basis functions [64, 51, 39], Splines [70], Gaussian Processes [62], etc. Most of the aforementioned methods also utilized parametric approximations of the objective function and sequential "sampling" policies, but generally lack an inherent formulation to characterize complex uncertainty structures. Therefore, our model will provide not only an efficient model for NDP/NDPU, but also valuable insights to a larger class of discrete stochastic optimization problems involving black-box objective functions, response surface modeling and surrogate optimization.

In Section 2, we first describe the traditional formulation of the NDP/NDPU problem and how it can be formulated as a Bayesian Ranking and Selection problem with parametric beliefs. Then we describe the design of our parametric approximation and the surrogate-assisted knowledge gradient sampling policy. Section 3 provides computational examples and Section 4 provides conclusions and discussions.

3.2 Methodology

3.2.1 The Network Design Problem with Uncertainty

Assume a generic network (graph) $G = (V, E)$, with V and E being the vertex set and the edge set respectively. A NDPU is formulated when the decision maker wants to optimize certain objectives (e.g. maximizing throughput, minimizing congestion and etc.) over the network by modifying the edge set E with k improvement "projects". In this paper we focus on discrete NDPU where projects are binary decisions (e.g. link addition). Let the vector $\mathbf{a} = (a_1, a_2, ...a_k)$ be a decision (solution) with each element a_i representing the decision on whether to implement project i, and let $\mathbf{c} = (c_1, c_2, ...c_k)$ be the corresponding project costs. Uncertainty in the NDPU problem is usually characterized by a countable scenario space Ω, with the probability of each scenario $\{p(\omega) : \omega \in \Omega\}$ well defined. Attributes in NDPU (e.g. link flows) can take random values based on ω, and the goal of a NDPU is to choose the decision $\mathbf{a}^*$ that will optimize the expected objective function subject to a budget cap B.

The NDPU problem is usually formulated as a Bi-level mathematical program. The upper level problem models the decision makers' objectives while the lower level models the network users' responsive behavior to changes of the network. Without loss of generality, in this paper we focus on a representative formulation of the NDPU problem to maximize improvement in total travel time over the network, assuming that uncertainty only exists in travel demands. The corresponding formulation is then given as below:

$$\text{(Upper Level)} \max_{\mathbf{a}\in A} \quad \mathbb{E}(T_{\mathbf{a}}) = T_0 - \sum_{\omega\in\Omega} p(\omega) \cdot \sum_{(i,j)\in E\cup E'_{\mathbf{a}}} x^*_{ij}(\omega) t_{ij}(x^*_{ij}(\omega)) \tag{3.1}$$

$$\mathbf{c}^T \cdot \mathbf{a} = \sum_{i=1}^{k} c_i \cdot a_i \leq B \tag{3.2}$$

$$a_i \in \{0,1\}, \forall i \in \{1, 2...k\} \tag{3.3}$$

$$\{x_{ij}(\omega)^* : (i,j) \in E\} \tag{3.4}$$

is the optimal solution for the lower level problem:

(3.5)

$$\text{(Lower Level)} \operatorname*{Min}_{x} \quad \sum_{(i,j)\in E\cup E'_{\mathbf{a}}} \int_0^{x_{ij}} t_{ij}(u)du \tag{3.6}$$

$$\text{s.t.:} \quad \sum_{p\in P_{rs}} f^{rs}_p = d(\omega)_{rs}, \forall (r,s) \in D \tag{3.7}$$

$$x_{ij} = \sum_{(r,s)\in D} \sum_{p\in P_{rs}} f^{rs}_p \cdot \delta^{rs}_{ij,p} \tag{3.8}$$

$$t_{ij} = t^0_{ij} \cdot (1 + \alpha \cdot (\frac{x_{ij}}{C_{ij}})^{\beta}) \tag{3.9}$$

$$f^{rs}_p \geq 0, \forall p \in P_{rs}, \forall (r,s) \in D \tag{3.10}$$

where

A	: the set of all network configurations, i.e. $\{0,1\}^k$
T_0	: expected total travel time on the base network
$E'_{\mathbf{a}}$	: the set of additional links selected by decision **a**.
x_{ij}	: the equilibrium traffic flow on link (i,j).
$t_{ij}(x)$	: travel time on link (i,j) when flow is x
t^0_{ij}	: free-flow travel time on link (i,j)
C_{ij}	: capacity of link (i,j)
α, β	: parameters for calculating travel time
D	: the set of traffic demands indexed by Origin(r) and Destination(s).
P_{rs}	: the set of paths (contiguous links) which starts in node r and ends in node s
f^{rs}_p	: the flow between origin r and destination s on path p
$\delta_{ij,p}$	: equals 1 if link (i,j) belongs to path p and 0 otherwise

Equation 3.1-3.5 define the Upper level problem. Equation 3.1 is the objective function to maximize improvement in total travel time and Equation 3.2 is the budget constraint. Equation 3.6-3.10 specify the lower level problem which is assumed to be the deterministic User Equilibrium problem [75] in this example. In particular, Equation 3.9 assumes the volume-delay function follows the traditional BPR formula, which can be replaced by other volume-delay functions. We can also replace the entire lower level problem by other formulations such as stochastic user equilibrium[75] or system capacity maximization [79].

3.2.2 NDPU as a Bayesian Ranking and Selection Problem

NDP/NDPU as a Bayesian Ranking and Selection Problem

The Bayesian R& S formulation of NDPU treats each network configuration (solution) **a** as an alternative in the finite alternative set (denoted as A) and their expected upper-level objective values as the corresponding "reward" of sampling that alternative. We can think of **a** as a k-digit binary variable, the i_{th} digit of which indicates whether project i will be implemented. The decimal value of this binary variable can also be viewed as an index for the alternative set A and in this paper we will assume elements in A is ordered by this index. As we simulate alternatives $\mathbf{a}_1, \mathbf{a}_2 ... \mathbf{a}_N$, we obtain (perfect) measurements of the random "reward" of those "alternatives', denoted as $T_{\mathbf{a}_1}, T_{\mathbf{a}_2}, ... T_{\mathbf{a}_N}$ and recursively update our beliefs of all alternatives rewards through Bayesian inferences. After we exhausted all N samples, we select the alternative with the best posterior expected reward.

Prior and Likelihood For NDP/NDPU, we further assume our prior belief about the expected objective value of all alternatives follows a multi-variate normal distribution. The sequence of measurements $\mathbf{a}_1 ... \mathbf{a}_N$ follow normal distributions with known variances conditioned on the parameter values:.

$$\boldsymbol{\theta} := (\mathbb{E}(T_1), \mathbb{E}(T_2) ... \mathbb{E}(T_{|A|})) \sim \mathcal{N}(\mu^0, \Sigma^0) \tag{3.11}$$

$$T_{\mathbf{a}_i} | \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}_{\mathbf{a}_i}, \lambda_{\mathbf{a}_i}), \forall i \in 1 ... N \tag{3.12}$$

It is easy to see that the deterministic NDP problem is a special case of the formulation where $\lambda_{\mathbf{a}} = 0, \quad \forall \mathbf{a} \in A$. For most stochastic problems, $\{\lambda_{\mathbf{a}_i}\}_{i=1}^{N}$ is unknown and needs to be approximated/estimated from samples and prior beliefs. It is worth noticing that although the *actual* objective value for each alternative may be correlated, the sequence of measurements on which we place our Bayesian beliefs are drawn from independent simulations. Therefore, the likelihood function of those measurements in Equation 3.12 are all independent.

Parametric Prior We can directly apply the exact, "look-up table" belief structure of Equation 3.11 on θ (as we did in [90]) if $|A|$ is not too large. However, as $|A|$ can be of order 2^k in NDPU in the worst case, it is inefficient to maintain our belief about $\boldsymbol{\theta}$ as a discrete set even for moderate value of k. Instead, we can try to approximate θ as a parametric function of the alternative **a** and place our Bayesian beliefs on the parameters instead. More specifically, we can assume

$$\theta_{\mathbf{a}} = \Phi_{\mathbf{a}}\alpha = \sum_{i=1}^{m} \phi_i(\mathbf{a}) \cdot \alpha_i \quad \forall \mathbf{a} \in A \tag{3.13}$$

$$\text{and} \quad \alpha = (\alpha_1 ... \alpha_m) \sim \mathcal{N}(\beta^0, C^0) \tag{3.14}$$

where $\phi_1(\cdot)...\phi_m(\cdot)$ is a series of deterministic basis functions (e.g. polynomial basis, Fourier basis, Radius basis Function [39], etc.) and α is the vector of parameters we place our normal beliefs on. By the property of multivariate normal distribution, the prior belief induced on $\boldsymbol{\theta}$ via Equation 3.13 and 3.14 becomes

$$\boldsymbol{\theta} = \Phi\alpha \sim \mathcal{N}(\Phi\beta^0, \Phi C^0 \Phi^T) \tag{3.15}$$

where Φ is the $|A|$ by m matrix with elements $\Phi_{ij} = \phi_j(i)$. Equation 3.13 is analogous to a "Value Function Approximation" in the approximated dynamic programming literature [60] or a Response Surface in the design of experiment literature [44]. Equation 3.14 adds the Bayesian interpretation.

Posterior Update of the Parametric Belief From results in recursive least squares for linear regression models [60], the posterior distributions of α will still be normal under the likelihood function of Equation 3.12. Define $\mathbf{a}_n$ as the alternative sampled at iteration n and $\Phi_{\mathbf{a}_n}$ (1 by m) as the $\mathbf{a}_n$ th (in terms of the decimal value of $\mathbf{a}_n$) row of Φ. The corresponding hyper-parameters β^n and C^n can then be recursively updated as:

$$\beta^n = \beta^{n-1} + \frac{\hat{T}_{\mathbf{a}_n} - \Phi_{\mathbf{a}_n}\beta^{n-1}}{\lambda_{\mathbf{a}_n} + \Phi_{\mathbf{a}_n}C^{n-1}(\Phi_{\mathbf{a}_n})^T} C^{n-1}(\Phi_{\mathbf{a}_n})^T \tag{3.16}$$

$$C^n = C^{n-1} - \frac{C^{n-1}(\Phi_{\mathbf{a}_n})^T \Phi_{\mathbf{a}_n} C^{n-1}}{\lambda_{\mathbf{a}_n} + \Phi_{\mathbf{a}_n}C^{n-1}(\Phi_{\mathbf{a}_n})^T} \tag{3.17}$$

As β^0 and C^0 are updated to β^n and C^n, the posterior belief induced on $\boldsymbol{\theta}$ is implicitly updated to $\mathcal{N}(\Phi\beta^n, \Phi C^n \Phi^T)$ as well. The clear advantage of updating β^n and C^n is its computational efficiency compared with updating μ^n and Σ^n directly: β^n and C^n are only of size m and m $\times$ m respectively where m is typically polynomial to the number of projects k. The potential downside, however, is that $\Phi\beta^n$ is only an approximation of the true reward θ, and there is typically no analytical guidelines to determine the suitable structure of the basis function.

Construction of Basis Functions As discussed in [60], the design and construction of a parametric Value Function Approximation is usually problem-specific. Our parametric prior for NDPU is based on two observations:

- **Linear Dependence.** The "reward" of a particular alternative (i.e. a collection of projects) is usually correlated with the "reward" of the individual active projects in that alternative.
- **Diminishing Return.** The "reward" of an individual project is usually "discounted" when implemented together with other projects. The discount factor may grow as the number of project increases.

Therefore, it is natural to model the "reward" of a particular alternative in the NDPU as a discounted linear combination of the "reward" of each individual active project. Indeed, for a NDPU problem with a maximum of k projects, our proposed parametric approximation is:

$$\hat{\theta}_{\mathbf{a}} = \beta_0 + \sum_{i=1}^{k} \beta_i \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b}} \cdot \mathbb{1}_{\{a_i=1\}}, \quad \forall \mathbf{a} \in A \tag{3.18}$$

Here $\{\beta_1, ...\beta_k\}$ can be interpreted as the "main effect" of including each project in an alternative, and β_0 is a nuisance "intercept" term we incorporated to increase the flexibility of the approximated objective function. Our basis function for $\{\beta_1, ...\beta_k\}$ consists of two parts: a binary indicator $\mathbb{1}_{\{a_i=1\}}$ to label the active projects in the alternative, and a Gaussian discount factor $e^{-\frac{\|\mathbf{a}\|^2-1}{b}}$ which depends on the number of projects in alternative **a** (i.e. $\|a\|$) and a "bandwidth" parameter b.

The structure of the Gaussian-binary basis functions presents a straightforward method to set up an informative prior for $\beta^0 = (\beta_0^0 ... \beta_k^0)$. Since the discount factor is 1 when there is only one active project in an alternative (i.e. when $\|a\|^2 = 1$), we can set $\beta_0^0 = 0$, generate sample estimates of the expected objective values of each individual projects $\{\hat{\theta}_{2^i}\}_{i=0}^{k-1}$, and set $\beta_i^0 = \hat{\theta}_{2^{i-1}}, \forall i \in 1...k$. The bandwidth b can also be estimated by simulating an additional $\hat{\theta}_{2^k-1}$ (i.e. the alternative with all projects implemented). According to Equation 3.18, we will have

$$\hat{\theta}_{2^k-1} = \beta_0^0 + \sum_{i=1}^{k} \beta_i^0 \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b}} \cdot \mathbb{1}_{\{a_i=1\}} = \sum_{i=1}^{k} \hat{\theta}_{2^{i-1}} \cdot e^{-\frac{k-1}{b}} \tag{3.19}$$

$$\Leftrightarrow \hat{b} = \frac{1}{1-k} \cdot \ln\left(\frac{\hat{\theta}_{2^k-1}}{\sum_{i=1}^{k} \hat{\theta}_{2^{i-1}}}\right) \tag{3.20}$$

Estimating b and β^0 would require an additional n(k+1) samples, where n is the number of replications needed to generate relatively reliable $\{\hat{\theta}_{2^i}\}_{i=0}^{k-1}$. Here we assume the value of b will be fixed for all subsequent iterations, but we can definitely update it recursively through Equation 3.20 if desirable. Though a non-trivial overhead, $\hat{b}$ and $\{\hat{\theta}_{2^i}\}_{i=0}^{k-1}$ can be re-used in replication runs. As it is common practice to run randomized optimization/learning algorithms for multiple replications for enhanced stability, the average computational cost distributed to each replication will be much less significant.

Estimation and Approximation of Alternative Variances Estimating and approximating the variances of alternatives is an important aspect of Bayesian R&S models and knowledge gradient sampling policies. In our model, since we have already used multiple samples to generate $\{\hat{\theta}_{2^i}\}_{i=0}^{k-1}$, we can use the very same samples to estimate their respective variances $\{\lambda_{2^i}\}_{i=0}^{k-1}$. Similar to our belief on θ, we can put a parametric belief structure on the "prior" of the variance of any alternative by setting

$$\hat{\lambda}_{\mathbf{a}}^0 = \Phi_{\mathbf{a}}^{\lambda}\beta^{\lambda} = \sum_{i=1}^{k} \hat{\lambda}_{2^{i-1}} \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b_\lambda}} \cdot \mathbb{1}_{\{a_i=1\}} \quad \forall \mathbf{a} \in A \tag{3.21}$$

The corresponding bandwidth b_λ can be estimated in a similar fashion to Equation 3.20. As the algorithm proceeds, we can collect more samples for a particular alternative **a** and update our "posterior" estimate of $\lambda_{\mathbf{a}}$ as in [90]. That is, for an iteration $n \in 1...N$ when we sampled alternative **a**, we set:

$$\hat{\lambda}_{\mathbf{a}}^n = \frac{v_0\hat{\lambda}_{\mathbf{a}}^0 + \sum_{i=1}^{N_{\mathbf{a}}}(y_{\mathbf{a}}^{n(i)} - \mu_{\mathbf{a}}^{n(i)})^2}{v_0 + N_{\mathbf{a}} - 2} \tag{3.22}$$

where $N_{\mathbf{a}}$ is the number of samples collected for alternative **a** at iteration n, $n(i)$ is the iteration at which the i_{th} sample of alternative **a** is collected, $\mu_{\mathbf{a}}^{n(i)}$ is the posterior mean for alternative **a** at iteration $n(i)$ and v_0 is a non-negative weight parameter. To marginalize the inaccuracy from the posterior means, we require $n_{\mathbf{a}}$ to be at least larger than a threshold n_{min} (typically 3-5) before Equation3.22 can be applied. Equation 3.22 can be viewed as a weighted average of the prior estimate in Equation 3.21 and the subsequent sample estimate. As the number of samples accumulates, it is easy to see that the variance estimates will converge to the true variances.

Decision After we have exhausted all N samples, we select the network configuration with the best posterior reward as the proposed optimal solution:

$$\mathbf{a}_N^* \in \arg\max_{\mathbf{a}\in A^f} \mathbb{E}(T_{\mathbf{a}}|\mathcal{F}^N) = \arg\max_{\mathbf{a}\in A^f} \mu_{\mathbf{a}}^N = \arg\max_{\mathbf{a}\in A^f} \Phi\beta^N \tag{3.23}$$

Here $A^f \subset A$ denotes the set of feasible solutions in A. As $|A^f|$ becomes large, Equation 3.23 itself becomes a discrete optimization problem. However, as the objective function in Equation 3.23 is deterministic and computationally cheap, we can afford to run a generic iterative algorithm like Simulated Annealing [43] or Genetic Algorithms for many iterations/replications to obtain high-accuracy solutions. With the power of modern computers, the optimal solution can even be enumerated in a few seconds for practical-sized NDPU problems with millions of network configurations. $\mathbf{a}_N^*$ will often be different from the true optimal solution $\mathbf{a}^* \in \arg\max_{\mathbf{a}} \boldsymbol{\theta}_{\mathbf{a}}$ due to two sources of errors: 1)the inaccuracies from the approximated parametric belief, and 2) the imprecision due to imperfect knowledge from finite samples. Therefore, our goal in practice is usually to identify $\mathbf{a}_N^*$ such that $\boldsymbol{\theta}_{\mathbf{a}_N^*}$ is as close as possible to $\boldsymbol{\theta}_{\mathbf{a}^*}$.

3.2.3 Solving NDP/NDPU with Bayesian R&S Formulation

Overview of Solution Procedure

The solution procedure for NDP/NDPU with parametric Bayesian Ranking and Selection formulations can be summarized in Algorithm 3.1 below. In short, we iteratively update our Bayesian belief about the alternative rewards through sequential simulation decisions. After exhausting all evaluation budgets, we select the feasible alternative with the best expected posterior reward. In next section we describe the surrogate-assisted knowledge gradient sampling policy used on Line 3 of Algorithm 3.1.

Algorithm 3.1: Solving NDPU with Bayesian $R\&S$ formulation

```
Require: Inputs $\beta^0$, $C^0$, budget cap B, sampling budget N
1: for $n = 1 \to N$ do
2:     Determine $\mathbf{a}_n$, the next sample to simulate via a sampling policy.
3:     Simulate network configuration $\mathbf{a}_n$ and obtain objective value $\hat{T}_{\mathbf{a}_n}$
4:     Update alternative variance $\hat{\lambda}^n_{\mathbf{a}_n}$ using Equation 3.22
5:     Given $\mathbf{a}_n$, $\hat{\lambda}^n_{\mathbf{a}_n}$ and $\hat{T}_{\mathbf{a}_n}$, update posterior belief $(\beta^{n-1}, C^{n-1})$ to $(\beta^n, C^n)$ with
   Equations 3.16 and 3.17.
6: end for
7: return $\hat{\mathbf{a}}^*_N \in \arg\max_{\mathbf{a} \in A^f} \Phi\beta^N_{\mathbf{a}}$
```

Surrogate-Assisted Knowledge Gradient Sampling Policy

The KGCB Sampling Policy Our surrogate-assisted policy is inspired by a value of information sampling policy known as the Knowledge Gradient policy with Correlated Beliefs (KGCB) [29]. Given an intermediate iteration n, the KGCB policy selects the alternative with the largest "Knowledge Gradient". In the case of the constrained NDPU, it is presented in [90] that we should choose:

$$\mathbf{a}_f^{KG,n} \in \arg\max_{\mathbf{a}\in A} v_f^{KG,n}(\mathbf{a}) = \arg\max_{\mathbf{a}\in A} \mathbb{E}^n(\max_{i\in A^f} \mu_i^{n+1} | \mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}) - \max_{i\in A^f} \mu_i^n \tag{3.24}$$

The knowledge gradient of each alternative represents the expected improvement in the posterior optimal value from sampling that alternative. By calculating the conditional *predictive* expectation of $max_i \mu_i^{n+1}$ for each alternative, we can project the value of information of all alternatives without taking actual samples of them. The KGCB policy can be described in NDPU notations in Algorithm 3.2. For each feasible alternative, Algorithm 3.2 computes its knowledge gradient (i.e. $v_f^{KG}(\mathbf{a})$) and records the solution with maximum $v_f^{KG}(\mathbf{a})$. The time complexity of Algorithm 3.2 is $O(\log(|A^f|)|A^f||A|)$, which can be a quite substantial as $|A|$ grows to a few thousand.

Algorithm 3.2: The KGCB policy for NDPU with parametric belief

```
Require: Inputs $\beta^n$, $\Phi$ and $C^n$.
 1: for each $\mathbf{a} \in A'$ do
 2:     $p \leftarrow (\Phi\beta^n)_{A^f}$, $q \leftarrow (\Phi C^n \Phi' \cdot e_{\mathbf{a}})_{A^f} / \sqrt{\lambda_{\mathbf{a}} + (\Phi C^n \Phi')_{\mathbf{aa}}}$
 3:     (v$_{A^f}$ returns the feasible sub-vector of the vector v)
 4:     Sort the sequence of pairs $(p_i, q_i)_{i=1}^{|A^f|}$ so that the $q_i$ are in non-decreasing
        order and ties in q are broken so that $p_i \leq p_{i+1}$ if $q_i = q_{i+1}$.
 5:     Remove all entry i in $(p_i, q_i)_{i=1}^{|A^f|}$ where $q_i = q_{i+1}$
 6:     $c_0 \leftarrow -\infty$, $c_{|A^f|} \leftarrow +\infty$, $c_i \leftarrow -(p_{i+1} - p_i)/(q_{i+1} - q_i)$, $\forall i \in 1...|A^f| - 1$
 7:     Remove all entry i in $(p_i, q_i, c_i)_{i=1}^{|A^f|}$ where $c_i \geq c_{i+1}$
 8:     $v^{KG}(\mathbf{a}) \leftarrow \log(\sum_{i=1}^{|A^f|-1} (q_{i+1} - q_i)(\varphi(-|c_i|) - |c_i|\Phi(-|c_i|)))$
 9:     ($\varphi(.)$ and $\Phi(.)$ is the pdf and the cdf of a standard normal variable)
10:     return $\mathbf{a}_n = \arg\max_{\mathbf{a} \in A^f} v_f^{KG}(\mathbf{a})$
11: end for
```

Surrogate-Assisted Knowledge Gradient Sampling Policy The computational burden of Algorithm 3.2 comes both from the outer for-loop that iterates over all $|A|$ feasible solutions and the individual knowledge gradient calculation from line 3 to line 8 of Algorithm 3.2 which manipulates the full mean vector p of size $|A^f|$. However, empirical studies [65] have suggested that measurement policies like KGCB often spend most of their efforts among a small group of "promising" alternatives. [66] proposed a Monte-Carlo knowledge gradient sampling policy in which K Monte Carlo samples of the (stochastic) posterior means are simulated and only the optimal solution in each simulation is included in the subsequent knowledge gradient calculation. This approximation reduces the computational cost of Algorithm 3.2 to $O(\ln(K)K^2)$, but incurs the additional cost of simulating the a full random posterior mean vector for K times. For our NDPU study, the mean vector is usually too large to be fully simulated, so we instead solve the auxiliary "surrogate" optimization problem for its optimal solution:

$$\max_{\mathbf{a}} \quad (\Phi\beta^n)_{\mathbf{a}} \tag{3.25}$$

$$s.t. \quad \mathbf{c} \cdot \mathbf{a} \leq B \tag{3.26}$$

Equations 3.25 and 3.26 form a simplified NDPU problem with a deterministic and explicit objective function. They are sometimes known as surrogate models in the heuristic optimization literature (e.g. [1]). Solving Equations 3.25 and 3.26 would give us a "surrogate" optimal solution $\hat{\mathbf{a}}_n^*$. Rather than using the random outcome of stochastic simulations to maintain diversity of the candidate set as [65] did, we propose to include alternatives in the neighborhood of $\hat{\mathbf{a}}_n^*$ instead. Define a d-degree neighborhood of an alternative $\mathbf{a}$ as:

$$N^d(\mathbf{a}) := \{i \in A : \sum_{j=1}^{k} (\mathbf{a}_j \oplus i_j) \leq d\} \tag{3.27}$$

Where $\oplus$ is the XOR operator which takes value 0 when $\mathbf{a}_j = i_j$ and 1 otherwise. In other words, $N^d(\mathbf{a})$ is the set of alternatives with at most d different project status from $\mathbf{a}$. We can then construct the localized approximate knowledge gradient policy over $N^d(\mathbf{a})$ as

$$\hat{\mathbf{a}}_f^{KG,n} \in \arg\max_{\mathbf{a} \in N^d(\mathbf{a})} \hat{v}_f^{KG,n}(\mathbf{a}) \tag{3.28}$$

$$= \arg\max_{\mathbf{a} \in N^d(\mathbf{a})} \mathbb{E}^n\left(\max_{i \in N^d(\mathbf{a}) \cap A^f} \mu_i^{n+1} | \mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}\right) - \max_{i \in N^d(\mathbf{a}) \cap A^f} \mu_i^n \tag{3.29}$$

As shown in Equation 3.29, $\hat{v}_f^{KG,n}(\mathbf{a})$ is defined as the one-step expected improvement in posterior optimal value in the localized region of $N^d(\mathbf{a})$ if alternative $\mathbf{a}$ is sampled. We can also quickly verify that $\hat{\mathbf{a}}_f^{KG,n} = \mathbf{a}_f^{KG,n}$ if and only if

$\mathbf{a}_f^{KG,n} \in N^d(\hat{\mathbf{a}}_f^{KG,n})$ and $\arg\max_{\mathbf{a}\in A^f}(\mu_{\mathbf{a}}^{n+1}|\mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}_f^{KG,n}) \in N^d(\hat{\mathbf{a}}_f^{KG,n})$. Nevertheless, since most solutions in $N^d(\hat{\mathbf{a}}_n^*)$ have relatively high posterior means, they are likely highly correlated with (or even contain) the set of potential posterior optimal solutions at iteration n+1. Therefore, a large $\hat{v}_f^{KG,n}(\mathbf{a})$ value would be a good indication of a large expected increment in the true posterior optimal value (i.e. $v_f^{KG,n}(\mathbf{a})$) as well, ensuring the relative effectiveness of our approximated sampling policy at Equation 3.28.

Algorithm 3.3: Surrogate-assisted knowledge gradient policy for NDPU

```
Require: Inputs $\beta^n$, $\Phi$, $C^n$. Neighborhood degree d.
1: Solve the surrogate problem (Equations 3.25 and 3.26) to obtain $\hat{\mathbf{a}}_n^*$
2: Generate the index set for $N^d(\hat{\mathbf{a}}_n^*)$, the d-degree feasible neighborhood of $\hat{\mathbf{a}}_n^*$.
3: for each $\mathbf{a} \in N^d(\hat{\mathbf{a}}_n^*)$ do
4:     Calculate $\hat{v}_f^{KG,n}(\mathbf{a})$ on $N^d(\hat{\mathbf{a}}_n^*)$ using Algorithm 3.2
5:     return $\mathbf{a}_n = \arg\max_{\mathbf{a}\in N^d(\hat{\mathbf{a}}_n^*)} \hat{v}_f^{KG,n}(\mathbf{a})$
6: end for
```

The surrogate-assisted knowledge gradient sampling policy then follows in Algorithm 3.3. $\hat{v}_f^{KG,n}(\mathbf{a})$ in Equation 3.28 can be calculated like $v_f^{KG,n}(\mathbf{a})$ by passing only the sub-vector containing elements of $N^d(\hat{\mathbf{a}}_n^*)$ as inputs to Algorithm 3.2. Algorithm 3.3 reduces the computational complexity of the sampling policy to only $O(\ln(|N^d(\hat{\mathbf{a}}_n^*)|)|N^d(\hat{\mathbf{a}}_n^*)|^2) = O(d\ln(k)k^{2d})$, where typically $d \leq 2$. As Algorithm 3.2 can efficiently handle vector size of a few thousand, Algorithm 3.3 will remain tractable for NDPU with at least 40∼50 projects and billions of feasible solutions.

3.3 Computational Examples

3.3.1 Data Source and Simulation Design

For our case studies we use the famous Sioux Fall network and a larger Anaheim network [3]. Data for both networks are publicly available at [3]. Summary information about the two networks is presented in Table3.1. The absolute size of the network is moderate but they are already quite computationally expensive for NDPU as we need to solve the corresponding lower level User Equilibrium problem for hundreds of iterations and tens of repetitions in the experiments. For the sources of uncertainty, we assume that $d_{rs} : (r, s) \in D$, the demand for each Origin-Destination pair in Equation 3.7, is subject to a $p\%$ perturbation. p follows a normal distribution with mean 0 and standard deviation cv (coefficient of variation), where $cv > 0$ controls the magnitude of uncertainty. p is generated prior to each simulation and applied to all OD pairs in the testing network. If any d_{rs} becomes negative after the perturbation, it is truncated to 0. On top of the base network, we define 20 candidate projects, whose (fictitious) costs are summarized in Table 3.2. Projects are typically constructed as bi-directional links that connect OD paths with high volumes. The time for each User Equilibrium evaluation in Table 3.1 as well as the other computational times later reported are based on a workstation with two quad-core Intel Xeon processor @2.40Ghz and 12 GB of ram.

The primary performance measure of our experiments is the Relative Opportunity Cost (RelOC) of each algorithm. RelOC is defined as the percentage difference between the expected optimal value of the true optimal solution and the expected objective value evaluated at the "best" alternative proposed by

Table 3.1: Size of Networks

Network Name	**# of Nodes**	**# of Links**	**# of OD Pairs**	**Sim Time(sec)**
Sioux Fall	24	76	576	0.096
Anaheim	416	914	1444	0.202

Table 3.2: List of Projects

ProjectID	**Budget ($)**	**ProjectID**	**Budget ($)**
1	1800	2	1500
3	1000	4	1950
5	1650	6	2100
7	1200	8	625
9	650	10	850
11	100	12	1250
13	1300	14	1450
15	1600	16	750
17	850	18	950
19	1150	20	1250

the algorithm when it terminates. Thus, once the RelOC is zero, the algorithm has found the best solution possible. We also define the RelOC of all infeasible solutions to be 1, as recommending an infeasible solution does not help our decision-making at all. For our Bayesian R&S model, RelOC is a more reliable performance measure than the traditional "difference in objective values" measure, since each of our samples has the potential to randomly generate large optimal values. In our experiments, the "true" objective value for each network configuration is approximated by the mean of 50 Monte Carlo samples.

For all tests, the lower level User Equilibrium problem (Equation 3.6-3.10) is solved by an open-source solver [78] with the relative gap of convergence fixed at 10^{-6}. The implementation of the KGCB subroutine (i.e. Algorithm 3.2) is based on the MatlabKG library available at [30]. We use a modified Simulated Annealing algorithm [43] as the solver for our auxiliary surrogate optimization problem at each iteration. The modified SA algorithm will reject all infeasible solutions in the neighborhood of the candidate solution during the state transition calculation. The initial "temperature" of the SA algorithm is set to 100, and decreases by 5% after each simulation evaluation. The degree of neighborhood for each solution is set to 3. Except for the GA solver which is coded in C, all other computational procedures are coded in Matlab.

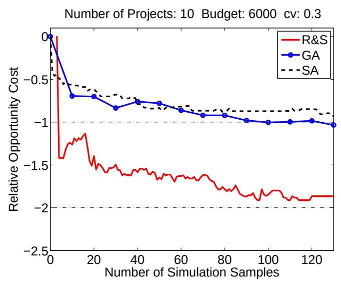
For bench-marking, we solve the same problems using Genetic Algorithm (GA) and Simulated Annealing, both of which are very popular for solving NDP/NDPU and other similar black-box optimization problems. The Genetic Algorithm uses evolutionary strategies to evolve a population of alternatives to the optimal solution while Simulated Annealing generates a random walk in the solution space to approach the optimal solution. For the GA algorithm we use the package from [21]. The population size of the GA solver is set to the number of projects in the test, the crossover probability is set to 0.9, and the mutation probability is set to 0.02. Infeasible solutions are penalized in their objective value in the GA solver. The SA algorithm is identical to the one used to solve the surrogate problem.

3.3.2 Computational Performance

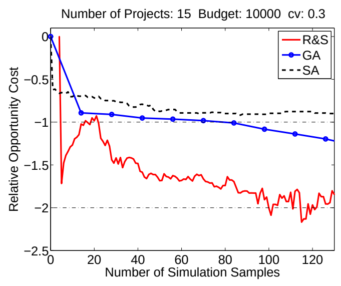
In our first experiment we compare the computational performance of the Bayesian R&S model with parametric beliefs over different project numbers and uncertainty levels. For each of our two testing networks, we test the algorithm under three scenarios: 10 projects with cv=0.3, 15 projects with cv=0.3, and 20 projects with cv=0 (we were unable to test either network for uncertainty scenarios with 20 projects due to the months of computational time expected to simulate the "true" expected objective value of every alternative for multiple replications). The corresponding budget levels are 6000 and 10,000 for the 10 project scenarios and 10,000 and 15,000 for the 15 and 20 project scenarios. They correspond to 398/1023, 950/1023, 16,645/32,768, 32,544/32,768, 313,267/1,048,575 and 942,259/1,048,575 feasible solutions respectively. All three algorithms (R&S, GA and SA) were executed for 20 replications for each of the six scenarios. For our parametric Bayesian $R\&S$ model, we use 5 replications per project to estimate β^0 and b for stochastic scenarios and 1 sample per project for deterministic ones. Consequently, the initial estimation requires an additional 50, 75, and 20 samples respectively for our three project/uncertainty levels, which is prorated to an overhead of less than 5 samples per replication. In this experiment, the surrogate optimization problem is solved for 1000 iterations and the degree of neighborhood is set to 2 for the subsequent approximated knowledge gradient calculation.

Performance comparisons of the GA, SA and the Bayesian R&S model are presented in Figure 3.1. To better illustrate the difference we use the log (base 10) of the RelOC on the y-axis. Reference lines are plotted at $y = -1$ and $y = -2$, which indicate 10% and 1% RelOC respectively. It is evident that the Bayesian

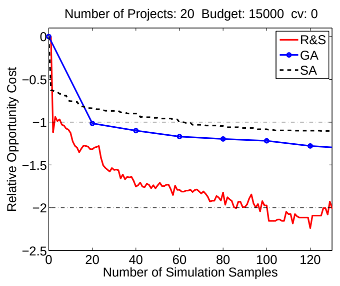
Sioux Falls:



(a) k=10, B=6000, cv=0.3

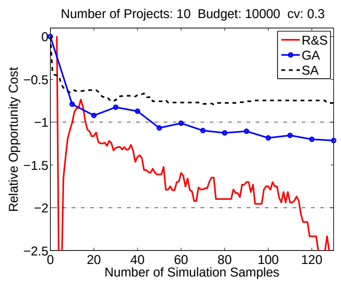


(b) k=15, B=10000, cv=0.3

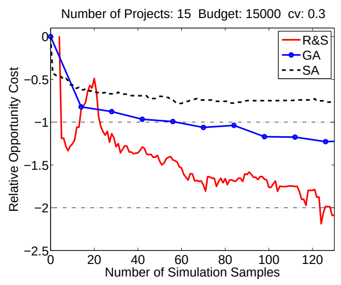


(c) k=20, B=15000, cv=0

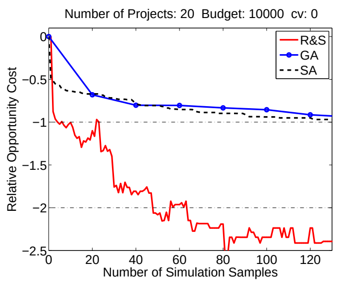
Anaheim:



(d) k=10, B=10000, cv=0.3



(e) k=15, B=15000, cv=0.3



(f) k=20, B=10000, cv=0

Figure 3.1: Performance comparison for NDPU: 120 simulation budget

$R\&S$ model significantly out-performs both the SA and GA algorithm in all scenarios. It converges to around 1% of the true optimal solution in about 100 iterations, while GA and SA have only approached $7 \sim 10\%$ RelOC. A recurring feature for the convergence path of the Bayesian R&S model in Figure 3.1 is the initial "spike" of RelOC around the first 20-30 iterations. This behavior, well documented in the approximate dynamic programming literature [60], represents a learning phase when the approximated Bayesian parametric belief actively adjusts the parameter values to mimic the true objective function. On the other hand, the parametric belief structure provides great flexibility to account for randomness in the samples and inaccuracies in the approximations, resulting in very stable performances across all budget levels, uncertainty levels, project numbers, even in presence of the approximated alternative variances.

Table 3.3: Average Number of Samples Needed for Different RelOC Levels

SiouxFalls	**k=10, B=6000, cv=0.3**			**k=15, B=10000, cv=0.3**			**k=20, B=15000, cv=0**		
RelOC	**5%**	**2%**	**1%**	**5%**	**2%**	**1%**	**5%**	**2%**	**1%**
GA	$>$1000	$>$1000	$>$1000	196	520	$>$1000	140	440	$>$1000
SA	$>$1000	$>$1000	$>$1000	$>$1000	$>$1000	$>$1000	367	$>$1000	$>$1000
$R\&S$	16	75	142	28	85	137	24	40	101

Anaheim	**k=10, B=10000, cv=0.3**			**k=15, B=15000, cv=0.3**			**k=20, B=10000, cv=0**		
RelOC	**5%**	**2%**	**1%**	**5%**	**2%**	**1%**	**5%**	**2%**	**1%**
GA	170	810	$>$1000	154	490	$>$1000	460	$>$1000	$>$1000
SA	$>$1000	$>$1000	$>$1000	$>$1000	$>$1000	$>$1000	903	$>$1000	$>$1000
$R\&S$	37	75	114	36	105	129	28	30	64

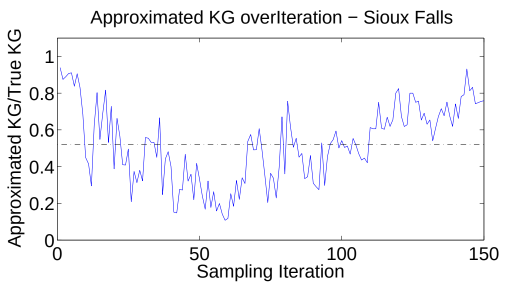
The efficiency of the Bayesian $R\&S$ model is better summarized in Table 3.3, where we run all three algorithms for a larger simulation budget of 1000 and compare the number of simulations required for the mean RelOC to reach (i.e. never rise above) practical levels of optimality. For all scenarios, the Bayesian $R\&S$ model uses less than 150 samples to reach an RelOC level of 1%. On the other hand, the GA and SA algorithm converged very slowly beyond $\%5$ RelOC, making little to no progress thereafter for several hundred iterations. As the total number of alternatives and the level of uncertainty increases, local search algorithms like GA and SA can suffer from their " myopia" and are more likely to be trapped in local optima due to both the large state space and the stochastic objective values. In contrast, our Bayesian $R\&S$ model is able to keep track of the *distributions* of *all* alternatives through the updated parametric beliefs. Our model accounts for the shape of the objective function as well as the level of variability for each alternative, and is therefore able to identify highly optimal solutions in large solution spaces and high uncertainty level.

Table 3.4: Average Processing Time (sec) per Iteration

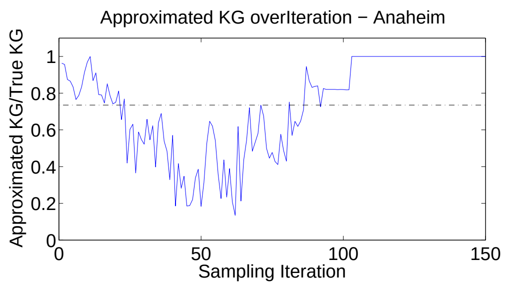
Time (sec)	**k=10**	**k=15**	**k=20**
GA	0.00299	0.00202	0.00184
SA	0.00017	0.000176	0.000195
R&S	0.440	0.445	0.512
Surrogate	*0.413*	*0.39*	*0.417*
Policy	*0.0271*	*0.055*	*0.0951*
Update	*0.00029*	*0.00036*	*0.00041*

Table 3.4 further compares the per-iteration computational overhead of the three algorithms. Although the R&S model takes about more time to solve the surrogate model and calculate the approximated KG policy, its faster convergence speed can potentially save hundreds of simulations/objective value evaluations, each of which already takes 0.1∼0.2 sec for our toy examples. In addition, the computational time of the Bayesian R&S model is dominated by the surrogate optimization procedure. As the SA surrogate solver (as well as most heuristic solvers which we could substitute) applies a myopic local search heuristic, the computational cost of the entire Bayesian $R\&S$ model grows very slowly as the number of project increases.

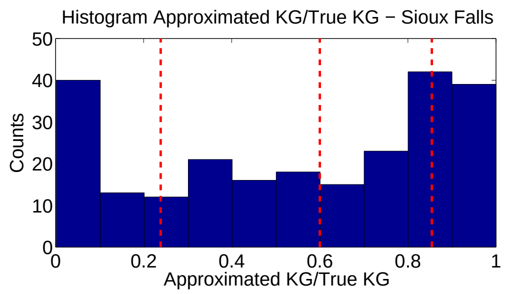
Lastly, in Figure 3.2 we present the relative efficiency of the surrogate-assisted KGCB algorithm (Algorithm 3.3) with respect to the exact KGCB policy (Algorithm 3.2). For this comparison we use the two previous scenarios with 10 projects, whose exact knowledge gradients are still tractable. Each scenario was run for 5 replications. In each sampling iteration, we first propose a solution via the surrogate-assisted KGCB policy, then calculate another solution from the exact KGCB policy, and finally take the ratio of their knowledge gradients as the



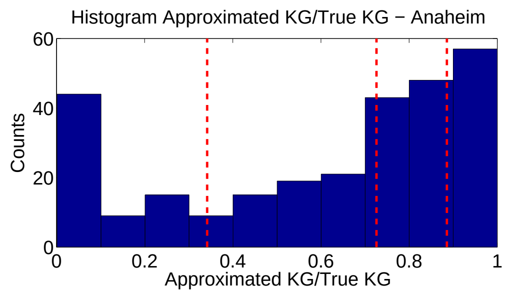
(a) VOI ratio over iterations, Sioux Falls



(b) VOI ratio over iterations, Anaheim



(c) Histogram of VOI ratio, Sioux Falls



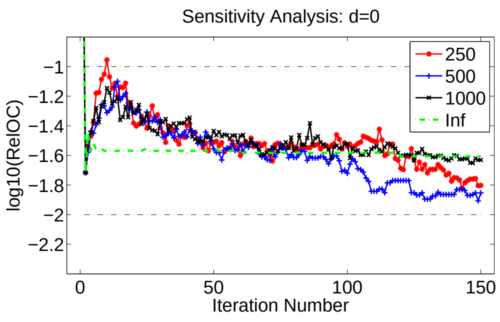
(d) Histogram of VOI ratio, Anaheim

Figure 3.2: Value Of Information (VOI) ratio $(\hat{v}^{KG}(\cdot)/v^{KG}(\cdot))$ of the surrogate-assisted KGCB policy

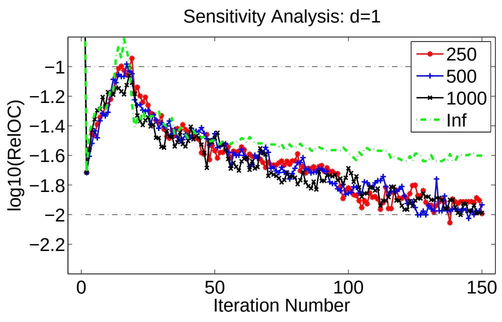
measure of relative effectiveness. As Figure 3.2 (a) and (b) illustrate, the ratios display a "V" shape along the sampling iterations. This is likely due to that most learning algorithms tend to make smaller progresses in intermediate iterations, and therefore the surrogate-assisted KGCB policy has a higher chance to miss a few insignificant improvements. In Figure 3.2 (c) and (d) we display the VOI ratio histogram of iterations where the exact KGCB has improved the objective value by at least 0.1%. As illustrated by the red dashed quartiles lines, the surrogate-assisted KGCB policy is able to generate a median VOI ratio of 0.600 and 0.726 for the two networks respectively. Given that the approximated knowledge gradients are computed on a 2nd-degree neighborhood of 56 alternatives, the surrogate-assisted KGCB policy have roughly reached $60\% \sim 70\%$ of the exact KGCB policy's performance using only 5.5% (i.e. 56/1023) of the available alternatives.

3.3.3 Sensitivity Analysis

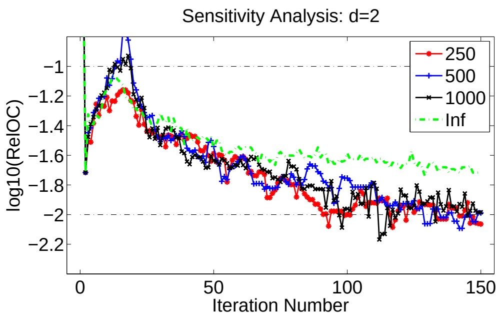
In our next set of experiments we examine the impact of key parameters in our surrogate-assisted knowledge gradient policy, namely the number of iterations in each surrogate optimization run and the degree of neighborhood d for the approximated knowledge gradient calculation. We fix our testing scenario as the Sioux Falls network with 15 projects, 0.3 cv and 10000 budget. We then run our Bayesian $R\&S$ model over the 16 combinations of 4 degree of neighborhood (0,1,2,3) and 4 surrogate iterations (250, 500, 1000, ∞). When the surrogate iteration is ∞, we enumerate the true surrogate optimal solution.



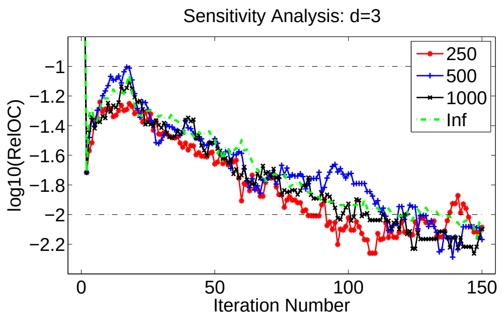
(a) d=0



(b) d=1



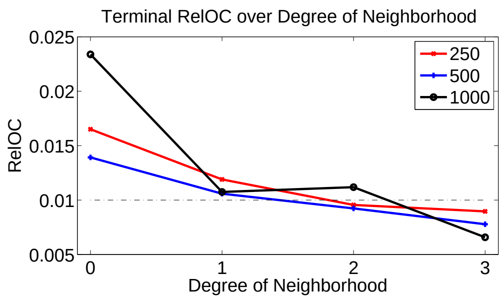
(c) d=2



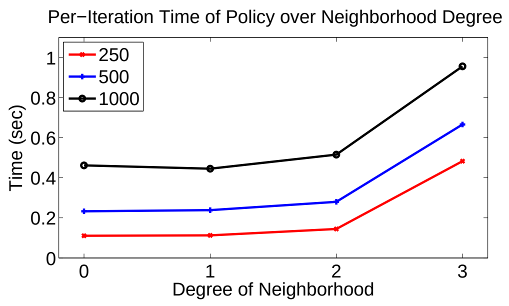
(d) d=3

Figure 3.3: Impact of surrogate iterations and degrees of neighborhood

Figure 3.3 shows the RelOCs of our algorithm over different d value and surrogate iterations. Scenarios with non-zero d value all performed significantly better than those with $d = 0$, demonstrating the value of our approximated knowledge-gradient policy. The performance of the algorithm also generally improves as d increases from 1 to 3. This is expected as a larger degree of neighborhood not only includes more candidate solutions into the knowledge gradient comparison, but also increases the accuracy of each approximated knowledge gradient (i.e. $\hat{v}_f^{KG}(\cdot)$) itself. On the other hand, the number of iterations of the surrogate optimizer demonstrates a subtle trade-off between exploration and exploitation. While high surrogate iteration somewhat improves the quality of the approximated knowledge gradient, they are also more likely to focus on a constrained areas of the solution space. In particular, selecting the best surrogate solution at each iteration consistently produces the worst performance across all degrees of neighborhood. This observation corroborates with earlier studies in response surface modeling [39] in which "greedily" selecting the optimal surrogate solution at each iteration actually decreases the performance of the algorithm. On the other hand, low-surrogate iterations can explore more areas of the solution space, which could sometimes over-compensate the relatively poor quality of the surrogate solutions. As presented in Figure 3.3 (a), algorithms with low (i.e. 250 or 500) surrogate iterations often slightly out-perform those with high surrogate iterations.



(a) Average RelOC , Iteration 141 $\sim$ 150



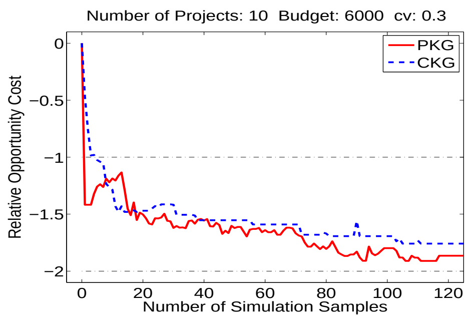
(b) Per-Iteration Computational Time

Figure 3.4: Time-RelOC trade-off of surrogate iterations and d

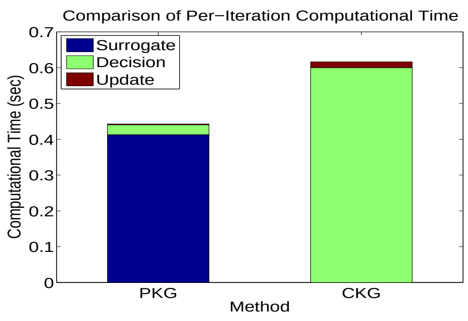
Figure 3.4 further depicts the trade-off between accuracy and computational time of different parameter configurations. Most importantly, the computational time of the surrogate solver is proportional to the number of surrogate iterations, but the approximated KG iteration has computational complexity of $O(d\log(k)k^d)$, which grows more than exponentially in d. Indeed, as Figure 3.4 (b) illustrates, the computational time of the our algorithm is dominated by the surrogate solver for small d until it dramatically increases at d=3 due to the soaring cost of the approximated knowledge gradient calculation. Overall, scenarios with d value of 2 enjoyed great computational efficiency without significant computational cost. Moderate surrogate iterations (e.g. 500) also seems to produce more stable performance across different d levels.

3.3.4 Comparison with the Bayesian Ranking & Selection model with Correlated Belief

In our last experiment we compare the performance of our new parametric Bayesian R&S model (labeled PKG) with the Bayesian $R\&S$ model in [90] with exact correlated beliefs and constrained KGCB sampling policy (labeled CKG). We choose the scenario of the Sioux Fall Network, with 10 projects, 6000 budget and cv=0.3. Figure 3.5 compares the convergence speed and per-iteration computational time of both algorithms. For RelOC, the PKG algorithm converges slightly faster than the CKG algorithm but was less stable due to the inaccurate parametric belief. For computational time, the combined time of the surrogate optimization and the localized approximated knowledge gradient calculation is already lower than directly applying the KGCB sampling policy. Moreover, the update of parametric beliefs through Equations 3.16 and 3.17 is also much faster than directly updating the posterior mean in CKG. Although the difference in computational time is relatively small in Figure 3.5 (b) , we shall bear in mind that the complexity of the PKG algorithm is *polynomial* in the number of projects while that of the CKG algorithm is *exponential*. In fact, when we have 15 candidate projects, the total time for one iteration of the CKG algorithm would exceed *5 minutes*, while the computational cost of the PKG algorithm still remains to be 0.5 seconds.



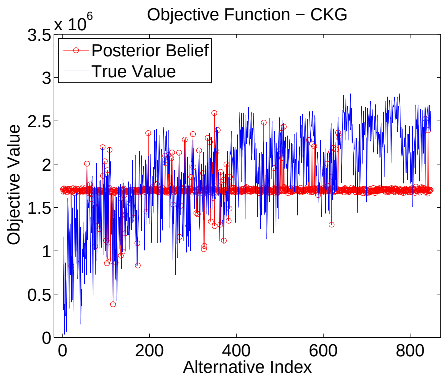
(a) RelOC



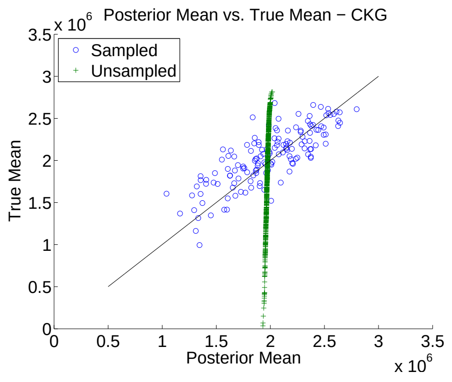
(b) Per-Iteration Computational Time

Figure 3.5: Performance Comparison between CKG and PKG

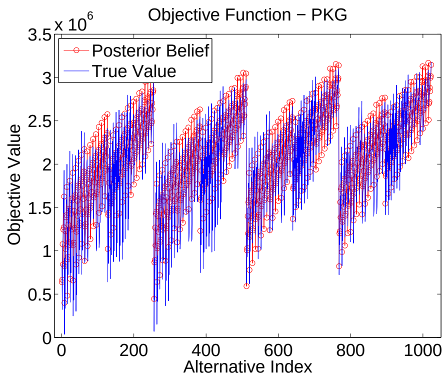
Figure 3.6 shows a more interesting comparison on the final posterior mean of the CKG and PKG algorithm. For both algorithms we select the replication with median terminal RelOC. The objective function of the CKG algorithm looks different due to pre-elimination and rearrangement of the alternatives (see [90]). While the CKG algorithm did utilize information of all alternatives in its sampling decisions, its posterior update is relatively unbalanced: sampled alternatives are updated to around their true values while un-sampled alternatives are estimated to a flat aggregated value. This is especially true for problems with non-informative priors, where the correlation between alternatives needs to be gradually learned. Nevertheless, the posterior mean of all sampled alternatives are symmetrically distributed around the diagonal reference line, showing the asymptotic unbiasedness of the posterior belief.



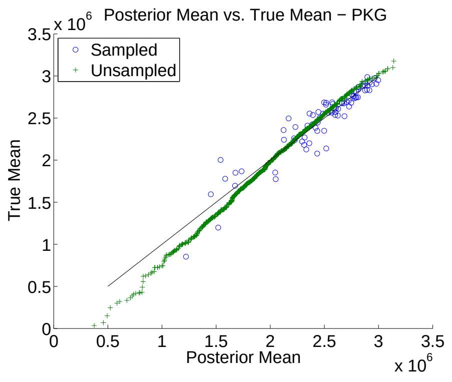
(a) CKG: True and Posterior Means



(b) CKG: True vs. Posterior Means



(c) PKG: True and Posterior Means



(d) PKG: True vs. Posterior Means

Figure 3.6: Comparison of Posterior Means: CKG vs. PKG

In sharp contrast, the posterior means of unsampled alternatives in the PKG model is very close to their true mean values albeit a mild deviation from the theoretical reference line. This is because the parametric belief by default introduces linear dependence among alternatives through their shared β parameters, which updates un-sampled alternatives as accurately as sampled ones. As Figure 3.6 (c) illustrates, the PKG algorithm quickly captured the global shape of the objective function in limited iterations. Computationally, a more informative belief can further improve the effectiveness of surrogate-assisted knowledge-gradient-related policies, which rely on posterior and predictive distributions of the objective function. In real-world applications, a globally well-approximated objective function is also more informative for the decision makers than an isolated optimal solution.

In summary, our new PKG model trades off a little unbiasedness and performance stability for significant improvement in scalability and a more global approximation of the objective function. For problems like NDPU whose combinatorial structure to some extent dictates the shape of the objective function, setting up a parametric model is straightforward and can be very beneficial. Nevertheless, the CKG model assumes no prior knowledge about the problem and can usually produce more stable convergence results.

3.4 Conclusion

We proposed a new Bayesian Ranking and Selection (R&S) model for the Network Design Problem with Uncertainty (NDPU). The model adapts an approximated parametric representation of the belief of the objective function, which is exponentially faster to update than the traditional exact "lookup-table" belief structure. It also features a surrogate-assisted knowledge gradient sampling policy which uses solutions of an auxiliary surrogate optimization problem and a localized approximated knowledge gradient to balance between the effectiveness and computational cost of the expensive KGCB calculation. The exploration vs. exploitation trade-off of our sampling policy can be customized with the degree of neighborhood in the approximated knowledge gradient calculation and the number of iterations used to solve the surrogate problem.

The parametric Bayesian R&S model inherits most of the advantages of the traditional Bayesian R&S models with correlated beliefs, namely: 1) It keeps track of the means and correlations of *all* candidate solutions (via parameters) and is thus more likely to identify globally optimal solutions; 2) It enables the use of sequential sampling policies like KGCB, in which later simulation decisions can directly benefit from information gathered in earlier samples; 3) It models uncertainty with continuous probability distributions, which are more efficient and flexible than discrete scenario sets. In addition, the new model's parametric belief and approximated sampling policy reduced the per-iteration computational complexities to a low degree polynomial in the number of projects. This greatly improves the scalability of the Bayesian R&S formulation for large numbers of projects and alternatives. Indeed, our computational experiments on the studied networks suggest that the parametric Bayesian R&S model converged more than 10 times faster than both the Genetic Algorithm and the Simulated Annealing Algorithm in all scenarios. It even outperformed the old Bayesian $R\&S$ model with exact correlated beliefs by a slight margin, although the latter has a smoother convergence path. Compared with the old model, our approximations delivers significant computational efficiency at the expense of very little bias and performance fluctuation.

Our new method demands a parametric approximation of the objective function. An ideal parametric approximation should have 1) a flexible structure with the potential to quickly capture the shape of the true objective function, 2) an efficient procedure to update parameter values and 3) little overhead to initialize the prior parameter values. For our NDPU problem with k candidate projects, we are able to use only $O(k)$ basis functions and $O(k)$ additional samples to construct a fairly accurate linear value function approximation. While the design of parametric approximations adds a prerequisite to the model, it is shown in our study that parametric approximations can facilitate better global characterization of the objective function, which may provide more information to the final decision makers.

In conclusion, we believe our parametric Bayesian R&S model provide valuable insights to the formulation and computation of NDP/NDPU and other stochastic discrete optimization problems. The surrogate-assisted knowledge gradient policy also greatly extends the potential applicability of Bayesian R&S models in large-scale optimization problems. For future work, we are currently seeking to adapt a more rigorous treatment of the unknown alternative variances (e.g. [61]) to our model and also expand our methodology to multi-objective NDPU problems.

CHAPTER 4
BAYESIAN RANKING AND SELECTION MODEL FOR MULTI-OBJECTIVE DISCRETE NETWORK DESIGN PROBLEM WITH UNCERTAINTIES

4.1 Introduction

The Network Design Problem with Uncertainty (NDPU) is a classical problem in transportation sciences and engineering. Given a network and its users (demand), NDPU optimizes network-wide objective(s) (e.g. total travel time, system capacity, environmental impact, etc.) through a set of candidate "projects" (modifications to the network), subject to randomness in traffic demand, travel cost and etc. The effectiveness of NDPU used to be measured primarily by congestion-related metrics such as total travel time, demand/capacity ratio, maximum capacity and etc. In recent years, there has been increased emphasis on the externality (e.g. environmental impact, energy consumption) of transportation systems and thus attempts to design sustainable transportation systems [26, 2, 6, 95, 35]. As a result, environmental/energy objectives are now often considered *in parallel* with congestion management goals, making NDPU inherently multi-objective. As objectives in multi-objective network design problems (MONDP/MONDPU) usually do not align with each other (e.g. system capacity vs. financial budget), the optimal solution of MONDP/MONDfPU is typically not a single point but a set of solutions with non-dominated objective values know as Pareto Optimal front. Due to the non-uniqueness and non-dominance of Pareto Optimal solutions, it is generally desirable for decision makers to obtain at least a diverse set of optimal solutions. Therefore, population-based evolutionary algorithms especially NSGA-II [22] are very popular choices for solving MONDP/MONDPU [74, 77, 12, 50]. Meanwhile, MONDP/MONDPU were also studied in the context of water resources systems [25, 1], petro-chemical sensor networks [11], green supply chain [89], and sales networks [27], where alternative solution algorithms such as ParEGO [45], parallel variable neighborhood search [27] and GOMORS [1], etc. were proposed. However, most aforementioned methods were formulated under a deterministic setting and often most suitable for problems with continuous decision variables. Moreover, evolutionary algorithms typically require simulating the objective values of hundreds to thousands of candidate solutions, which can be computationally intensive for MONDPU problems on large networks. Therefore, we are motivated to design a modeling framework for discrete MONDPU with minimal objective value evaluations and efficient management of uncertainty scenarios.

In this paper, we propose a novice Multi-Objective Bayesian Ranking and Selection (R&S) formulation to the MONDP/MONDPU problem. Ranking and Selection procedures [42] compare the stochastic performances of finite number of alternatives through limited sampling budget. Under the Bayesian R&S setting, we view each candidate solution to the MONDPU as an alternative. The random performances (i.e. "reward") of *every* alternative is learned by taking "samples" from selected alternatives and updating the beliefs we place on each alternative and each objective function through Bayesian inferences. The updated Bayesian belief is then utilized by the sampling policy to determine the most "informative" alternative to sample in the next iteration. Bayesian R&S model with scalar rewards have been extensively studied [29, 71, 68, 52, 67] in various belief structures and both discrete and continuous decision variables. In our recent studies [90, 91], it was also successfully adapted to model single-objective NDP/NDPU and have demonstrated superior performances in both sampling efficiency and uncertainty management. On the other hand, R&S/ Bayesian R&S studies on multi-objective problems have so far mainly focused on adaptive scalarization [31] or independent rewards [13, 81], either of which fit very well with the structure or motivation of MONDPU. To solve MONDP/MONDPU in Bayesian R&S formulation, we extend the model to model multi-dimensional rewards and also generalize the Knowledge Gradient(KGCB) [29] sampling policies to multi-objective problems. We believe our new formulation can be a valuable addition to existing Bayesian R&S models and can in general provide great insights to the construction and solution of MONDP/MONDPU problems or even other multi-objective stochastic optimization problems.

4.2 Methodology

4.2.1 The MONDPU Problem

Assume we have a generic network (graph) $G = (V, E)$, with V and E being the corresponding vertex set and edge set. MONDP/MONDPU optimizes multiple objectives over the network by modifying the edge set E with k improvement "projects". In this paper we focus on discrete NDP, where each project adds a few new links to the network. Let the vector **a** $= (a_1, a_2, ...a_k)$ be an overall decision (solution) with each element a_i representing the binary decision on whether to implement project i, and let **c** $= (c_1, c_2, ...c_k)$ be the corresponding project costs. Uncertainty in MONDPU is usually characterized by a countable scenario space Ω, with the probability of each scenario $\{p(\omega) : \omega \in \Omega\}$ well defined. The objective of a MONDPU problem is to identify the decision(s) $\mathbf{a}^*$ with Pareto optimal expected objective values.

NDPU/MONDPU can be viewed as a Stackelberg game [88] in which the "leader" (transportation agencies) initiates modifications to the network and the "follower" (network users) behave responsively to those changes. Consequently, they are typically formulated as Bi-level programs [23], which are NP-hard even in their simplest forms [87]. In this paper we focus on a stylized MONDPU formulation to optimize three objectives: 1) reduction in total travel time 2) reduction in PM10 emissions and 3) total financial budget of the projects. We also assume that uncertainty only exists in travel demands. The corresponding formulation is then given as below:

$$\text{(Upper Level)} \quad \max_{\mathbf{a} \in A} \quad \mathbb{E}(T_{\mathbf{a}}) = T_0 - \sum_{\omega \in \Omega} p(\omega) \sum_{(i,j) \in E \cup E'_{\mathbf{a}}} x^*_{ij}(\omega) t_{ij}(x^*_{ij}(\omega)) \tag{4.1}$$

$$\max_{\mathbf{a} \in A} \quad -1 \cdot \mathbb{E}(B_{\mathbf{a}}) = -1 \cdot \mathbf{c}^T \cdot \mathbf{a} = -1 \cdot \sum_{i=1}^{k} c_i \cdot a_i \tag{4.2}$$

$$\max_{\mathbf{a} \in A} \quad \mathbb{E}(P_{\mathbf{a}}) = P_0 - \sum_{\omega \in \Omega} p(\omega) \sum_{(i,j) \in E \cup E'_{\mathbf{a}}} er_{ij} x^*_{ij}(\omega)\, t_{ij}(x^*_{ij}(\omega)) \tag{4.3}$$

$$\text{s.t.} \quad a_i \in \{0, 1\}, \forall i \in \{1, 2...k\} \tag{4.4}$$

$$\{x_{ij}(\omega)^* : (i,j) \in E\} \tag{4.5}$$

is the optimal solution for the lower level problem: (4.6)

$$\text{(Lower Level)} \ \underset{x}{\text{Min}} \quad \sum_{(i,j)\in E\cup E'_{\mathbf{a}}} \int_0^{x_{ij}} t_{ij}(u)du \tag{4.7}$$

$$\text{s.t.:} \quad \sum_{p\in P_{rs}} f_p^{rs} = d(\omega)_{rs}, \forall (r,s) \in D \tag{4.8}$$

$$x_{ij} = \sum_{(r,s)\in D} \sum_{p\in P_{rs}} f_p^{rs} \cdot \delta_{ij,p}^{rs} \tag{4.9}$$

$$t_{ij} = t_{ij}^0 \cdot (1 + \alpha \cdot (\frac{x_{ij}}{C_{ij}})^\beta) \tag{4.10}$$

$$f_p^{rs} \geq 0, \forall p \in P_{rs}, \forall (r,s) \in D \tag{4.11}$$

where

A : the set of all network configurations, i.e. $\{0,1\}^k$

T_0 : expected total travel time on the base network

P_0 : expected total emission inventory on the base network

$E'_{\mathbf{a}}$: the set of additional links selected by decision **a**

x_{ij} : the equilibrium traffic flow on link (i,j)

er_{ij} : emission rate (grams/ mile) of pollutants on link (i,j)

$t_{ij}(x)$: travel time on link (i,j) when flow is x

t_{ij}^0 : free-flow travel time on link (i,j)

C_{ij} : capacity of link (i,j)

α, β : parameters for calculating travel time

D : the set of traffic demands indexed by Origin(r) and Destination(s).

P_{rs} : the set of paths (contiguous links) which starts in node r and ends in node s

f_p^{rs} : the flow between origin r and destination s on path p

$\delta_{ij,p}$: equals 1 if link (i,j) belongs to path p and 0 otherwise

Equation 4.1-4.6 define the Upper level problem. Equation 4.1-4.3 are the objective functions in congestion management, financial budget and environmental impact respectively. Equation 4.7-4.11 specify the lower level problem as deterministic User Equilibrium [75], a very common lower-level formulation for NDPU/MONDPU.

4.2.2 MONDPU as a Bayesian R&S Problem

The Bayesian R&S formulation of the MONDPU problem treats the lower level problem as part of an expensive evaluation of the upper-level objective functions. Our goal is to identify a set of alternatives with Pareto optimal rewards through N sample measurements. In this setting, each solution to the MONDPU becomes an alternative **a** in the alternative set A. More specifically, we can think of **a** as a k-digit binary variable, the i_{th} digit of which indicates whether project i will be implemented. The decimal value of this binary variable can also be viewed as an index for the alternative set A and we will assume elements in A is ordered by this index. The upper-level objective values of each alternative, i.e. $R_{\mathbf{a}} := (T_{\mathbf{a}},\ B_{\mathbf{a}},\ P_{\mathbf{a}})$ is modeled as a multi-dimensional "reward" we receive from sampling that alternative. The Bayesian R&S model starts by placing a probabilistic prior belief on $\boldsymbol{\Theta} := (\mathbb{E}(R_1))...\mathbb{E}(R_{|A|}))$, the expected vector reward of all alternatives. As we sample alternatives $\mathbf{a}_1, \mathbf{a}_2...\mathbf{a}_N$, we obtain measurements of the random "rewards" $R_{\mathbf{a}_1}, R_{\mathbf{a}_2}, ...R_{\mathbf{a}_N}$ and can update our knowledge of all alternatives recursively through Bayesian inferences. After we exhausted all N samples, we select the set of alternatives with Pareto optimal posterior mean rewards.

Prior and Likelihood For MONDP/MONDPU, we assume our prior belief on the expected value of all alternatives rewards is independent across objectives. In particular, each of them follows a multi-variate normal distribution and that the sequence of measurements follow independent normal distributions conditioned on the parameter values, i.e.:

$$\mathbf{\Theta} := \begin{pmatrix} \mathbb{E}(T_1) & \cdots & \mathbb{E}(T_{|A|}) \\ \mathbb{E}(B_1) & \cdots & \mathbb{E}(B_{|A|}) \\ \mathbb{E}(P_1) & \cdots & \mathbb{E}(P_{|A|}) \end{pmatrix}^T \sim \begin{pmatrix} \mathcal{N}(\mu^{T0}, \Sigma^{T0}) \\ \mathcal{N}(\mu^{B0}, \Sigma^{B0}) \\ \mathcal{N}(\mu^{P0}, \Sigma^{P0}) \end{pmatrix} \tag{4.12}$$

$$R_{\mathbf{a}_i}|\mathbf{\Theta} \sim \mathcal{N}(\mathbf{\Theta}_{\mathbf{a}_i}, \Lambda_{\mathbf{a}_i}), \forall i \in 1...N \tag{4.13}$$

where $\Lambda_{\mathbf{a}_i} := \left(\begin{smallmatrix} \lambda^T_{\mathbf{a}_i} & 0 & 0 \\ 0 & \lambda^B_{\mathbf{a}_i} & 0 \\ 0 & 0 & \lambda^p_{\mathbf{a}_i} \end{smallmatrix}\right)$ is the variance-covariance matrix for the likelihood of the three objectives. For most stochastic problems, $\{\Lambda_{\mathbf{a}_i}\}_{i=1}^N$ is unknown and needs to be approximated/estimated, which we discuss in a subsequent section.

Parametric Prior To improve computational efficiency for large numbers of alternatives, we further assume an approximated linear dependency structure on the beliefs of each objective. The resulting model is generally known as Bayesian R&S model with parametric beliefs [60] and is analogous to other optimization methods (e.g. [39, 1, 40]) in which a response surface is fitted to approximate the expensive objective function(s). More specifically, we can assume that:

$$\theta^m_{\mathbf{a}} = \Phi^m_{\mathbf{a}} \alpha^m = \sum_{i=1}^{p} \phi^m_i(\mathbf{a}) \cdot \alpha^m_i \quad \forall \mathbf{a} \in A, m \in \{T, B, P\} \tag{4.14}$$

$$\text{and} \quad \alpha^m = (\alpha^m_1 ... \alpha^m_p) \sim \mathcal{N}(\beta^{m0}, C^{m0}) \quad \forall m \in \{T, B, P\} \tag{4.15}$$

where $\theta^m_{\mathbf{a}}$ is the expected reward of alternative **a** at objective m, $\phi^m_1(\cdot)...\phi^m_p(\cdot)$ is a series of deterministic basis functions and α^m is the vector of parameters we place our normal beliefs on. According to [91], for MONDPU we can set

$$\mathbb{E}(T_{\mathbf{a}}) = \beta_0^{T0} + \sum_{i=1}^{k} \beta_i^{T0} \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b^T}} \cdot \mathbb{1}_{\{a_i=1\}}, \quad \forall \mathbf{a} \in A \tag{4.16}$$

$$\mathbb{E}(P_{\mathbf{a}}) = \beta_0^{P0} + \sum_{i=1}^{k} \beta_i^{P0} \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b^P}} \cdot \mathbb{1}_{\{a_i=1\}}, \quad \forall \mathbf{a} \in A \tag{4.17}$$

$$\mathbb{E}(B_{\mathbf{a}}) = \beta_0^{B0} + \sum_{i=1}^{k} \beta_i^{B0} \cdot \mathbb{1}_{\{a_i=1\}}, \quad \forall \mathbf{a} \in A \tag{4.18}$$

Here the corresponding $\{\beta_1^m, ...\beta_k^m\}$ for each objective m can be interpreted as the "main effect" of including each project in an alternative, and β_0^m is a nuisance "intercept" term we incorporated to improve the flexibility and account for the inaccuracy of the parametric approximation. b^m is a "bandwidth" parameter. By the property of multivariate normal distributions, the prior beliefs induced on $\mathbf{\Theta}$ via Equation 4.14 and 4.15 still follows independent multivariate normal distributions for each objective. As discussed in [91], the initial values of each sequence of $\{\beta_1, ...\beta_k\}$ can be estimated by setting the corresponding β_0^0 to 0 and generating sample estimates of the expected objective values of each individual projects $\{\hat{\theta}_{2^i}^m\}_{i=0}^{k-1}$ as our initial values for $\{\beta_0^0...\beta_k^0\}$. The bandwidth b can also be estimated by simulating an additional $\hat{\theta}_{2^k-1}^m$ (i.e. the alternative with all projects implemented) and setting

$$\hat{b}^m = \frac{1}{1-k} \cdot \ln\left(\frac{\hat{\theta}_{2^k-1}^m}{\sum_{i=1}^k \hat{\theta}_{2^{i-1}}^m}\right) \quad \forall m \in \{T, B, P\} \tag{4.19}$$

For objectives where a strict linear structure is apparent (e.g. Equation 4.18) , we can also set b to ∞ directly.

The parametric prior structure implicitly introduces linear dependence among alternatives. According to [91], such dependence structure could facilitate better estimation the global shape of each objective function. For

MONDPU, this feature can be very useful for identifying Pareto optimal solutions, in which each objective function can take a range of values in the Pareto Optimal set.

Posterior Update The diagonal structure of $\Lambda_{\mathbf{a}_i}$ in Equation 4.13 and the independent assumption in Equation 4.12 ensure that we can update the beliefs of each objective function separately. As discussed in [60], the posterior distributions of α will still be normal under the likelihood function of Equation 4.13. Define $\mathbf{a}_n$ as the alternative sampled at iteration n and $\Phi_{\mathbf{a}_n} = (\phi_0(\mathbf{a}_n), ...\phi_k(\mathbf{a}_n))$ as the vector of basis functions for each objective of alternative $\mathbf{a}_n$. The corresponding β^n and C^n for each objective can then be recursively updated as:

$$\beta^{m,n} = \beta^{m,n-1} + \frac{\hat{R}^m_{\mathbf{a}_n} - \Phi^m_{\mathbf{a}_n}\beta^{m,n-1}}{\lambda^m_{\mathbf{a}_n} + \Phi^m_{\mathbf{a}_n} C^{m,n-1}(\Phi^m_{\mathbf{a}_n})^T} C^{m,n-1}(\Phi^m_{\mathbf{a}_n})^T \quad \forall m \in \{T, B, P\} \tag{4.20}$$

$$C^{m,n} = C^{m,n-1} - \frac{C^{m,n-1}(\Phi^m_{\mathbf{a}_n})^T \Phi^m_{\mathbf{a}_n} C^{m,n-1}}{\lambda^m_{\mathbf{a}_n} + \Phi^m_{\mathbf{a}_n} C^{m,n-1}(\Phi^m_{\mathbf{a}_n})^T} \quad \forall m \in \{T, B, P\} \tag{4.21}$$

Our belief of each objective function is thus compactly stored in the sequence $\{\beta^{m,i}, C^{m,i}\}_{i=0...N}$. By the linearity of multi-variate normal distribution, our belief of each objective function is implicitly updated to $N(\Phi^m\beta^{m,i}, \Phi^m C^{m,i}(\Phi^m)^T)$.

Prior Estimation and Update of Alternative Variances We can recycle the samples used to estimate $\{\hat{\theta}^m_{2^i}\}_{i=0}^{k-1}$ to estimate their respective variances $\{\lambda^m_{2^i}\}_{i=0}^{k-1}$. Similar to our belief on Θ, for each objective we can put a parametric belief structure on the "prior" of the variances of any alternative by setting

$$\hat{\lambda}^{m,0}_{\mathbf{a}} = \sum_{i=1}^{k} \hat{\lambda}^m_{2^i-1} \cdot e^{-\frac{\|\mathbf{a}\|^2-1}{b_\lambda}} \cdot \mathbb{1}_{\{a_i=1\}} \quad \forall \mathbf{a} \in A, m \in \{T, B, P\} \tag{4.22}$$

The corresponding bandwidth b_λ can be estimated in a similar fashion to Equation 4.19. As the algorithm proceeds, we can collect more samples for a particular alternative **a** and update our "posterior" estimate of $\lambda_{\mathbf{a}}$ as in [90]. That is, for an iteration $n \in 1...N$ when we sampled alternative **a**, we set:

$$\hat{\lambda}_{\mathbf{a}}^{m,n} = \frac{v_0 \hat{\lambda}_{\mathbf{a}}^0 + \sum_{i=1}^{N_{\mathbf{a}}} (y_{\mathbf{a}}^{n(i)} - \mu_{\mathbf{a}}^{n(i)})^2}{v_0 + N_{\mathbf{a}} - 2} \quad \forall \mathbf{a} \in \{\mathbf{a}_1 ... \mathbf{a}_N\}, m \in \{T, B, P\} \tag{4.23}$$

where $N_{\mathbf{a}}$ is the number of samples collected for alternative **a** at iteration n, $n(i)$ is the iteration at which the i_{th} sample of alternative **a** is collected, $\mu_{\mathbf{a}}^{n(i)}$ is the posterior mean for alternative **a** at iteration $n(i)$ and v_0 is a non-negative weight parameter. Equation 4.23 can be viewed as a weighted average of the prior estimate in Equation 4.22 and the subsequent sample estimate. As the number of samples accumulates, it is easy to see that the variance estimates will converge to the true variances.

Decision After we exhaust all N samples, we select the network configurations with Pareto optimal posterior rewards as the proposed optimal solution:

$$A_N^* \in \arg\max_{\mathbf{a} \in A^f} \mathbb{E}(R_{\mathbf{a}} | \mathcal{F}^N) = \arg\max_{\mathbf{a} \in A^f} \begin{pmatrix} \mu_{\mathbf{a}}^{TN} & \mu_{\mathbf{a}}^{BN} & \mu_{\mathbf{a}}^{PN} \end{pmatrix}^T \tag{4.24}$$

Here $A^f \subset A$ denotes the set of feasible solutions, and the $\arg\max$ operator is defined to return the set with non-dominated posterior objective values. [22] gives an algorithm to identify all the Pareto optimal solutions in a discrete solution set of size N in $O(mN^2)$ time, where m is the number of objectives. When N is large, we can run a regular multi-objective optimization package for many iterations and replications to recover most solutions in A_N^*. A_N^* will usually be different from the true Pareto optimal set $A^* \in \arg\max_{\mathbf{a}} \mathbf{\Theta}_{\mathbf{a}}$ due to both the inaccuracy from the approximated parametric belief and the imperfect knowledge from finite samples. Therefore, our practical goal is usually to identify a diverse set of A_N^* such that every solution in $\mathbf{\Theta}_{A_N^*}$ is as close to $\mathbf{\Theta}_{A^*}$ as possible.

4.2.3 Solving the multi-Objective Bayesian R&S Model

Overview of Solution Procedure

Algorithm 4.1: Solving MONDPU with Bayesian $R\&S$ formulation

```
Require: Inputs $\{\beta^{m0}, C^{m0}, \Phi^{m}\}_{m\in\{T,B,P\}}$, sampling budget N
1: for $n = 1 \to N$ do
2:     Determine $A_n$, the next set of samples to simulate.
3:     Sample each solution in $A_n$ and obtain their rewards $\{\hat{R}_{\mathbf{a}_n} : \mathbf{a}_n \in A_n\}$
4:     Update alternative variances $\{\hat{\Lambda}^n_{\mathbf{a}_n} : \mathbf{a}_n \in A_n\}$ using Equation 4.23
5:     Given $A_n$, $\{\hat{\Lambda}^n_{\mathbf{a}_n} : \mathbf{a}_n \in A_n\}$ and $\{\hat{R}_{\mathbf{a}_n} : \mathbf{a}_n \in A_n\}$, update posterior beliefs
       of each objective from $(\beta^{m,n-1}, C^{m,n-1})$ to $(\beta^{m,n}, C^{m,n})$
6: end for
7: return $\arg\max_{\mathbf{a}\in A^f} \begin{pmatrix} \Phi^T_{\mathbf{a}}\beta^{TN}_{\mathbf{a}} & \Phi^B_{\mathbf{a}}\beta^{TN}_{\mathbf{a}} & \Phi^P_{\mathbf{a}}\beta^{TN}_{\mathbf{a}} \end{pmatrix}^T$
```

The solution procedure for the multi-objective Bayesian R&S model can be summarized in Algorithm 4.1 [91]. At each iteration n, we use information from the posterior belief at iteration n-1 to make our next sampling decision, then use the last set of samples to update our posterior beliefs. The most important step in Algorithm 4.1 is arguably the sequential sampling policy. In next section we describe the surrogate-assisted multi-objective knowledge gradient sampling policy used on Line 3 of Algorithm 4.1.

The Multi-Objective Knowledge Gradient Sampling Policy

The KGCB Sampling Policy Our sampling policy is inspired by the Knowledge Gradient policy with Correlated Beliefs (KGCB) [29, 90]. Given an intermediate normal posterior belief $N(\mu^n, \Sigma^n)$, the KGCB policy selects the alternative which maximizes the "Knowledge Gradient" function. That is, we choose

$$\mathbf{a}_f^{KG,n} \in \arg\max_{\mathbf{a}\in A} v_f^{KG}(\mathbf{a}) = \arg\max_{\mathbf{a}} \mathbb{E}^n(\max_{i\in A^f} \mu_i^{n+1}|\mu^n, \Sigma^n, \mathbf{a}_n = \mathbf{a}) - \max_{i\in A^f} \mu_i^n \tag{4.25}$$

The knowledge gradient of each alternative represents the expected "one-step" improvement in the constrained posterior optimal value if it is sampled next. By calculating the corresponding conditional predictive expectation $max_i\mu_i^{n+1}$, we can forecast the knowledge gradient of all alternatives without taking any actual sample. The KGCB policy can be described in the notations of NDP/NDPU in Algorithm 4.2 ([91]). For each alternative, Algorithm 4.2 computes its knowledge gradient (i.e. $v_f^{KG}(\mathbf{a})$) and returns the alternative with the largest. $v_f^{KG}(\mathbf{a})$ value. The time complexity of Algorithm 4.2 is $O(\log(|A^f|)|A^f||A|)$. Algorithm 4.2 is executed at each iteration of the outer Algorithm 4.1.

Algorithm 4.2: The KGCB sampling policy for NDPU with parametric belief

```
Require: Inputs $\beta^n$, $\Phi$ and $C^n$.
1: for each $\mathbf{a} \in A$ do
2:     $p \leftarrow \Phi\beta^n$, $q \leftarrow \Phi C^n \cdot \Phi e_{\mathbf{a}} / \sqrt{\lambda_{\mathbf{a}} + (\Phi C^n \Phi)_{\mathbf{aa}}}$
3:     Sort the sequence of pairs $(p_i, q_i)_{i=1}^{|A^f|}$ so that the $q_i$ are in non-decreasing order and ties in q are broken so that $p_i \leq p_{i+1}$ if $q_i = q_{i+1}$.
4:     Remove all entry i in $(p_i, q_i)_{i=1}^{|A^f|}$ where $q_i = q_{i+1}$
5:     $c_0 \leftarrow -\infty$, $c_{|A^f|} \leftarrow +\infty$, $c_i \leftarrow -(p_{i+1} - p_i)/(q_{i+1} - q_i)$, $\forall i \in 1...|A^f| - 1$
6:     Remove all entry i in $(p_i, q_i, c_i)_{i=1}^{|A^f|}$ where $c_i \geq c_{i+1}$
7:     $v_f^{KG}(\mathbf{a}) \leftarrow \log(\sum_{i=1}^{|A^f|-1}(q_{i+1} - q_i)(\varphi(-|c_i|) - |c_i|\Phi(-|c_i|)))$
8:     ($\varphi(.)$ and $\Phi(.)$ is the pdf and the cdf of a standard normal variable)
9:     return $\mathbf{a}_n = \arg\max_{\mathbf{a} \in A^f} v_f^{KG}(\mathbf{a})$
10: end for
```

The Multi-Objective KGCB Sampling Policy Our multi-objective KGCB algorithm is constructed by analogously maximizing the multi-objective knowledge gradient $V^{KG}(.)$. $V^{KG}(.)$ is defined as the vector containing the knowledge gradient of each objective. At each iteration, we select

$$A_f^{KG,n} \in \arg\max_{\mathbf{a}} \; V_f^{KG,n}(\mathbf{a}) \tag{4.26}$$

$$:= \arg\max_{\mathbf{a}} \; \mathbb{E}^n(\max_{i \in A^f} \begin{pmatrix} \mu_i^{T(n+1)} \\ \mu_i^{B(n+1)} \\ \mu_i^{P(n+1)} \end{pmatrix} \mid \begin{pmatrix} (\mu^{Tn}, \Sigma^{Tn}) \\ (\mu^{Bn}, \Sigma^{Bn}) \\ (\mu^{Pn}, \Sigma^{Pn}) \end{pmatrix}, \mathbf{a}_n = \mathbf{a}) - \begin{pmatrix} \max_i \mu_{i \in A^f}^{Tn} \\ \max_i \mu_{i \in A^f}^{Bn} \\ \max_i \mu_{i \in A^f}^{Pn} \end{pmatrix} \tag{4.27}$$

Here again we define the $\arg\max$ operator to return the set of solutions with non-dominated $V_f^{KG,n}(.)$ values. Similar to the single-objective case, the multi-objective knowledge gradient of each alternative represents the expected improvement in objective values for the subsequent posterior objective functions. As the optimal value of each objective improves, the parametric beliefs will automatically updates the reward of other solutions through their assumed linear dependency structure and push the entire posterior Pareto optimal front forward. Since every solution in $A_f^{KG,n}$ has non-dominated $V_f^{KG,n}(.)$ values, we can use the Crowding Distance metric (with respect to the iteration-n posterior beliefs) from [22] to further limit the maximum number of candidate solutions we sample at each iteration when necessary. The crowding distance of a solution is defined as the normalized sum of its distances to its adjacent solutions with respect to each objective function. Solutions with larger crowding distance are more isolated from other solutions and thus have better potential to maintain the diversity of the final solution set.

Algorithm 4.3: Multi-objective knowledge gradient policy for MONDPU

```
1: **Inputs**: $\{\beta^{mn}, C^{mn}, \Phi^m\}_{m\in\{T,B,P\}}$, $s_{max}$.
2: **for each** alternative **do**
3:     Calculate $V_f^{KG,n}(\mathbf{a})$ using Algorithm 4.2
4: **end for**
5: Find $A_f^{KG,n} \in \arg\max_{\mathbf{a}\in A^f} V_f^{KG,n}(\mathbf{a})$
6: **if** $|A_f^{KG,n}| \leq s_{max}$ **return** $A_n = A_f^{KG,n}$
7: **else for each a** $\in A_f^{KG,n}$
8:     Calculate crowding distance $cd_{\mathbf{a}}^n$ with respect to $(\mu^{Tn}\ \mu^{Bn}\ \mu^{Pn})^T$
9: **return** solutions with top $s_{max}$ largest $cd_{\mathbf{a}}^n$ as $A_n$
```

With the new multi-objective knowledge gradient definition, our MOKGCB algorithm is sketched in Algorithm 4.3. In each iteration, we first generate the set $A_f^{KG,n}$, then select the top s_{max} solutions with largest crowding distances as the candidate solutions to sample in the next iteration. When the size of $A_f^{KG,n}$ is no larger than s_{max}, the entire $A_f^{KG,n}$ is returned directly. The computational complexity of Algorithm 4.3 is $O(m \log(|A|)|A|^2)$, where m is the number of objectives in the MONDPU problem.

The Surrogate-Assisted MOKGCB Sampling Policy The computational burden of Algorithm 4.3 can be very high if $|A|$ is large. In this section we propose a surrogate-assisted approximation for Algorithm 4.3 above. This approximation can be viewed as a multi-objective extension of the method we developed in [91]. Similar works for surrogate-assisted multi-objective optimization algorithms can also be found in [1, 47, 80, 54]. The general idea of our surrogate-assisted policy is to compute the expensive knowledge gradient only on a set of "promising" (i.e. surrogate optimal) solutions recommended by the surrogate optimizer. Our approximated policy starts by solving the surrogate optimization problem constructed from the approximated objective functions:

$$\max_{\mathbf{a} \in A^f} \quad \begin{pmatrix} \mu^{Tn} & \mu^{Bn} & \mu^{Pn} \end{pmatrix}^T := \begin{pmatrix} \Phi^T \beta^{Tn} & \Phi^B \beta^{Bn} & \Phi^P \beta^{Pn} \end{pmatrix}^T \tag{4.28}$$

Equations 4.28 forms a simplified MONDPU with a deterministic and computationally cheap objective function, which we can efficiently solve via many multi-objective optimization packages. Solving Equation 4.28 would give us the set of "surrogate" solutions $\hat{A}_n^*$. $\hat{A}_n^*$ is then used as the candidate set on which we apply the approximated knowledge gradient calculation. We can estimate the approximated knowledge gradient over $\hat{A}_n^*$ as

$$\hat{V}^{KG,n}(\mathbf{a}) = \mathbb{E}^n(\max_{i\in\hat{A}_n^*} \begin{pmatrix} \mu_i^{T(n+1)} \\ \mu_i^{B(n+1)} \\ \mu_i^{P(n+1)} \end{pmatrix} \mid \begin{pmatrix} (\mu^{Tn}, \Sigma^{Tn}) \\ (\mu^{Bn}, \Sigma^{Bn}) \\ (\mu^{Pn}, \Sigma^{Pn}) \end{pmatrix}, \mathbf{a}_n = \mathbf{a}) - \begin{pmatrix} \max_{i\in\hat{A}_n^*} \mu_i^{Tn} \\ \max_{i\in\hat{A}_n^*} \mu_i^{Bn} \\ \max_{i\in\hat{A}_n^*} \mu_i^{Pn} \end{pmatrix} \tag{4.29}$$

Note that the similarity between Equation 4.25 and Equation 4.29 ensures that $\hat{V}^{KG,n}(\mathbf{a})$ can be calculated like $V^{KG,n}(\mathbf{a})$ via Algorithm 4.2. The surrogate-assisted multi-objective KGCB policy is summarized in Algorithm 4.4 below.

Algorithm 4.4: Surrogate-assisted multi-objective knowledge gradient policy for MONDPU

```
Require: Inputs: $\{\beta^{mn}, C^{mn}, \Phi^m\}_{m\in\{T,B,P\}}$, $s_{max}$.
1: Solve Equation 4.28 for the surrogate optimal solution set $\hat{A}_n^*$
2: for each alternative do
3:     Calculate $\hat{V}^{KG,n}(\mathbf{a})$ using Algorithm 4.2
4: end for
5: Compute $\hat{A}_f^{KG,n} \in \arg\max_{\mathbf{a}\in\hat{A}_n^*} \hat{V}^{KG,n}(\mathbf{a})$
6: if $|\hat{A}_n^{KG}| \le s_{max}$ return $\hat{A}_f^{KG,n}$ as $A_n$
7: else for each $\mathbf{a} \in \hat{A}_f^{KG,n}$
8:     calculate crowding distance $cd_{\mathbf{a}}^n$ with respect to $(\mu^{Tn}\ \mu^{Bn}\ \mu^{Pn})^T$
9: return solutions with top $s_{max}$ largest $cd_{\mathbf{a}}^n$
```

Algorithm 4.4 reduces the computational complexity of the sampling policy to only $O(m\ln(|\hat{A}_n^*|)|\hat{A}_n^*|^2)$ at each iteration. $|\hat{A}_n^*|$ is typically similar to $|A^*|$, the size of the true Pareto optimal set, which is much smaller than $|A^f|$. $|\hat{A}_n^*|$ can be strictly limited by applying a population-based surrogate optimizer like NSGAII, where $|\hat{A}_n^*|$ is at most the pre-defined population size.

4.3 Computational Experiments

4.3.1 Testing Data and Algorithm Packages

For our computational experiments we use the famous Sioux Fall network and a larger Anaheim network as the testing bed. Data for both networks are publicly available at [3]. Summarized information about the two networks is listed in Table 4.1.

Table 4.1: Size of Networks

Network Name	**# of Nodes**	**# of Links**	**# of OD Pairs**	**Sim Time(sec)**
Sioux Fall	24	76	576	0.084
Anaheim	416	914	1444	0.243

For the uncertainty structure, we assume that $d_{rs} : (r, s) \in D$, the demand for each Origin-Destination pair in Equation 4.8, is subject to a $p\%$ perturbation. p follows a normal distribution with mean 0 and standard deviation cv (coefficient of variation), where $cv > 0$ controls the magnitude of uncertainty. p is generated prior to each simulation and applied to *all* OD pairs in the testing network. On top of the base network, we define 12 candidate projects, whose (fictitious) costs are summarized in Table 4.2. To fully test the uncertainty management capability of our model, we add the financial objective an artificial normal perturbation with mean 0 and standard deviation at 10% of the mean budget. Our PM10 emission rate er_{ij} in Equation 4.3 is adapted from the AP42 procedure [85], which calculates road-dust PM10 emission rate as:

$$er = k \cdot (sL)^{0.91} \times (W)^{1.02} \tag{4.30}$$

where k is a particle size multiplier that equals 1 for PM10, sL is the road surface silt loading(g/m^2), and W is the average weight (tons) of the vehicles traveling the road. For our experiments W is set to be 8 and sL is set to be 0.03 for freeways and 0.09 for arterial and local roads.

Table 4.2: List of Projects

ProjectID	**Expected Budget (\$)**	**ProjectID**	**Expected Budget (\$)**
1	1800	2	1500
3	1000	4	1950
5	1650	6	2100
7	1200	8	625
9	650	10	850
11	100	12	1250

For all tests, the lower level User Equilibrium problem (Equation 4.7-4.11) is solved by an open-source solver [78] with the threshold relative gap of convergence fixed at 10^{-6}. The implementation of the KGCB subroutine (i.e. Algorithm 4.2) is based on the MatlabKG library at [30]. For bench-marking, we conduct the same tests with NSGAII, which is arguably one of the most popular solution algorithms for MONDP/MONDPU and perhaps the only viable method which can be easily adapted for our discrete formulation. For the implementation of NSGAII we use the package from [21]. The crossover probability is set to 0.9, and the mutation probability is set to 0.02. The NSGAII algorithm is also used as the surrogate optimizer in Algorithm 4.4, where its default population size and generation number are both set to 50.

4.3.2 Performance Measures

Unlike in single-objective optimization, there is no single metric to measure the effectiveness of a multi-objective algorithm. Ideally, solutions of multi-objective problems should possess: 1) Close proximity to the true Pareto optimal front. 2) Good coverage over the full range of the true Pareto optimal solution set and 3) Good internal distribution/diversity. Previous studies have proposed many useful metrics such as the Hyper-volume Index [16], Inverted Generational Distance Index [86] and various diversity measures [92]. [96] and [20] have detailed reviews about the performance metrics used for multi-objective optimization. Nevertheless, many performance measures are deigned primarily for continuous decision variables and some only for two-dimensional problems, which does not ideally fit in the structure of discrete MONDPU. In the next few paragraphs we describe our modified performance metrics for the MONDPU model. To balance the impact of different objectives, we assume each objective of our final solution set A^*_N has been normalized by the true Pareto optimal set A^* to take values between 0 and 1 before the performance metrics are computed.

Point-wise Convergence To measure the average distance between A^*_N and A^*, we use the Inverted Generational Distance (IGD) Index [86]. This metric is defined as the average distance between each solution in the A^*_N and their nearest point in A^*, i.e.

$$IGD(A) := \frac{\sum_{a \in A} min_{a^* \in A^*} |\Theta_{a^*} - \Theta_a|_1/d}{|A|} \tag{4.31}$$

Here d is the dimension of the objective functions. We define the distance between two solutions as the L1-norm (i.e. sum of absolute values) of the differences in their objective values. The IGD index can take values between 0 and

1. Unlike problems with continuous Pareto front where the shortest distance is simulated with a large set of equally spaced Pareto Optimal solutions, the nearest element of A^* to each solution in A_N^* is exact for our discrete MONDPU. Therefore, the IGD index can take value 0 if A_N^* matches exactly with A^*.

Spread The spread of a solution set A with respect to A^* is defined as the average differences between the minimum or maximum objective values of the two sets in each objective dimension:

$$SPD(A) = \frac{\sum_{i=1}^{d} |max_{a^* \in A^*} \Theta_{a^*}^i - max_{a \in A} \Theta_a^i| + \sum_{i=1}^{d} |min_{a^* \in A^*} \Theta_{a^*}^i - min_{a \in A} \Theta_a^i|}{2d} \tag{4.32}$$

The spread of a solution set measures its convergence in terms of range in each objective dimension. It can take values between 0 and 1, with 0 taken if the range of A_N^* matches with A^* in every dimension.

Diversity The most common diversity measure for a multi-objective solution set is the standard deviation of the distance between adjacent solutions [22]. However, the "adjacency" of solutions is only uniquely-defined for bi-objective problems. For discrete optimization problems, elements in A^* are not expected to be equally spaced either. To adjust the diversity measure for discrete and high dimensional problems, we propose the combined diversity metric as

$$Div(A) := (1 - \eta(A, A^*)) \cdot std(CrowdingDistance(A)) \tag{4.33}$$

The second term is the standard deviation of the crowding distance of the candidate solution set, which is well-defined for any number of objectives. For the first term, we define

$$\eta(A, A^*) := \frac{|\{\arg\min_{a^* \in A^*,\ a \in A} |\Theta_{a^*} - \Theta_a|_1\}|}{|A^*|} \tag{4.34}$$

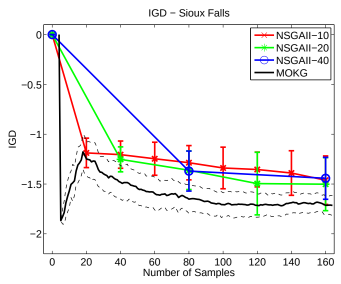
In other words, $\eta(A, A^*)$ is the percentage of Pareto Optimal solutions chosen as the nearest neighbor to solutions in the candidate set A. $\eta(A, A^*)$ measures the diversity of A in terms of the number of Pareto optimal solutions it "covers". Nevertheless, given the same $\eta(A, A^*)$ value, we still prefer solution sets that are more equally spaced, as they tend to provide decision makers with a larger variety of choices. Div(A) can also take value between 0 and 1 with 0 taken if A_N^* is identical to A^*.

4.3.3 Result Analysis

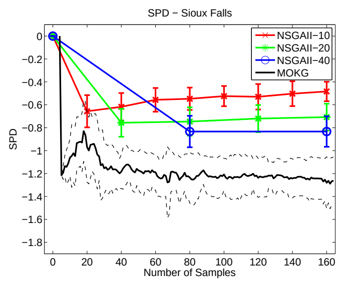
Comparison of Performance Measures

In our first set of analysis we focus on the comparison of the three quantitative performance measures between NSGAII and our multi-objective Bayesian R&S model. We run both models for 20 replications on the Sioux Falls and the Anaheim networks. The sampling budget per replication is fixed at 160. The population size of the NSGAII algorithm is set to 10, 20 and 40. To off-set the impact of uncertainty, we use the mean of two Monte-Carlo samples to evaluate the fitness (objective values) of each solution in the NSGAII population.

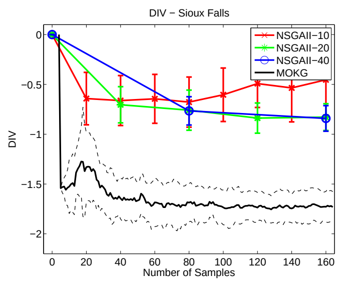
Figure 4.1 plots the three performance measures over the number of samples. Note that we used log_{10} scale on the y axis. The error bars and the dashed lines indicate standard deviation of the performance measures. It is apparent that our MO Bayesian $R\&S$ model outperformed the NSGAII algorithm in all three aspects, especially in spread and diversity. The performance advantage of our model comes from both the high value of information per sample and its more flexible sampling scheme. The NSGAII algorithm simulates a fixed



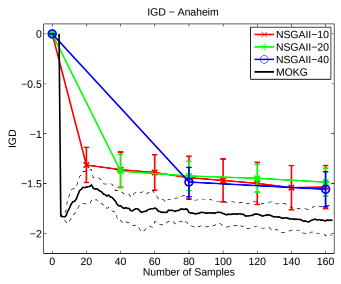
(a) IGD - Sioux Falls



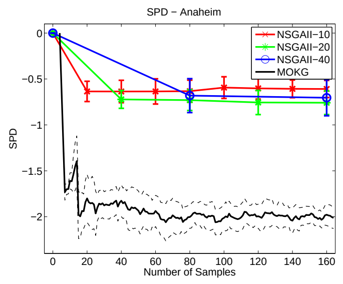
(b) SPD - Sioux Falls



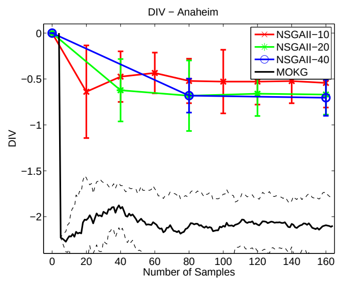
(c) DIV - Sioux Falls



(d) IGD - Anaheim



(e) SPD - Anaheim



(f) DIV - Anaheim

Figure 4.1: Comparison of Performance Measures between NSGAII and Multi-Objective Bayesian R & S

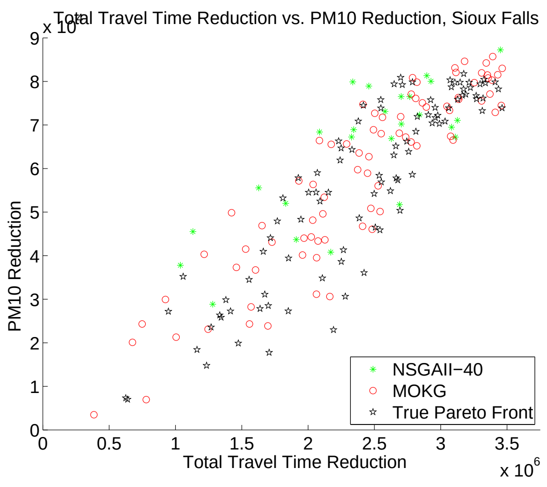
number of samples at each iteration. This approach limits the information we maintain to only alternatives in each population and may also force the algorithm to sample large number of redundant alternatives. When sampling budget is low, this population-based approach also forces an unnecessary trade-off between the diversity (population size) and accuracy (generations number) of the final solution set. On the other hand, the multi-objective Bayesian $R\&S$ model maintains information about the entire objective functions through the parametric beliefs. The final solution set is inferred from the parametric beliefs, and the diversity of solutions is automatically maintained as the accuracy of the beliefs improves. Moreover, the number of samples in each iteration of the multi-objective Bayesian $R\&S$ model is totally based on the knowledge gradient front at each iteration. As a result, alternatives are sampled only when their value of information is high, which greatly improves its sampling efficiency.

A recurring feature for all convergence paths of the multi- objective Bayesian R&S model in Figure 4.2 is the initial "spike" around the first 20-30 iterations. This behavior represents a learning phase in which the approximated parametric belief actively adjusts the parameter values to capture the dynamic of the true objective functions. This behavior is well documented in the (single-objective) approximate dynamic programming literature [60] and also observed in our previous studies [91].

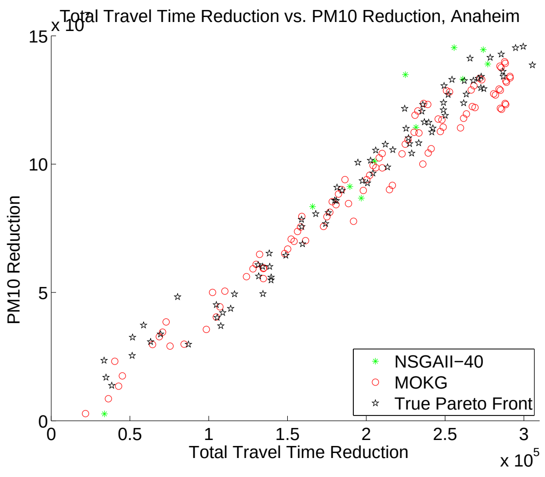
Decision Support Capability

In this section we illustrate the effectiveness of our model in practical multi-objective decision making. Unlike in single-objective NDPU where a single alternative is recommended and then simulated, it is usually impractical to simulate every solution proposed by the MONDPU solver due to their large numbers and high computational cost. Instead, it is more sensible to use the final *posterior* objective values supplied by the solution algorithm as the guideline for the subsequent decision making. For our MO Bayesian R&S model, the posterior objective values are generated by the approximated posterior objective function. For direct sampling algorithms like NSGAII, the posterior objective value of each solution is their corresponding Monte Carlo estimation.

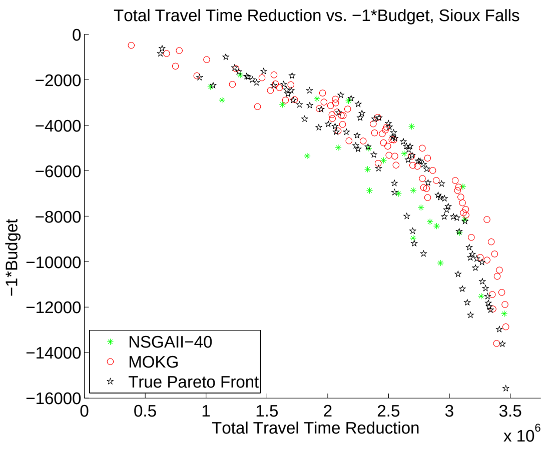
Figure 4.2 plots the three 2-dimensional trade-off curves of the two networks. The trade-off curves illustrated the typical negative correlations between financial budget and network design goals, while a more convoluted relationships is found between PM10 reduction and total travel time reduction. While the two goals are positively correlated in general, their correlation structure vary between networks. In the Anaheim network, PM10 reduction and Total Travel



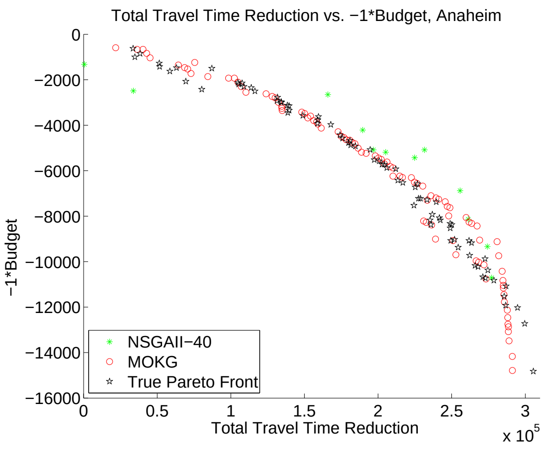
(a) Obj1 vs. Obj2 - Sioux Falls



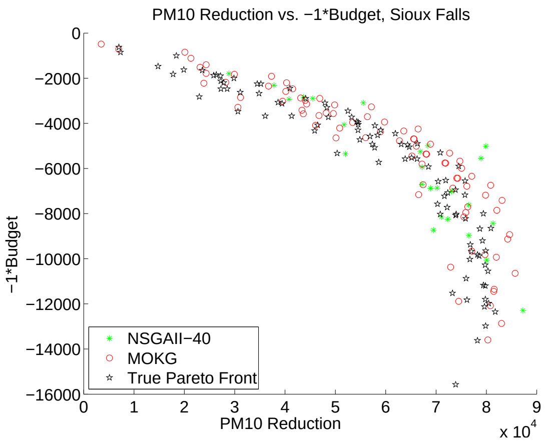
(b) Obj1 vs. Obj2 - Anaheim



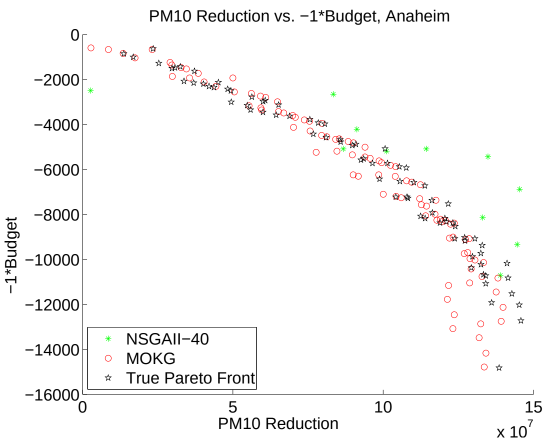
(c) Obj1 vs. Obj3- Sioux Falls



(d) Obj1 vs. Obj3- Anaheim



(e) Obj2 vs. Obj3 - Sioux Falls



(f) Obj2 vs. Obj3 - Anaheim

Figure 4.2: Visual Analysis of Median Posterior Goodness of Fit

time reductions aligned well with each other, making the final design decision relatively straightforward. However, in the Sioux Falls network, there are typically multiple Pareto optimal solutions for each budget level, not to mention the almost stagnant PM10 reduction in sub-figure e) after the budget level hits 10,000. Although our testing networks and the calculation of objective functions are highly stylized, these observed dynamics still showcased the level of complication involved in multi-objective decision making and the need to understand the interaction among objectives through MONDPU models.

The posterior means of the final solutions set from NSGAII with population size 40 (NSGAII-40) and our multi-objective Bayesian R&S model are plotted along with the trade-off curves. For both algorithms we choose the replication with median *combined* rank of the IGD, SPD and DIV values. For the two networks with 95 and 78 respective Pareto Optimal solutions, NSGAII-40 was only able to generate 26 and 12 solutions. This corroborates with the low SPD and DIV values we observed in the previous analysis. We can also observe especially in sub-figure d) and f) a few extremely deviated NSGAII solutions with much larger objective values. The deviations most likely comes from the randomness in the Monte Carlo simulation of objective values, which could greatly distort the decision makers' understanding of the problem. In short, the applicability of the NSGAII algorithm in MONDPU is largely constrained by both the limited sampling budget and the randomness in samples.

Meanwhile, it is clear from all plots that the posterior mean of our multi-objective Bayesian R&S model is able to supply a large variety of solutions. In fact, our posterior mean was able to supply 80 and 85 solutions respectively. As our MO Bayesian R&S algorithm explicitly keeps track of the variances of the objective values of each alternatives, our posterior objective values also closely mimic the dynamics of the true Pareto Optimal set. Our proposed solutions would most likely give potential decision makers a much complete picture of the problem at hand.

Sensitivity Analysis of MOKGCB Parameters

In this section we conduct a few sensitivity analysis on the tunable parameters of the multi-objective Bayesian R&S model, namely, the maximum number of samples per-iteration (s_{max}), the population size (nPop) and generation number (nGen) of the NSGAII surrogate solver, and the number of surrogate solvers (nSur). s_{max} controls the number of calls to the sampling policy and is thus directly associated with the computational cost of the algorithm. nPop, nGen and nSur are associated with the both the accuracy and the diversity of the surrogate optimal solutions. We run the multi-objective Bayesian R&S model for 10 replications for each tested parameter configuration and record the average performance measures of the last 10 samples in each replication. The performance measures are then further averaged over replications for presentation. The best value in each performance measure and each network is bolded.

Table 4.3: Impact of s_{max} (Sioux Falls/Anaheim)

s_{max}	IGD	SPD	DIV	Time (sec/sample)
1	0.0188 / 0.0158	0.0643 / 0.0280	0.0219 / 0.0108	0.263 / 0.302
3	0.0194 / 0.0149	0.0651 / 0.0299	**0.0177** / 0.0108	0.138 / 0.126
5	0.0192 / 0.0151	0.0629 / 0.0279	0.0203 / **0.0083**	0.116 / 0.084
10	0.0211 / 0.0145	0.0672 / **0.0270**	0.0194 / 0.0089	0.112 / 0.091
∞	**0.0171** / **0.0139**	**0.0587** /0.0292	0.0202 / 0.0099	**0.096** / **0.078**

Table 4.3 lists the three performance measures as we change s_{max} from 1 to ∞ (i.e. the sampling policy will sample every solution the surrogate optimizer recommends at each iteration n). As we allow more alternatives to be sampled at each iteration, the per-sample computational cost of our sampling policy is reduced. Quite interestingly, almost all the performance measures also improve as s_{max} increases albeit the fact that higher s_{max} calls the sampling policy fewer times. This indicates that our MO Bayesian R&S model favors a more myopic sampling scheme. Indeed, large numbers of earlier samples can help construct better parametric approximations of the objective functions, which will in turn increase the value of information of later samples.

Table 4.4: Impact of nGen and nPop (Sioux Falls/Anaheim)

nGen	nPop	IGD	SPD	DIV	Time (sec/sample)
10	10	0.0196 /0.0158	0.0674 / **0.0218**	0.0176 / 0.0088	0.1190 / 0.0792
20	20	0.0188 / 0.0156	0.0592 / 0.0304	**0.0163** / 0.0125	0.0841 / 0.0563
50	50	**0.0169** / **0.0131**	**0.0512** / 0.0292	0.0205 /**0.0078**	**0.0826** / **0.0515**
100	100	0.0200 / 0.0142	0.0655 / 0.0324	0.0182 / 0.0116	0.1496 / 0.0865
∞	∞	0.0192 / 0.0145	0.0786 / 0.0450	0.0305 / 0.0135	0.3433 /0.3345

Table 4.4 summarizes the analysis on nPop and nGen. The s_{max} value in this set of tests is set to 10. The ∞ parameter value is defined to enumerate all the Pareto optimal solutions from the approximated objective function without calling the surrogate optimizer. The testing results showed that our sampling policy performed relatively stably over the different surrogate configurations. The per-sample computational time of the sampling policy is dominated by frequent calls to the knowledge gradient calculation in small surrogate population/generation numbers and dominated by the extensive surrogate optimization solution process for large population/generation numbers. Similar to sin-

gle objective surrogate optimization [39, 91], ”greedily” exploiting the optimal surrogate solutions (i.e. large or infinite population/generation numbers) at each iteration slightly decreases the diversity and robustness of surrogate candidate solutions, which in turn decreases the performance of the algorithm.

Table 4.5: Impact of Parallel Surrogate Optimizer (Sioux Falls/Anaheim)

nPop	nSur	IGD	SPD	DIV	Time (sec/sample)
120	1	0.0196 / 0.0151	**0.0501** / 0.0291	0.0209 / **0.0142**	0.1242 / 0.0921
60	2	0.0165 / 0.0186	0.0780 / **0.0270**	0.0206 / 0.0104	0.0971 / 0.0779
40	3	**0.0164** / 0.0160	0.0617 / 0.0271	0.0213 / **0.0102**	0.0915 / **0.0754**
30	4	0.0188 / 0.0147	0.0744 / 0.0330	0.0208 / 0.0122	**0.0896** / 0.0770
20	6	0.0187 / 0.0142	0.0680 / 0.0296	**0.0201** / 0.0103	0.0945 / 0.0880

Table 4.5 summarizes the comparison on nSur given the same total surrogate population size of 120. The s_{max} value in this test is set to 10 and nGen to 50. The surrogate optimizers are executed in parallel using the multiple cores of our workstation. In general, multiple surrogate optimizers performed better than single ones, as they tend to provide more diverse surrogate optimal solutions. The diversity of surrogate optimal solutions will likely increase the number of solutions Algorithm 4.4 recommends, which reduces the computational time per sample. However, as the degree of parallelism increases, the population size in each surrogate optimizer become too small to maintain satisfactory accuracy, and the performance of the sampling policy is slightly decreased.

In summary, we feel that the multi-objective Bayesian R&S model has relatively robust performances over the large range of parameter values we tested. Most importantly, the effectiveness of the surrogate optimizer has very little impact on the performance of the MOKGCB policy, which illustrates the great flexibility of our surrogate-assisted MOKGCB sampling policy. For better computational performances, it may be desirable to set s_{max} to a large value and apply a small number of parallel surrogate optimizers.

4.4 Conclusion

We proposed a Bayesian Ranking and Selection (R&S) model for the Multi-Objective Network Design Problem with Uncertainty (MONDPU). In this model we place an independent approximated linear Bayesian belief on each objective function and recursively updates our parametric beliefs from samples. The alternatives to sample at each iteration are selected through a surrogate-assisted multi-objective version of the knowledge gradient sampling policy, which extends the value of information definition to multiple dimensions. To the best of our knowledge, our model is the first to generalize the Bayesian R&S model and the knowledge gradient sampling policies to discrete multi-objective optimization problems. Compared with traditional population-based multi-objective optimization algorithms, our Bayesian R&S model has an inherent uncertainty management capability from its statistical formulation, a dynamic "population size" depending on the value of information of samples, and a parametric belief structure which compactly keeps track of the global curvature of each objective function and the Pareto optimal front. In our testing examples, the multi-objective Bayesian R&S model overcame the limited evaluation budget and the uncertainty structure of MONDPU to recommend a highly Pareto optimal and diverse set of solutions. These solutions would give transportation policy makers a much more comprehensive understanding of the trade-offs between various financial, environmental and congestion-related objectives and enable the design of more sustainable transportation systems.

In conclusion, we believe our multi-objective Bayesian R&S model and the surrogate-assisted MOKGCB policy provide valuable insights to the formulation and computation of MONDPU and other discrete multi-objective optimization problems. For future work, we are currently seeking to adapt a more rigorous treatment of the unknown alternative variances (e.g. [61]) to our model and also extend our formulation to multi-period problems.

CHAPTER 5

CONCLUSION

In this dissertation we have introduced the Bayesian Ranking and Selection formulation for single objective and multi-objective discrete Network Design Problem with Uncertainty (NDPU). With this formulation, each solution to the NDPU problem represents an "alternative" with a stochastic "reward" (i.e. objective value). We start the Bayesian R&S process by placing a prior belief on the mean and correlation structure of the expected rewards of all alternatives. At each iteration, we make sequential sampling decisions based on the sampling history maintained by the recursively updated Bayesian beliefs. Constraints in the NDP/NDPU problems can be utilized to pre-eliminate infeasible alternatives. Uncertainty structures are modeled by probability (normal) distributions whose non-uniform variances are estimated and updated from samples. With the Bayesian R&S formulation, we are able to view NDPU/MONDPU from a statistical learning/simulation optimization perspective and focus our effort on directly exploring the inherent relationships among different solutions.

The advantages of this new Bayesian R&S formulation for NDPU/MONDPU are mainly: 1) It allows us to incorporate prior belief structure into the solutions procedure, which could significantly narrow down the search space and consequently improve convergence speed 2) It records and updates the means and correlation structures of *all* candidate solutions, which gives us a more global understanding of the objective functions to avoid local optima and also to identify Pareto optimal solutions for multi-objective problems 3) It enables us to use sequential sampling policies like KGCB, in which later simulation decisions can directly benefit from information collected via earlier samples and 4) It models uncertainty with continuous probability distributions, which are more efficient and flexible than discrete scenario sets. Indeed, in our first study of Bayesian R&S model with exact correlated beliefs in Chapter 2, our computational experiments have already suggested that our model performed better than both the Genetic Algorithm and the Simulated Annealing Algorithm for both NDP and NDPU. The superior performance of the Bayesian R&S model also implies its potential in solving similar discrete optimization problems.

Chapter 3 further extended the basic ideas of Chapter 2 by adding a parametric (linear) belief structure and a surrogate-assisted knowledge gradient sampling policy. The parametric approximation allows us to set up a more informative prior for both the expected rewards and the corresponding variance of all alternatives. It is exponentially faster to update than the traditional exact "lookup-table" belief structure in Chapter 2 and has the potential to closely mimic the global shape of the objective function in just a few iterations. The surrogate-assisted sampling policy uses the approximated objective function to construct a surrogate optimization problem, whose solution and its neighboring alternatives are then passed to a localized approximated knowledge gradient computation. The policy combines the efficiency of off-the-shelf optimization algorithms with the predictive power of knowledge gradient calculations, and have achieved great balance between accuracy and computational cost. Altogether, the two powerful approximations reduced the per-iteration computational complexities of our Bayesian R&S model to a low degree polynomial in the number of projects, which greatly improved the scalability of Bayesian R&S models. In fact, our computational experiments with as many as 1 million alternatives suggest that the parametric Bayesian R&S model converged more than 10 times faster than both the Genetic Algorithm and the Simulated Annealing

Algorithm in all scenarios, adding only minimal additional computational cost per iteration. Compared with our old model in Chapter 2, our approximated model in Chapter 3 delivered significant computational efficiency at the expense of very little bias and performance fluctuation.

The advantage of the parametric belief structure to quickly capture the global shape of the objective function further motivated us to extend the Bayesian R&S formulation to *posteriori* MONDPU. *Posteriori* multi-objective optimization methods aim to propose a set of alternative or trade-off solutions to the decision maker, after which he/she can make an informed decision. As solutions in the Pareto optimal front can take a range of objective values in each objective dimension, the objective function approximations need to be uniformly accurate across the domain of the decision variables. In our multi-objective model, we place an independent parametric Bayesian belief on each objective function. The alternatives to sample at each iteration are selected via a surrogate-assisted multi-objective version of the knowledge gradient sampling policy, which extends the knowledge gradient definition to multiple dimensions. Compared with traditional population-based multi-objective optimization algorithms, our multi-objective Bayesian R&S model has an inherent uncertainty management capability from its statistical formulation, a dynamic "population size" (i.e. number of samples per iteration), and a parametric belief structure which compactly keeps track of the global curvature of the Pareto optimal front. In our testing examples, the multi-objective Bayesian R&S model overcame the limited evaluation budget (i.e. 160) and the uncertainty structure of MONDPU to recommend a highly diverse set of Pareto optimal solutions, while the bench-marking NSGAII failed to adequately illustrate the trade-offs among objectives. Our Bayesian R&S model demonstrated great potential for real-world multi-objective decision making. To the best of our knowledge, our study is the first to generalize the Bayesian R&S model and the knowledge gradient sampling policies to discrete multi-objective optimization problems. It is also one of the first studies which formally incorporated randomness in *posteriori* multi-objective optimization problems.

The main contributions of the dissertation are threefold. Firstly, we have introduced the idea of modeling NDPU/MONDPU as a simulation optimization problem through the Bayesian R&S models. Due to their explicit mathematical form, NDPU/MONDPU problems have been studied mostly from a mathematical programming perspective. This viewpoint often limits us to model the uncertainty structure of NDPU/MONDPU as discrete scenario sets, which typically leads to inefficient solution algorithms. Previous heuristic methods such as genetic algorithms, simulated annealing and NSGAII have implicitly treated NDPU/MONDPU as a single-level information collection procedure, but none has effective mechanisms to account for either the expensive evaluation cost or the randomness of the upper-level objective function(s). With the Bayesian R&S formulation, uncertainty can be characterized flexibly by probability distributions and samples can be saved via the one-step forecasts from knowledge-gradient policies. While this simulation-optimization interpretation of NDPU/MONDPU might overlook some subtle structures of bi-level problems, the computational improvement in convergence speed, computational time, and even the solution diversity for MONDPU is significant. As a result, the Bayesian R&S formulation have greatly extended the practicality of NDPU and MONDPU and related problems in real-world decision making.

Our next contribution is the introduction of surrogate-assisted knowledge gradient sampling policies to Bayesian R&S models. Surrogate-assisted sampling policies/decision rules have been actively used in earlier studies of simulation optimization/black-box optimization algorithms (e.g. [1]). However, most previous studies about Bayesian R&S models focused more on the mathematical rigor of the sampling policies instead of their potentials in real-world large-scale applications. We developed the surrogate-assisted knowledge gradient policy by chaining a surrogate optimizer with a localized knowledge gradient calculation. The surrogate optimizer can quickly rule out the majority of "unpromising" alternatives, after which the knowledge gradient calculation compares in detail the value of information of a small group of highly competent solutions. Although the theoretical grantee for surrogate assisted policies are likely loose, they have demonstrated superior performance in our case studies. Coupled with the Bayesian parametric belief, we were also able to extend the Bayesian R&S models to mutli-objective problems, which essentially united the formulation of MONDPU and NDPU under a single framework. The parametric belief + surrogate-assisted knowledge gradient policy can be viewed as a stochastic Bayesian version of response surface modeling [44], which greatly extended the applicability of Bayesian R&S models in generic black-box optimization/simulation optimization problems.

The last contribution is our inclusion of non-uniform alternative variances in Bayesian R&S models. At the time of writing, most earlier studies in Bayesian R&S models (e.g. [29, 52]) still assumed a simple known constant "measurement noise" for all alternatives. As such assumption is generally invalid for NDPU/MONDPU, we established a separate prior estimate and posterior update procedure to learn about the variability of objective function(s) from the very limited samples. Albeit the general lack of accuracy in variance estimates, the Bayesian R&S models in our studies all performed reasonably well, especially with parametric beliefs. This performance stability may be attributed to the uncertainty "buffer" provided by the covariance of the Bayesian beliefs and has also demonstrated the robustness of the Bayesian R&S formulation. The success of the light-weighted variance estimation procedure also enables Bayesian R&S models to be applied to stochastic optimization problems whose uncertainty structure cannot be efficiently described via traditional optimization models.

Nevertheless, this dissertation still leaves plenty of room for further improvements of the Bayesian R&S models and the expansion of their potential application areas. The first area of improvement is the variance estimation procedure. Our current models all require a non-trivial number of "first stage" samples, which could create overhead for large-scale problems. The knowledge gradient sampling policy we adapted from [29] also assumed known alternative variances, which could generate unpredictable biases if the variance estimates are directly plugged in as did in all our studies.Therefore, it would be very desirable for future studies to adapt a more rigorous treatment of the unknown alternative variances (e.g. [61]). Next, to generalize the Bayesian R&S model to other classes of optimization problems, we may also need to conduct more experiments with non-parametric Bayesian beliefs (e.g. [49, 4]), high-dimensional/integer decision variables and high-dimensional objectives to stress-test the effectiveness of both the approximated Bayesian beliefs and the surrogate-assisted knowledge gradient sampling policies. A third interesting direction would be to extend the Bayesian R&S formulation to multi-period optimization problems.

In conclusion, we believe our Bayesian R&S models and the surrogate-assisted knowledge gradient sampling policy provide a highly practical alternative for applying NDPU and MONDPU in real-world, large scale, single and multi-objective decision making. The new formulation is intuitive to understand and easily applicable to similar classes of optimization problems. We believe the models themselves as well as this unique statistical learning perspective is of great interest and value for network modelers and simulation optimization practitioners.

BIBLIOGRAPHY

[1] Taimoor Akhtar. *Efficient Multi-Objective Optimization of Computationally Expensive Models with Application to Watershed Model Calibration*. PhD thesis, Cornell University, 2012.

[2] Anjali Awasthi, Satyaveer S. Chauhan, and Hichem Omrani. Application of fuzzy topsis in evaluating sustainable transportation systems. *Expert Systems with Applications*, 38:12270–12280, 2011.

[3] Hillel Bar-Gera. Transportation network test problems. http://www.bgu.ac.il/∼bargera/tntp/, March 2013.

[4] Emre Barut and Warren B. Powell. Optimal learning for sequential sampling with non-parametric beliefs. *Journal of Global Optimization*, March 2013.

[5] Robert E Bechhofer and Dominique Jean-Marie Nocturne. Optimal allocation of observations when comparing several treatments with a control. ii: 2-sided comparisons. *Technometrics*, 14 (2):423–436, 1972.

[6] William R Biack. Sustainable transportation: a us perspective. *Journal of Transport Geography*, 4(3):151159, 1996.

[7] D.E. Boyce, A. Farhi, and R Weischedel. Optimal network design problem: A branch and bound algorithm. *Environment and Planning*, 5:519–533, 1973.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[9] G.E. Cantarella, G. Pavone, and A Vitetta. Heuristics for the network design problem. In *Ewg 2002 (the 13th Mini Euro Conference)*, 2002.

[10] Enrique Castillo, Inmaculada Gallego, Jos Mara Menndez, and Ana Rivas. Optimal use of plate-scanning resources for route flow estimation in traffic networks. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):380–391, 2010.

[11] Roco L. Cecchini, Ignacio Ponzonia, and Jessica A. Carballido. Multi-objective evolutionary approaches for intelligent design of sensor networks in the petrochemical industry. *Expert Systems with Applications*, 39(3):26432649, 2012.

[12] Anthony Chen, Juyoung Kim, Seungjae Lee, and Youngchan Kim. Stochastic multi-objective models for network design problem. *Expert Systems with Applications*, 37(2):16081619, 2010.

[13] E. Jack Chen and LooHay Lee. A multi-objective selection procedure of determining a pareto set. *Computers and Operations Research*, 36:18721879, 2009.

[14] Stephen E. Chick, Jurgen Branke, and Christian Schmidt. Sequential sampling to myopically maximize the expected value of information. *INFORMS Journal on Computing*, 22(1):71–80, 2009.

[15] Stephen E. Chick and Koichiro Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49(5):732–743, 2001.

[16] C.A. Coello, G.L. Lamont, and D.A. van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, Berlin, Heidelberg, 2nd edition, 2007.

[17] J.A. Rojas Cruz and A.G.C. Pereira. the elitist non-homogeneous genetic algorithm: Almost sure convergence. *Statistics and Probability Letters*, 83:2179–2185, 2013.

[18] G.D. Dantzig, R.P. Harvey, Z.F. Lansdowne, D.W. Robinson, and S.F Maier. Formulating and solving the network design problem by decomposition. *Transportation Research, Part B*, 13:5–17, 1979.

[19] M. S Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, Inc., New York, 1985.

[20] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1st edition, 2001.

[21] Kalyanmoy Deb. Kanpur genetic algorithms laboratory/source codes. http://www.iitk.ac.in/kangal/codes.shtml, July 2012.

[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolotionalry Computation*, 6(2):182–197, 2002.

[23] Stephen Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, 2002.

[24] D.Huang, T.T.Allen, W.I.Notz, and R.A.Miller. sequential kriging optimization using multiple-fidelity evaluations. *Strucural and multi-disciplinary optimization*, 32:369–382, 2006.

[25] Francesco di Pierro, Soon-Thiam Khu, Dragan Savi, and Luigi Berardib. Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms. *Environmental Modelling and Software*, 24(2):202213, 2009.

[26] Jennifer Duthie and S. Travis Waller. Incorporating environmental justice measures into equilibrium-based network design. *Transportation Research Record: Journal of the Transportation Research Board*, 2089:58–65, 2008.

[27] Majid Eskandarpour, Seyed Hessameddin Zegordi, and Ehsan Nikbakhsh. A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem. In press, 2012.

[28] Ulrich Faigle and Rainer Schrader. On the convergence of stationary distributions in simulated annealing algorithms. *Information Processing Letters*, 27:189–194, 1988.

[29] P. I. Frazier, W. B. Powell, and S. Dayanik. The knowledge gradient policy for correlated normal rewards. *INFORMS Journal of Computing*, 21(4):599–613, 2009.

[30] Peter Frazier. Kg library. http://people.orie.cornell.edu/pfrazier/src.html, July 2012.

[31] Peter I. Frazier and Aleksandr M. Kazachkov. Guessing preferences: A new approach to multi-attribute ranking and selection. In *Proceedings of the 2011 Winter Simulation Conference*, pages 4319 – 4331, 2011.

[32] T.L. Friesz, H.J. Cho, N.J. Mehta, R.L. Tobin, and G Anadalingam. A simulated annealing approach to the network design problem with variational inequality constraints. *Transportation Science*, 26:18–26, 1992.

[33] A. Gelman, B. Carlin, H. Stern, and D Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, FL, 2004.

[34] Parikshit Gopalan, Adam Klivans, Raghu Meka, Daniel Stefankovic, Santosh Vempala, and Eric Vigoda. An fptas for #knapsack and related counting problems. In *FOCS'11*, pages 817–826, 2011.

[35] D.L. Greene. Sustainable transportation. *International Encyclopedia of the Social and Behavioral Sciences*, page 1533515339, 2001.

[36] Shanti S. Gupta and Klaus J. Miescke. Bayesian look ahead one-stage sampling allocations for selection of the best population. *Journal of Statistical Planning and Inference*, 54:229–244, 1996.

[37] S.S. Gupta and K.J. Miescke. Sequential selection procedures: a decision theoretic approach. *Annals of Statistics*, 12:336–350, 1984.

[38] Jun He and Xinghuo Yu. Conditions for the convergence of evolutionary algorithms. *Journal of Systems Architecture*, 47:601–612, 2001.

[39] H.M.Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[40] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455492, 1998.

[41] Seong-Hee Kim and Barry L. Nelson. A fully sequential procedure for indifference-zone selection in simulations. *ACM Transactions on Modeling and Computer Simulation*, 11 (3):251–273, 2001.

[42] Seong-Hee Kim and Barry L. Nelson. Recent advances in ranking and selection. In *Proceedings of the 2007 Winter Simulation Conference*, pages 162–172, 2007.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[44] Jack P.C. Kleijnen. *Design and Analysis of Simulation Experiments*. Springer, Berlin, Germany, 2010.

[45] Joshua Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolotionalry Computation*, 10(1):5066, 2005.

[46] D. E Knuth. *The Art of Computer Programming*. Addison-Wesley, 2005.

[47] George Kourakos and Aristotelis Mantoglou. Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management. *Journal of Hydrology*, 479:13–23, 2013.

[48] J.B. Lasserre, P.P Varaiya, and J.Warrand. Simulated annealing, random search, multistart or sad? *Systems and Control Letters*, 8:297301, 1987.

[49] Martijn R.K. Mes, Warren B. Powell, and Peter I. Frazier. Hierarchical knowledge gradient for sequential sampling. *Journal of Machine Learning Research*, 12:2931–2974, 2011.

[50] Elnaz Miandoabchi, Farzaneh Daneshzand, W.Y. Szeto, and Reza Zanjirani Farahani. Multi-objective discrete urban road network design. *Computers and Operations Research*, 40(10):2429–2449, 2001.

[51] Juliane Muller, Christine A.Shoemaker, and Robert Piche. So-mi: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers and Operations Research*, 40:1383–1400, 2013.

[52] D. Negoescu, P. Frazier, and W. B. Powell. The knowledge gradient algorithm for sequencing experiments in drug discovery. *Informs Journal on Computing*, 23(3):346–363, 2011.

[53] New York Metropolitan Transportation Council. Nymtc best practice model. http://www.nymtc.org/project/BPM/nybpmfuture.html, July 2012.

[54] Mahdi Arian Nik, Kazem Fayazbakhsh, Damiano Pasini, and Larry Lessard. Surrogate-based multi-objective optimization of a composite laminate with curvilinear fibers. *Composite Structures*, 94:23062313, 2012.

[55] G. Patil and S Ukkusuri. Sample average approximation technique for flexible network design problem. *J. Comput. Civ. Eng.*, 25(3):254–262, 2011.

[56] Samuel Pierre, Michel ange Hyppolite, Jean-Marie Bourjolly, and Oumar Dioume. Topological design of computer communication networks using simulated annealing. *Engineering Application of Artifical Intelligence*, 8 (1):6169, 1995.

[57] H. Poorzahedy and F Abulghasemi. Application of ant system to network design problem. *Transportation*, 32:251–273, 2005.

[58] H. Poorzahedy and M.A Turnquist. Approximate algorithms for the discrete network design problem. *Transportation Research, Part B*, 16:45–56, 1982.

[59] Hossain Poorzahedy and Omid M. Rouhani. Hybrid meta-heuristic algorithms for solving network design problem. *European Journal of Operational Research*, 182:578–596, 2007.

[60] Warren. B Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition*. John Wiley and Sons, Inc., New York, 2011.

[61] Huashuai Qu, Ilya O. Ryzhov, and Michael C. Fu. Simulation selection with unknown correlation structures. Submitted to Operations Research, 2013.

[62] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[63] Carmen G. Rawls and Mark A. Turnquist. Pre-positioning of emergency supplies for disaster response. *Transportation Research Part B*, 44:521534, 2010.

[64] Rommel G. Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38:837–853, 2011.

[65] I. Ryzhov and W. Powell. The knowledge gradient algorithm for online subset selection. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 137–144, 2009.

[66] I. Ryzhov and W. Powell. A monte carlo knowledge gradient method for learning abatement potential of emissions reduction technologies. In *Proceedings of the 2009 Winter Simulation Conference*, pages 1492–1502, 2009.

[67] Ilya O. Ryzhov and Warren B. Powell. Information collection for linear programs with uncertain objective coefficients. *SIAM Journal on Optimization*, 22(4):1344–1368, 2012.

[68] I.O. Ryzhov and W.B. Powell. A monte carlo knowledge gradient method

for learning abatement potential of emissions reduction technologies. *Proceedings of the 2009 Winter Simulation Conference*, 21(4):1492–1502, 2009.

[69] Lothar M. Schmitt. theory of genetic algorithms ii: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. www.ElsevierComputerScience.com, 2004.

[70] Larry Schumaker. *Spline Functions: Basic Theory (Cambridge Mathematical Library)*. Cambridge University Press, 3rd edition, 2007.

[71] W. Scott, P. I. Frazier, and W. B. Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(4):996–1026, 2011.

[72] Alexander Shapiro and Huifu Xu. Stochastic mathematical programs with equilibrium constraints, modeling and sample average approximation. *SIAM Journal of Optimization*, 57:395–418, 2008.

[73] S. Sharma, T. Mathew, and S Ukkusuri. Approximation techniques for transportation network design problem under demand uncertainty. *J. Comput. Civ. Eng.*, 25(4):316–329, 2011.

[74] Sushant Sharma and Tom V Mathew. Multiobjective network design for emission and travel-time trade-off for a sustainable large urban transportation network. *Environment and Planning B: Planning and Design*, 38:520–538, 2011.

[75] Yosef Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Inc. Englewood Cliffs, NJ, 1985.

[76] Siddique, Abu Bakar, and Muhammad Tahir. Joint brightness control and data transmission for visible light communication systems based on white leds. In *INFORMS Journal on Computing*, pages 1026–1030, 2011.

[77] Keemin Sohn. Multi-objective optimization of a road diet network design. *Transportation Research Part A: Policy and Practice*, 45(6):499511, 2011.

[78] Matthew Steel. A user-equilibrium traffic assignment library. http://github.com/MatthewSteel/EquilibriumSolver, July 2012.

[79] Yao Sun and M.A Turnquist. Investment in transportation network capacity under uncertainty: Simulated annealing approach. *Transportation Research Record: Journal of the Transportation Research Board*, pages 67–74, 2007.

[80] Anna Syberfeldta, Amos Nga, Robert I. Johnb, and Philip Moorec. Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *European Journal of Operational Research*, 204(3):533544, 2010.

[81] Suyan Teng, Loo Hay Lee, and Ek Peng Chew. Integration of indifference-zone with multi-objective computing budget allocation. *European Journal of Operational Research*, 203:419429, 2010.

[82] B. Toktas, J. W. Yen, and Z.B Zabinsky. Addressing capacity uncertainty in resource constrained assignment problems. *Comput. Oper. Res*, pages 169–175, 2006.

[83] S. V. Ukkusuri, T. V. Mathew, and S.T Waller. Robust networks design model using multi-objective evolutionary algorithms. *Comput. Aided Civ. Infrastruct. Eng.*, 22(1):9–21, 2007.

[84] S. V. Ukkusuri and S.T Waller. Single-point approximations for traffic equilibrium problem under uncertain demand. *Transportation Research Record: Journal of the Transportation Research Board*, pages 169–175, 2006.

[85] USEPA. Ap42, fifth edition, volume i chapter 13: Miscellaneous sources. http://www.epa.gov/ttn/chief/ap42/ch13/index.htm, January 2011.

[86] David A. Van Veldhuizen. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Technical report, Evolutionary Computation, 1999.

[87] L.N. Vicente, G. Savard, and J.J. Judice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81:379–399, 1994.

[88] H. von Stackelberg. *Market Structure and Equilibrium: 1st Edition Translation into English.* Springer, 2011.

[89] Fan Wang, Xiaofan Lai, and Ning Shi. A multi-objective optimization for green supply chain network design. *Decision Support Systems*, 51(2):262269, 2011.

[90] Xun Wang and H. Oliver Gao. A bayesian ranking and selection model for the network design problem with uncertainties. Submitted, 2013.

[91] Xun Wang and H. Oliver Gao. A bayesian ranking and selection model with parametric beliefs and surrogate-assisted knowledge gradient policy for the discrete network design problem with uncertainties. Submitted, 2013.

[92] Jingyu Yan, Chongguo Li, Zhi Wang, Lei Deng, and Demin Sun. Diversity metrics in multi-objective optimization: Review and perspective. In *Proceedings of the 2007 IEEE International Conference on Integration Technology*, pages 553–557, 2007.

[93] H. Yang and M.G.H Bell. Models and algorithms for road network design: A review and some new developments. *Transport Review*, 18(3):257–278, 1998.

[94] Y Yin. Genetic algorithm-based approach for bilevel programming models. *Journal of Transportation Engineering:ASCE*, 26:115–120, 2000.

[95] Jiangping Zhou. Sustainable transportation in the us: A review of proposals, policies, and programs since 2000. *Frontiers of Architectural Research*, 1(2):150165, 2012.

[96] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionalry Computation*, 3(4):257–271, 1999.