ABSTRACT

Title of Dissertation: SENSITIVITY ANALYSIS AND STOCHASTIC OPTIMIZATIONS IN STOCHASTIC ACTIVITY NETWORKS

Peng Wan, Doctor of Philosophy, 2022

Dissertation Directed by: Professor Michael C. Fu
R.H. Smith School of Business
& Institute of Systems Research

Activity networks are a powerful tool for modeling and analysis in project management, and in many other applications, such as circuit design and parallel computing. An activity network can be represented by a directed acyclic graph with one source node and one sink node. The directed arcs between nodes in an activity network represent the precedence relationships between different activities in the project. In a stochastic activity network (SAN), the arc lengths are random variables.

This dissertation studies stochastic gradient estimators for SANs using Monte Carlo simulation, and the application of stochastic gradient estimators to network optimization problems. A new algorithm called Threshold Arc Criticality (TAC) for estimating the arc criticalities of stochastic activity networks is proposed. TAC is based on the following result: given the length of all arcs in a SAN except for the one arc of interest, that arc is on the critical path (longest path) if and only if its length is greater than a threshold. By applying Infinitesimal Perturbation Analysis (IPA)

to TAC, an unbiased estimator of the derivative of the arc criticalities with respect to parameters of arc length distributions can be derived. The stochastic derivative estimator can be used for sensitivity analysis of arc criticalities via simulation.

Using TAC, a new IPA gradient estimator of the first and second moments of project completion time (PCT) is proposed. Combining the new PCT stochastic gradient estimator with a Taylor series approximation, a functional estimation procedure for estimating the change in PCT moments caused by a large perturbation in an activity duration's distribution parameter is proposed and applied to optimization problems involving time-cost tradeoffs.

In activity networks, crashing an activity means reducing the activity's duration (deterministic or stochastic) by a given percentage with an associated cost. A crashing plan of a project aims to shorten the PCT by reducing the duration of a set of activities under a limited budget. A disruption is an event that occurs at an uncertain time. Examples of disruptions are natural disasters, electrical outages, labor strikes, etc. For an activity network, a disruption may cause delays in unfinished activities. Previous work formulates finding the optimal crashing plan of an activity network under a single disruption as a two-stage stochastic mixed integer programming problem and applies a sample average approximation technique for finding the optimal solution. In this thesis, a new stochastic gradient estimator is derived and a gradient-based simulation optimization algorithm is applied to the problem of optimizing crashing under disruption.

SENSITIVITY ANALYSIS AND STOCHASTIC OPTIMIZATIONS IN STOCHASTIC ACTIVITY NETWORKS

by

Peng Wan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Michael C. Fu, Chair/Advisor
Professor Dilip B. Madan,
Professor Steven I. Marcus,
Professor Ilya O. Ryzhov,
Professor Eric V. Slud

Dedication

I dedicate this dissertation to my parents, Ailan Wan and Yiliang Cheng.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Michael Fu for giving me an invaluable opportunity to work on challenging and interesting projects over the past three years. It is because of his invaluable guidance, generous support, and constant patience that my graduate study has been smooth and fruitful. He has always made himself available for help and advice. Even during the COVID-19 period, he always replied to my e-mail within a couple of hours. I appreciate the time and effort he has put in this work, from my choice of research topics to the detailed editing of the thesis. It has been a pleasure to work with Professor Fu and learn from such an extraordinary individual.

I would like to thank Professor Steven Marcus for leading in-depth discussions and offering invaluable guidance, without which this thesis would have been a distant dream. Thanks are due to Professor Dilip Madan, Steven Marcus, Ilya Ryzhov and Eric Slud for their acceptance to serve as my committee of dissertation defense, their valuable time and effort in examining the quality of my work.

I owe my deepest thanks to my parents - my mother Ailan Wan and father Yiliang Cheng for their unconditional support and love. Thank you to my father, for supporting the family during the hard times. Thank you to professor Gaik Ambartsoumian and professor Andrzej Korzeniowski

for their generous support during my PhD application process. Thank you to Jessica Sadler for her help as the AMSC program coordinator.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Lastly, thank you all!

Table of Contents

List of Tables

List of Figures

Abbreviations and Notations

AN	Activity Network
IPA	Infinitesimal Perturbation Analysis
LR	Likelihood Ratio
ASE	Averaged Standard Error
PCT	Project Completion Time
SAN	Stochastic Activity Network
SE	Standard Error
WD	Weak Derivative
$\mathcal{A}$	set of all arcs in the activity network
$\mathcal{N}$	set of nodes in the network
X_i	length of arc i (activity i's completion time)
x_i	realization of random variable X_i
$F_i(x)$	distribution function for arc i
$\bar{F}_i(x)$	$1 - F_i(x)$
$f_i(x)$	probability density function for arc i
$C_a(i)$	criticality index of arc i
$I\{\cdot\}$	indicator function,
$\mathcal{P}$	set of all paths in the network
$\|P\|$	length of path P
P_j	set of arcs on path j
Y	project completion time
Y_i	length of path i
$\mathcal{P}_i$	set of paths containing arc i
$\mathcal{P}_{i-}$	set of paths not containing arc i
P^*	set of arcs on the longest path corresponding to path set $\mathcal{P}$
P_i^*	set of arcs on the longest path corresponding to path set $\mathcal{P}_i$
P_{i-}^*	set of arcs on the longest path corresponding to path set $\mathcal{P}_{i-}$
$\|\| \cdot \|\|$	operator that calculates the length of the longest path for a given set of paths
$\|\| \cdot \|\|^i$	operator that calculates the length of the longest path for a given set of paths under the condition that arc i has length 0
$\mathcal{P}_{N_D}$	set of all paths from source node to sink node
$\mathcal{P}_k$	set of all paths start from node 0 and end at node k
J_i	the set of crashing options for activity i
pred(k)	$\{i \mid (i,k) \in \mathcal{A}\}$

B	total crashing budget
H	disruption time
$f_H(x)$	density function of H
$\mathcal{N}_1^H$	$\{i \in \mathcal{N} \vert t_i > H\}$
$\mathcal{N}_2^H$	$\{i \in \mathcal{N} \vert t_i \leq H\}$
$\mathcal{A}_i^H$	given the disruption occurrence, the set of activities that belong to the ith Type constraint
N_A	number of arcs in the activity network
N_D	number of nodes in the activity network
X_i	delay of duration of activity i created by the disruption
$\mathbf{X}$	vector representation of all X_i
d_i	fixed duration of activity i
b_i^j	per unit cost of the jth crashing option of activity i
e_i^j	ratio of decreasing of activity i's duration per unit jth crashing option applied
$\bar{\theta}_i^j$	upper bound of jth option of activity i
t_k	activity starting time of activity k
θ_i^j	amount of jth crashing option applied to activity i
$\mathbf{t}$	vector representation of all t_k
$\boldsymbol{\theta}$	vector representation of all θ_i^j
$\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]}$	activity starting time of node k as a function of $\boldsymbol{\theta}$ and $\boldsymbol{\epsilon}$

Chapter 1: Introduction

1.1 Activity Network Representations

A project can be viewed as a collection of activities and events. An activity is any undertaking that consumes time and resources. An event is a well-defined occurrence in time [3]. Examples of activities are staff training, goods transportation, upgrade system, etc. Examples of events are completion of staff training, delivery of ordered products, completion of system upgrade, etc.

The most important relationship between activities in a project is the precedence relationship: an activity cannot start until some activities are completed. Each activity has a starting time, duration, and completion time. A project is completed if and only if all activities in the project are completed. The project completion time (PCT), which is also called makespan [4], is of interest to project managers and is to be minimized. The duration of an activity is the time it takes to finish the activity, which can be either deterministic or stochastic.

A network is used to represent the precedence relationships of activities in a project and is called an activity network (AN). There are two modes of representations of a project using a network: activity-on-arc representation (A-on-A) and activity-on-node representation (A-on-N) [3]. Both of the representations use a directed acyclic graph with one source node and one sink node to represent an activity. In this thesis, we use activity-on-arc representation for stochastic activity networks and time-cost tradeoffs optimizations, and activity-on-node representation for

the problem of optimal crashing of an activity network with single disruption.

1.1.1 Activity-on-arc Representation

In the activity-on-arc representation, an activity is represented by an arc in the network, and an event is represented by a node in the network. The node (event) at the tail of a directed arc (arrow) represents the start of the activity and the node (event) at the head node of the arrow represents the termination of the activity. The requirement that activity i precedes activity j is represented by having the tail node of arc (activity) i coincide with the head node of arc (activity) j. Except for the source node and sink node, each node in the activity network must have at least one arc entering it and one arc leaving it. In the activity-on-arc representation, arcs (activities) and nodes (events) are indexed by nonnegative integers, and nodes are indexed in a way such that a directed arc always leads from a smaller integer to a larger one. In the activity-on-arc representation, dummy arcs (activities) are needed in some cases and are represented by dashed arcs. A dummy activity has zero duration and does not represent an actual activity. An example of an AN with activity-on-arc representation is presented in Figure 1.1.

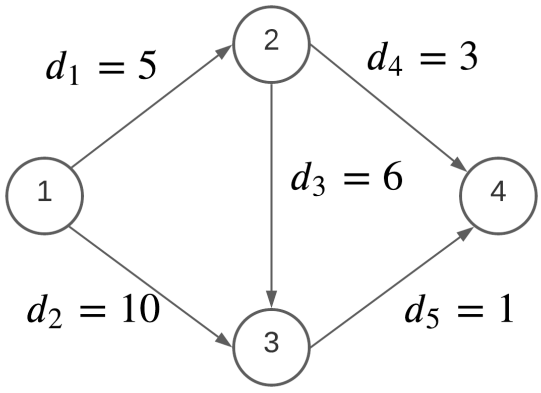


Figure 1.1: Activity-on-arc Network Example

In Figure 1.1, d_i represents the duration of activity i. Notice the difference between the index of nodes and arcs: i indexes node i, whereas the subscript of d_i is the index of an arc, and d_i is a numeric attribute (the duration) of the arc. In the activity-on-arc representation, we also refer to the duration of activity i as the length of arc i (not the actual length of arcs on the graph). The precedence relationships between activities in Figure 1.1 are: activities 3 and 4 cannot start until activity 1 is completed; activity 5 cannot start until activities 2 and 3 are both completed.

1.1.2 Activity-on-node Representation

In the activity-on-node representation, an activity is represented by a node in the network and an event is represented by an arc. The head node (activity) of a directed arc precedes the tail node (activity) of the arc. Similar to the activity-on-arc representation, nodes are indexed in a way such that a directed arc always leads from a node with smaller integer to a larger one. Notice that in the activity-on-node representation, arcs have no index. Moreover, in the activity-on-node representation, the source node and sink node are always dummy nodes (activities) with zero duration and represent no activity in the project. No dummy activities are needed in the A-on-N representation besides the source and sink node. The same example of an AN in Figure 1.1 is shown with the A-on-N representation in Figure 1.2, i.e., Figures 1.2 and 1.1 are two different network representations of the same project. The precedence relationship between activities represented in Figure 1.2 is the same as of Figure 1.1. In Figure 1.2, node 0 and node 6 are dummy nodes (activities) that represent no activities.

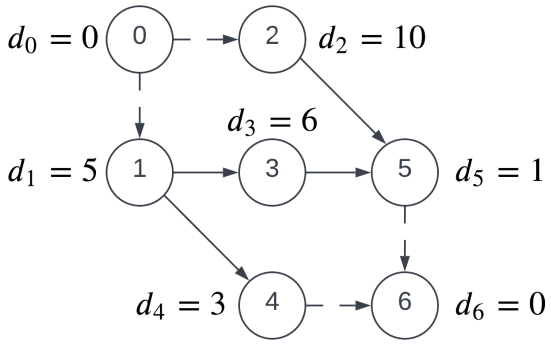


Figure 1.2: Activity-on-node Network Example

1.2 Features and Variables in ANs

1.2.1 Critical Path

We define the notion of a critical path for both A-on-A and A-on-N representations of ANs. In an activity network, a path is a route from the source node to the sink node and is represented by indices of activities that are on the path. The length of a path equals the sum of the durations of activities that are on the path. The path with the longest length is called the critical path, which is not necessarily unique. The path with the longest length is critical, because its length represents the project completion time, and decreasing the duration of activities that are on the critical path can decrease the PCT, whereas decreasing the completion time of activities that are not on the critical path is not helpful for decreasing the PCT.

1.2.2 Node Release Times

The concept of node release time is relevant only for A-on-A representations of ANs. The node release time of a given node j is the earliest time that arcs (activities) whose tail node is j

can start [2]. Since node j may have multiple incoming arcs, it cannot start until all its incoming arcs are completed. The node release time of a given node j, denoted by T_j, is the length of the longest path that starts from the source node and ends at node j. Using the above definition, the node release time of the sink node is the longest path length in the activity network.

A recursive way for calculating the node release time uses the relationship that the node release time of node j equals the maximum of the T_is that are tail node of node j's incoming arcs plus the length of the corresponding incoming arcs.

1.2.3 Project Completion Time

The Project Completion Time (PCT) or makespan [4] is the length of the critical path. There are two approaches to calculate the PCT for activity networks: (1) Calculate the path lengths for all paths and find the longest path length; (2) Use forward dynamic programming to calculate the node release time starting from the source node; the sink node's node release time is the project completion time.

1.2.4 Slack Values

The slack value of an activity i is defined to be the largest amount of time that activity i can be delayed without delaying the project completion time. Activities on the critical path have zero slack values.

1.2.5 Example

We illustrate the two approaches for calculating the PCT using the previous AN example of Figure 1.1. There are three paths in this AN: $\{\{1,3,5\},\{2,5\},\{1,4\}\}$ with their corresponding path lengths $\{d_1+d_3+d_5, d_2+d_5, d_1+d_4\} = \{12,11,8\}$. Path $\{1,3,5\}$ is the critical path and the PCT is 12. The node release time of nodes are given by: $T_1 = 0$, $T_2 = d_1 = 5$, $T_3 = \max\{d_2, T_2+d_3\} = 11$, $T_4 = \max\{T_2+d_4, T_3+d_5\} = 12$. Since node 4 is the sink node, its node release time equals the PCT. For large-scale activity networks, the second approach is more efficient, because identifying all paths for large-scale networks and locating the maximum value of a long list becomes computationally impractical.

In Figure 1.2, the same project with the A-on-N representation, there are three paths in the AN: $\{\{0,2,5\},\{0,1,3,5,6\},\{0,1,4,6\}\}$ with their corresponding path lengths $\{d_0+d_2+d_5, d_0+d_1+d_3+d_5+d_6, d_0+d_1+d_4+d_6\} = \{11,12,8\}$. Of course, in both representations, the number of paths and their corresponding path lengths are the same.

1.2.6 Matrix Representation of Activity Networks

An activity network is represented as a directed acyclic graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{1,...,n\}$ is the set of nodes and $\mathcal{A} \subset \{(i,j) : i,j = 1,2,...,n\}$ is the set of arcs. An arc $(i,j) \in \mathcal{A}$ means there is a directed arc starting from arc i and ending at arc j. For each node $k \in \mathcal{N}$, its predecessor nodes **Pred**(k) and successor nodes **Succ**(k) are defined as [4]:

$$\textbf{Pred}(k) = \{i \in \mathcal{N} : (i,k) \in \mathcal{A}\},\ \textbf{Succ}(k) = \{j \in \mathcal{N} : (k,j) \in \mathcal{A}\}.$$

To form a matrix representation of the activity networks, nodes of an AN with n nodes are indexed in a way that **Succ**$(k) \in \{k+1, ..., n\}$, and the activity network is represented as a $n \times n$ upper triangular square matrix **G** such that:

$$g_{i,j} = \begin{cases} 1, \text{ if } (i,j) \in \mathcal{A}, \\ 0, \text{ otherwise} \end{cases}$$

where $g_{i,j}$ is the entry of the i^{th} row and j^{th} column of matrix **G**. For the A-on-A representation network example in Figure 1.1, its structure matrix **G** and arc length matrix **M** are represented below, where $\mathbf{M}_{[i,j]}$ is the arc length of arc (i,j).

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 0 & 5 & 10 & 0 \\ 0 & 0 & 6 & 3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

For the A-on-A representation of AN in Figure 1.2, its structure matrix **G** and arc lengths vector **v** are represented as:

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 0 \\ 5 \\ 10 \\ 6 \\ 3 \\ 1 \\ 0 \end{bmatrix}.$$

Notice that the structure matrix for two AN representation modes are different. For A-on-A mode, an activity is indexed by a tuple (i,j). For A-on-N mode, an activity is indexed by an integer i. Although in Section 1.1.1, it is mentioned that activities (arcs) in A-on-A mode are indexed by integers, for convenience of problem formulation and programming implementation,

in this thesis, all activities in A-on-A mode representation are indexed by tuples (i, j), where i is the index of the tail node of the arc (activity) and j is that of the head node.

1.3 Stochastic Activity Network

Thus far, we have only discussed deterministic activity networks. In a Stochastic Activity Network (SAN), some or all of activities durations (arc lengths) are random variables. We assume these random variables are independently distributed with known continuous distribution functions $F_i(x)$. Then the project completion time becomes a random variable Y. In SANs, whether a path is critical and whether an arc is on the critical path is not certain. Therefore, we have two new concepts for a SAN: the criticality index of path j, denoted by $C_p(j)$, the probability that path j is a critical path; the criticality index of arc i, denoted by $C_a(i)$, the probability that arc i is on the critical path. For SANs, we are interested in estimating:

- The expectation of project completion time: $\mathbb{E}(Y)$
- The distribution function of project completion time: $F_Y(y)$
- Criticality index of arcs: $C_a(i)$

Criticality of arcs is of interest to project managers, since decreasing the completion time of arcs with high criticality index can decrease the PCT with high probability. In Chapter 4, criticality index of arcs plays an important role for optimization problems in SANs.

1.4 Generating Random SANs

To test various algorithms on a wide variety of problem settings, numerical experiments are performed on stochastic networks with different sizes and structures. To generate stochastic networks with different sizes and structures, we consider two random activity network generator algorithms proposed in [5], the deletion method (DM) and the addition method (AM), both of which are used for generating A-on-A mode ANs. The AM algorithm takes input n as the number of nodes and m as the number of arcs, and outputs a matrix representation of a randomly generated network with n nodes and m arcs. For example, with input $n = 5$, $m = 8$, the AM algorithm outputs an upper triangular matrix and activity network structure, as shown in Figure 1.3.

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

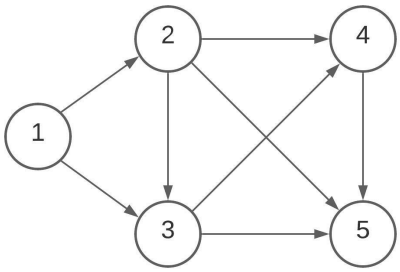


Figure 1.3: Randomly Generated AN

For a randomly generated SAN, not only is the structure of the network randomly generated; parameters of the distributions of the arc lengths are also randomly generated. The details

about how distribution parameters are randomly generated will be introduced in the numerical experiments section.

1.5 Applications of SANs

Project Management. SAN was first applied to project management [3]. The mostly commonly used project schedule techniques are project evaluation and review technique (PERT) and critical path method (CPM) [3]. In [6], PERT was employed to quantitatively design the emergency disposal operation procedure of deepwater drilling riser fracture failure.

Parallel Computing. SAN can also be applied to parallel computing multiprocessor algorithms [7]. A given task is be to executed on a system. In order to use parallel computing, the task needs to be decomposed into a set of subtasks. There are precedence relationships between subtasks. Therefore, subtasks completion time and their precedence relationships can be represented by SANs. The parallel computing algorithm uses SANs to determine which subtasks are computed in parallel.

Digital circuit design. SAN can also be used to model delay in a digital combinational logic circuit [4]. For a given digital circuit, we associate each gate with a node of an activity network. The precedence relationship between nodes represents the signal flow graph. The delay time of a gate represents the time it takes for the signal to get through the gate, and the critical path length here represents the cycle time of the digital circuit. Techniques of estimating SANs can be used for digital circuit sizing.

1.6 Outline of Thesis

In Chap. 2, we introduce estimators for different performance measures in SANs, including criticality index of arcs, criticality index of paths, and expected project completion times. We propose a new arc criticality estimator called the threshold arc criticality (TAC) estimator and compare it with existing estimators. The TAC estimator is shown to have low variance and low computing time in numerical experiments. Chap. 3 reviews different stochastic derivative estimation techniques that will be used in Chap. 4, including measure-based methods and sample-based methods. Chap. 4 derives the stochastic gradient estimators for the performance measures in Chap. 2 and also the stochastic gradient estimators of first and second moment of PCT. Existing gradient estimators for moments of PCT cannot be extended to higher-order derivatives. Using a technique similar to that used in deriving the TAC estimator, we propose new higher-order stochastic gradient estimators for the first and second moments of PCT. Combining the higher-order stochastic gradient estimators for the first and second moments of PCT, we derive formulas that can estimate higher-order gradients for the variance of PCT. In Chap. 5, gradient-based algorithms for solving time-cost tradeoff optimization problems in SANs are presented. Using the higher-order gradient estimators in Chap. 4 in a Taylor series approximation, we propose a new functional estimation of the arc criticality and gradient of expected PCT. We also propose a new algorithm called Knapsack Ratio (KR) algorithm for solving the time-cost tradeoff optimization problems. The KR algorithm is shown to require less computing time than existing algorithms. In Chap. 6, optimal crashing of AN with disruptions is introduced and solved with a new gradient-based simulation optimization algorithm. We introduce a new two-stage stochastic programming with indicator functions formulation for crashing of AN with a

single disruption. Our formulation improves upon an existing two-stage mixed integer stochastic programming formulation in two ways: (1) it can handle a more general class of disruption problems; (2) it reduces the computational time significantly. Under our formulation, a new stochastic gradient estimator is derived and proved to be unbiased. Using the new gradient estimator, a heuristic gradient-based simulation optimization algorithm is proposed and shown in numerical experiments to have significantly lower computational cost for the same statistical performance compared to existing algorithms.

1.7 Research Contributions

We view our main research contributions as the following:

- Developed a new arc criticality estimator called threshold arc criticality (TAC) estimator that is computationally efficient when the number of paths in the network is not too large.
- Derived higher-order stochastic gradient estimators for first and second moments of PCT in SANs, applied to estimate changes in PCT caused by large perturbations of distributional parameters.
- Developed a new algorithm called knapsack ratio (KR) algorithm for time-cost tradeoffs optimization problems, which when tested on large networks takes less computation time to find the optimal solution compared with existing algorithms.
- Proposed a new formulation for optimal crashing AN with disruptions, which can handle more general cases compared to an existing formulation.
- Derived new unbiased stochastic gradient estimators for a two-stage stochastic programming

problem with indicator functions.

- Developed a gradient-based optimization algorithm for optimal crashing AN with a single disruption, which compared with existing SAA-based algorithms, has significantly lower computational cost for comparable solutions.

Chapter 2: SAN Performance Estimators

As mentioned in Section 1.4, the following output performance metrics are of interest to project managers: criticality index of arcs (activities), criticality index of paths, and distribution function of the PCT, all of which are in the form of an expectation. Therefore, Monte Carlo simulation techniques are one approach for estimating the above. When estimating the criticality indexes and distribution function using Monte Carlo simulations, the original estimators all involve indicator functions, which present challenges when estimating the gradient in later chapters. Therefore, smoothed perturbation analysis (SPA) estimators [8] are used here by conditioning on a subset of activities' durations.

The SPA estimators have two advantages: (1) conditioning and unconditioning can reduce the variance of the estimator according to the law of total variance; (2) the estimator can be smoothed (higher-order continuously differentiable) by conditioning. For the SPA estimators in SANs, the choice of the set of activities' durations (arcs' lengths) to be conditioned on is key. A good choice of the set of arcs to be conditioned on can: (1) reduce the dimension of integration; (2) make the conditioned estimator smoother.

Sigal et al. [9] proposed a class of sets called Uniformly Directed Cutsets (UDC) whose complement can form a best set of arcs to be conditioned on. Using UDC [1], the dimension of the exprected value integral is reduced as much as possible and the indicator estimator becomes

a product of known distribution functions by conditioning. The definition and a method to find a UDC is introduced in Section 2.1.2. Then an example on how to find and use UDC is presented in Section 2.1.3. UDC is powerful for estimating criticality of arcs and paths, and also the distribution function of PCT in small ANs [1].

Although UDC works well for small networks, it does not work well for large and complex ANs. Thus, Bowman [2] proposed that instead of conditioning on a subset of arc lengths, it is better to condition on node release times. Hence, a new criticality index of arcs estimator conditioned on node release times is proposed in [2], named conditional arc criticality (CAC) estimator. We propose yet another set to be conditioned on: all arc lengths except for the length of one target arc, and proposed a new arc criticality estimator conditioned on the proposed set of arcs, named threshold arc criticality (TAC) estimator [10]. The performance of the TAC and CAC estimators are compared in numerical experiments on randomly generated ANs of different sizes and arc length distributions.

UDC and TAC can also be applied to estimate the distribution function of PCT.

2.1 Criticality Index of Paths

2.1.1 Criticality Index of Paths

Suppose there are n_p paths in the SAN and all paths are indexed by integers. The criticality index of path i is defined to be the probability that path i is the critical path, i.e. $C_p(i) = \Pr(Y_i \geq Y_j; 1 \leq j \leq n_p, j \neq i)$, where $C_p(i)$ denotes the criticality index of path i, Y_j denotes the length of path j. Without loss of generality, assume $1 < i < n_p$, then $C_p(i)$ can be calculated from the

integral:

$$C_p(i) = \int_{\mathbb{R}^m} \cdots \int \Pr(Y_i \geq Y_1, ..., Y_i \geq Y_{i-1}, Y_i \geq Y_{i+1}, ..., Y_i \geq Y_{n_p} \Big| X_i, 1 \leq i \leq m) \times \prod_{i=1}^{m} f_i(x_i)\, dx_i,$$

where X_i is the arc length (activity duration) of activity i, and the Monte Carlo estimator is given by the average over all simulated data-tuples $\underline{X}$ of:

$$I\{Y_i \geq Y_1, ..., Y_i \geq Y_{i-1}, Y_i \geq Y_{i+1}, ..., Y_i \geq Y_{n_p}\}, \tag{2.1}$$

where (2.1) is an indicator function taking value 1 if path i is a critical path and 0 otherwise.

Notice that the path length Y_j is the sum of the lengths of the arcs (durations of activities) that are on path j, i.e., $Y_j = \sum_{i \in P_j} X_i$. The event inside (2.1) contains n_p events, $\{Y_i \geq Y_j\}, j = 1, 2, ..., n_p$, that are not independent since an arc (activity) can be on more than one path. Our goal is to choose a set of arcs whose length is to be conditioned on, so that the expectation of (2.1) reduces to a probability of independent events. To find such a set, we will present the definition of Uniformly Directed Cutset (UDC) [1] and show that by conditioning on a subset of activities' durations that relates to UDC, (2.1) can be expressed as a product of known distribution functions. Also, using UDC, the integral dimension of (2.1) can be reduced.

2.1.2 Uniformly Directed Cutset

Sigal et al. [9] defines a set of arcs called a Uniformly Directed Cutset (UDC), whose complement together with an arc in UCD forms the best subset of arcs to be conditioned on, so that the corresponding SPA estimator has lower variance and is smoother as a function of sample

inputs $\{X_i\}$. This section introduces the definition of UDC and how to find UDC in an AN.

The challenge in expressing (2.1) as a product of known distribution functions is that an arc can be on more than one path in the network, which means the events inside the probability in (2.1) may be dependent. Thus, one necessary condition of a target subset of arcs in the network to be conditioned on is that in its complement no two arcs are on the same path. A set of arcs is called path independent if no two arcs in the set belongs to the same path. Suppose Ψ is a set of path-independent arcs and Ψ^c is its corresponding complement set. Then we have

$$\Pr(Y_i \geq Y_1, ..., Y_i \geq Y_{i-1}, Y_i \geq Y_{i+1}, ..., Y_i \geq Y_{n_p} \Big| \Psi^c) = \prod_{j \neq i} \Pr(Y_j \leq Y_i | \Psi^c)$$

and the criticality index of path is calculated by unconditioning:

$$C_p(i) = \int \cdots \int \Pr(Y_i \geq Y_1, ..., Y_i \geq Y_{i-1}, Y_i \geq Y_{i+1}, ..., Y_i \geq Y_{n_p} \Big| \Psi^c) dF_{\Psi^c} \tag{2.2}$$

where F_{Ψ^c} is the joint distribution function of arcs in the set Ψ^c. We want the cardinality of the path-independent set $|\Psi|$ to be as large as possible, so that $|\Psi^c|$ is as small as possible and as a result the integration dimension of Equation (2.2) is minimized.

In [1], the path-independent set with the largest cardinality is denoted χ^*, and it was proved that χ^* is indeed a cutset. A cutset C of an activity network is a set of arcs that 'cut' the set of nodes into two sets: W containing the source node and its complement $\bar{W}$ containing the sink node, such that all paths in the network have at least one arc in C. Furthermore, Sigal et al. [1] proved that χ^* is also a uniform directed cutset (UDC) [1]. The definition of UDC is a cutset such that all arcs in the cutset are oriented from W to $\bar{W}$, and such that each network path has exactly

one arc in the cutset. Therefore, χ^* has two important properties: (1) no two arcs in χ^* belong to the same path; (2) each network path has exactly one arc in χ^*. Therefore, to find the UDC set, we only need to find the path-independent set with largest cardinalitywhich is not necessarily unique.

Suppose we are interested in estimating the criticality index of path i. We estimate the probability that path i is the critical path by conditioning on the complement of the UDC set in the network $\bar{\chi}^*$ together with an arc in UCD that is on path i, x_j. Then the conditional criticality index of path l is denoted by K_i:

$$K_i = \Pr(Y_1 \leq Y_i, ..., Y_{n_p} \leq Y_i | \bar{\chi}^*, X_j)$$

And the formulation for K_i is presented in [1] as follows:

$$K_i = \prod_{j \in \chi^*, j \notin P_i} b_{ij} F_j(a_j)$$

where,

$$a_j = \min_{P \in \mathcal{P}_j} \left\{ Y_i - \sum_{k \in P, k \neq j} X_k \right\}$$

and

$$b_{ij} = \begin{cases} \text{if the UDC arc } j \text{ which is on path } i \text{ is} \\ 0, \text{on more than one path and } y_i \text{ is less} \\ \text{than all paths containing arc } j, \\ 1, \text{ otherwise} \end{cases}$$

To calculate a_j, first identify the set of paths in the network that contains arc j, $\mathcal{P}_j$. Then, for each

path P in $\mathcal{P}_j$, calculate the different between the length of path i and the length of P with arc j's length excluded, where a_j is the minimum among the set of difference and b is either 0 or 1, since by conditioning, for one of the events $\{Y_j \leq Y_i\}$, all arcs in this event are being conditioned on so that the event is not random but deterministic. Next we will present an example on how to use UDC to calculate criticality of paths in an AN.

2.1.3 Path Criticality Example

The following is an example from [1] indicating the use of UDC in estimating the criticality index of path. Figure 2.1 is a small network with 5 nodes and 8 arcs. Our goal is to estimate the probability that path $\{2, 6, 8\}$ is the critical path. In this example, we have the UDC set given by $\chi^* = \{X_2, X_3, X_4, X_7\}$, and its complement $\bar{\chi}^* = \{X_1, X_5, X_6, X_8\}$. The UDC arc that is on path $\{2 - 6 - 8\}$ is X_2. There are 6 paths in the network and we name path $\{2, 6, 8\}$ number 5. The conditional path criticality of path 5, K_5 is given by:

$$
\begin{aligned}
K_5 &= \mathbb{P}(Y_1 \leq Y_5, ..., Y_6 \leq Y_5 | \bar{\chi}^*, X_2) \\
&= \mathbb{P}(Y_1 \leq Y_5, ..., Y_6 \leq Y_5 | X_1 = x_1, X_5 = x_5, X_6 = x_6, X_8 = x_8, X_2 = x_2) \\
&= \mathbb{P}(X_7 \leq y_5 - x_1, X_4 \leq y_5 - x_1 - x_5, X_4 \leq y_5 - x_1 - x_6 - x_8, x_2 + x_5 \leq y_5, X_3 \leq y_5 - x_8) \\
&= \mathbb{P}(x_2 + x_5 \leq y_5)\mathbb{P}(X_7 \leq y_5 - x_1)\mathbb{P}(X_4 \leq \min\{y_5 - x_1 - x_5, y_5 - x_1 - x_6 - x_8\})\mathbb{P}(X_3 \leq y_5 - x_8) \\
&= b\mathbb{P}(X_7 \leq y_5 - x_1)\mathbb{P}(X_4 \leq \min\{y_5 - x_1 - x_5, y_5 - x_1 - x_6 - x_8\})\mathbb{P}(X_3 \leq y_5 - x_8)
\end{aligned}
$$

where $y_5 = x_2 + x_6 + x_8$ and $b = \Pr(x_2 + x_5 \leq y_5)$. The value of b is either 0 or 1 for the reason that both x_2, x_5 and y_5 are fixed numbers.

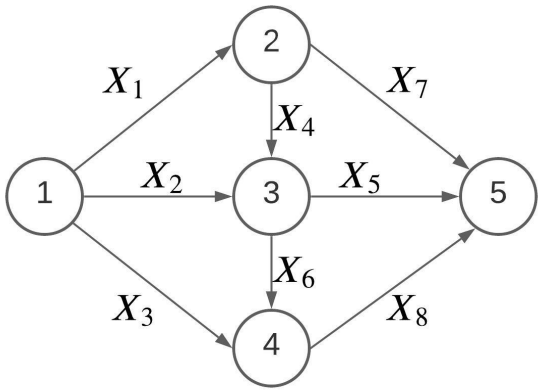


Figure 2.1: UDC Example in [1]

2.2 Criticality Index of Arcs

Criticality index of arc i is defined to be the probability that arc i is on the critical path and denoted by $C_a(i)$. The relationship between criticality index of arcs and paths in an AN is given by [11]:

$$C_a(i) = \sum_{j=1}^{n_p} C_p(j) I\{i \in P_j\} \tag{2.3}$$

that is, the criticality index of arc i equals to the summation of criticality indexes of all paths that contains arc i for the reason that different paths is critical are exclusive events and arc i is on the critical path if and only if one of the paths containing i is the critical path.

2.2.1 Indicator Arc Criticality

The criticality index of an arc is defined to be the probability that the arc is on the critical path. Using Monte Carlo simulation to estimate the criticality of arc i, we simulate all arc length and check whether arc i is on the critical path. And we assign 1 to output if arc i is on the critical path and 0 otherwise. Therefore the first arc criticality estimator is called Indicator Arc Criticality

(IAC) and is derived from the following:

$$C_a(i) = \int_{\mathbb{R}^m} \cdots \int I\{\|\mathcal{P}_i\| = \|\mathcal{P}\|\} \times \prod_{i=1}^{m} f_i(x_i)\, dx_1 \ldots dx_m \tag{2.4}$$

and the IAC estimator is given by: $I\{\|\mathcal{P}_i\| = \|\mathcal{P}\|\}$. In Equation (2.4), $\mathcal{P}_i$ is the set of path that contains activity i and $\mathcal{P}$ is the set of all paths in the AN. $\|\cdot\|$ here is an operator that calculates the length of the longest path in the given set.

2.2.2 Conditional Arc Criticality

Bowman [11] states that the criticality index of arcs can be calculated using Equation (2.3). Together with the UDC conditioning applied, the variance of the smoothed perturbation Monte Carlo estimator can be further reduced. And Bowman [11] also states the drawbacks of UDC set and Equation (2.3) in calculating the criticality index of arcs:

Bowman proposed a new criticality index estimator called Conditional Arc Criticality (CAC) estimator conditioned on node release times in [11]. The node release time of node i is defined to be the earliest starting time of activities with node i as its head node. The CAC estimator is based on a fact that arc i is on the critical path if and only if two conditions are satisfied: (1) arc i is on the longest path that start from the source node and end at the head node of arc i; (2) head node of arc i itself is on the critical path. Figure 2.2 explains the formulation of the CAC estimator. The CAC estimator is derived from the following:

$$C_a(i) = \int_{\mathbb{R}^n} \cdots \int C_a(i|T_1 = t_1, ..., T_n = t_n) \times f_{T_1,...,T_n}(t_1, ..., t_n)\, dt_1 \ldots dt_n, \tag{2.5}$$

and the CAC estimator is given by:

$$C_a(i|T_1 = t_1, ..., T_n = t_n) = \frac{f_i(t_* - t_i) \prod_{j=1, j\neq i}^{q} F_j(t_* - t_j)}{\sum_{k=1}^{q} f_k(t_* - t_k) \prod_{j=1, j\neq k}^{q} F_j(t_* - t_j)} \times C_n(*|T_1 = t_1, ..., T_n = t_n) \tag{2.6}$$

and

$$C_n(*|T_1 = t_1, ..., T_n = t_n) = \sum_{j=q+1}^{q+s} C_a(j|T_1 = t_1, ..., T_n = t_n). \tag{2.7}$$

The CAC estimator in Equation (2.6) consists of two parts, the left part as a likelihood ratio and the right part as a summation. The left part calculates the likelihood that arc i is on the longest path that starts from the source node and ends at the $*$ node. Arc i is on the longest path that ends at node $*$ if and only if $X_j = T_* - T_j$, when $j = i$ and $X_j < T_* - T_j$, when $j \neq i$. The right part calculates the probability that one of the arcs ejecting from the $*$ node is on the critical path. Thus, Equation (2.6) utilizes the two sufficient and necessary conditions mentioned before to calculate the probability that an arc is on the critical path. Equation (2.7) makes the calculation of the CAC estimator a recursive process. And in Equation (2.6), $C_n(*)$ stands for the criticality index of node, which is the probability that node $*$ is on the critical path.

One thing to notice here is that the joint density function of node release times $f_{T_1,...,T_n}(t_1, ..., t_n)$ is not known in Equation (2.5). Since we are using Monte Carlo simulation to estimate Equation (2.5), the joint density function is not necessarily to be known in order to estimate $C_a(i)$. At each simulation replication, only the arc lengths of all arcs are simulated and the node release times are then calculated based on the simulated arc lengths. The calculated node release times are jointly distributed with density function $f_{T_1,...,T_n}(t_1, ..., t_n)$.

In [2], Bowman states that one option to estimate the criticality index of arc i is to firstly

estimating the criticality index of all paths that include arc i and then applying the formula $C_a(i) = \sum_{j=1}^{n} C_p(j) I\{\text{arc } i \text{ is on path } j\}$. When estimating $C_p(j)$, the UDC technique is used. And Bowman states three drawbacks of the above approach: (1) the cardinal number of a UDC set for a large scale activity network is relatively small compared to the network size; (2) Identifying the UDC set is computationally intensive for large scale activity networks; (3) Translation from path criticality to arc criticality takes more time [2].

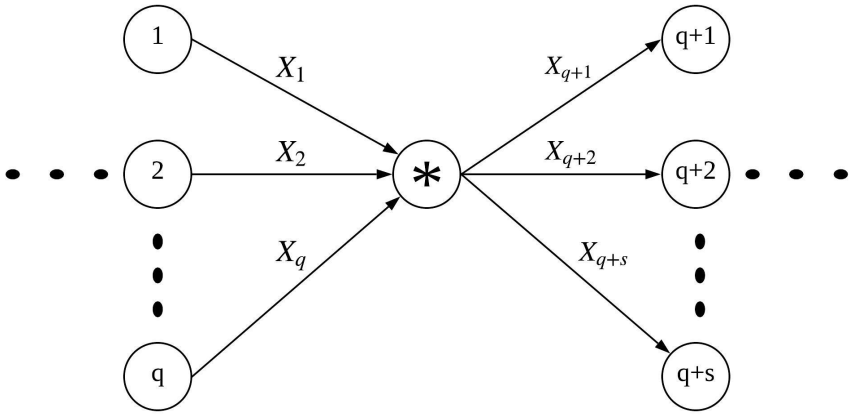


Figure 2.2: Focused Network View in [2]

2.2.3 Threshold Arc Criticality

We proposed a new arc criticality estimator called the Threshold Arc Criticality (TAC) in [10]. The TAC estimator of arc i is conditioned on all arc lengths except for the length of arc i. It is derived from the fact that given all the arc lengths except for arc i, arc i is on the critical path if and only if its own length is large enough, in other words, its own arc length is greater or equal to a threshold. The TAC is based on the following lemma from [10]:

Lemma 1. $C_a(i|X_j = x_j, 1 \le j \le m, j \ne i) = \Pr(X_i \ge m_i)$, *where* $m_i = max(\|\mathcal{P}_{i-}\| - \|\mathcal{P}_i\|^i, 0)\Big|_{X_j = x_j, 1 \le j \le m, j \ne i}$ *and it is a continuous function of* $\{x_j\}_{1 \le j \le m, j \ne i}$.

Proof.

$$
\begin{aligned}
C_a(i|X_j = x_j, 1 \le j \le m, j \ne i) &= \mathbb{P}(\text{arc } i \text{ is on the critical path} \mid X_j = x_j, 1 \le j \le m, j \ne i) \\
&= \mathbb{P}(\|\mathcal{P}_i\| = \|\mathcal{P}\| \Big| X_j = x_j, 1 \le j \le m, j \ne i) \\
&= \mathbb{P}(\|\mathcal{P}_i\| = max(\|\mathcal{P}_i\|, \|\mathcal{P}_{i-}\|) \Big| X_j = x_j, 1 \le j \le m, j \ne i) \\
&= \mathbb{P}(\|\mathcal{P}_i\| \ge \|\mathcal{P}_{i-}\| \Big| X_j = x_j, 1 \le j \le m, j \ne i) \\
&= \mathbb{P}(X_i \ge max(\|\mathcal{P}_{i-}\| - \|\mathcal{P}_i\|^i, 0) \Big| X_j = x_j, 1 \le j \le m, j \ne i) \\
&= \mathbb{P}(X_i \ge m_i).
\end{aligned}
$$

□

Lemma 3 is based on the fact that for a realization of all X_is, if the longest path length of all paths that do not include arc i is greater than the longest path length of all paths that include arc i, then arc i is not on the critical path, and arc i is on the critical path otherwise. When arc i's length is 0 and arc i is not on the critical path, the difference of the two longest path lengths (with arc i included and without) is the threshold such that when the length of arc i is greater than the threshold then it is on the critical path.

The TAC estimator is given by:

$$\bar{F}_i(M_i) = 1 - F_i(M_i)$$

where

$$M_i = max(\|\mathcal{P}_{i-}\| - \|\mathcal{P}_i\|^i, 0). \tag{2.8}$$

In Equation (2.8), the operator $\|\cdot\|^i$ calculates the length of the longest path in a set given that the duration of activity i is 0. Applying the TAC estimator, the criticality index of arc i is calculated by integration:

$$\begin{aligned} C_a(i) &= \int_{\mathbb{R}^{m-1}} \cdots \int C_a(i|X_j = x_j, 1 \le j \le m, j \ne i) \times \prod_{\substack{j=1 \\ j \ne i}}^{m} f_j(x_j)\, dx_j \\ &= \int_{\mathbb{R}^{m-1}} \cdots \int \bar{F}_i(M_i) \times \prod_{\substack{j=1 \\ j \ne i}}^{m} f_j(x_j)\, dx_j. \end{aligned} \tag{2.9}$$

2.2.4 Algorithms for Calculating the Threshold

For large scale SANs, calculating the threshold M_i in Equation (2.8) for each simulation replications is critical for using TAC here and the rest of chapters in this thesis. Two situations are discussed here for calculating the threshold M_i: (1) calculate the threshold for one fixed arc; (2) calculate the threshold for all arcs.

For situation one, algorithm 1 is used. Algorithm 1 firstly calculate the length of the longest path that does not include arc i by assigning a very small negative number $-L$ to the length of arc i. Then algorithm 1 secondly calculated the length of the longest path that includes arc i using the similar approach. Such algorithm has appeared in [12].

Algorithm 1: Calculate the Threshold for TAC

Input : Arc Length Matrix $\mathbf{X}_{n\times n}$ and (i^*, j^*) the target arc coordinate.
Output: Threshold value m_{i^*,j^*}

$L \leftarrow \sum_{i,j} |X_{i,j}|$.
$X_{i^*,j^*} \leftarrow -L$.
$v_0 \leftarrow T_s(\mathbf{X}_{n\times n})$, where T_s is the node release time of the sink node.
$X_{i^*,j^*} \leftarrow L$.
$v_1 \leftarrow T_s(\mathbf{X}_{n\times n})$.
return $max\{v_0 - (v_1 - L), 0\}$.

Algorithm 2: Calculate the Threshold for all arcs

Input : Arc Length Matrix $\mathbf{X}_{n\times n}$
Output: Threshold value matrix $\mathbf{M}$

calculate the length of all paths in the network
sort all paths lengths with descending order
for $(i,j) \in A$ **do**
    for all paths including arc (i,j), find their maximum length v_1
    for all paths excluding arc (i,j), find their maximum length v_0
    $\mathbf{M}[i,j] \leftarrow max\{v_0 - v_1, 0\}$
end

As for situation 2 when calculating the threshold for all arcs in the network, algorithm 2 is applied. In algorithm 2, it firstly calculate all path lengths and sorted them with a descending order. Then the threshold for each arcs are calculated as the difference between longest path length including the excluding the given arc.

2.2.5 Arc Criticality Example

The example used here is the AN presented in Figure 1.1. And the realization of the random duration of all arcs is given by: $X_1 = d_1 = 5, X_2 = d_2 = 10, X_3 = d_3 = 6, X_4 = d_4 = 3, X_5 = d_5 = 1$. Under such realizations, the TAC estimator and the CAC estimator for arc 2 are calculated below:

For the TAC estimator, we have $\mathcal{P}_{2-} = \{\{1,3,5\},\{1,4\}\}$, $\mathcal{P}_2 = \{\{2,5\}\}$, so that $\|\mathcal{P}_{2-}\| = \max\{X_1 + X_3 + X_5, X_1 + X_4\} = \max\{5+6+1, 5+3\} = 12$, and $\|\mathcal{P}_2\|^2 = 0 + X_5 = 1$. Therefore, we have $M_2 = \max(\|\mathcal{P}_{2-}\| - \|\mathcal{P}_2\|^2, 0) = 11$, and the TAC estimator is given by $\Pr(X_2 \geq 11) = 1 - F_2(11)$.

For the CAC estimator, we have $C_n(4) = 1$, because the sink node is on every paths. And we have calculated the node release times in Section 1.2.5, $T_1 = 0, T_2 = 5, T_3 = 11, T_4 = 12$.

Then we have the CAC estimator given by:

$$
\begin{aligned}
C_a(5) &= \frac{f_5(T_4 - T_3)F_4(T_4 - T_2)}{f_5(T_4 - T_3)F_4(T_4 - T_2) + f_4(T_4 - T_2)F_5(T_4 - T_3)} C_n(5) \\
&= \frac{f_5(1)F_4(7)}{f_5(1)F_4(7) + f_4(7)F_5(1)} \\
C_a(2) &= \frac{f_2(T_3 - T_1)F_3(T_3 - T_2)}{f_2(T_3 - T_1)F_3(T_3 - T_2) + f_3(T_3 - T_2)F_2(T_3 - T_1)} C_a(5) \\
&= \frac{f_2(11)F_3(6)}{f_2(11)F_3(6) + f_3(6)F_2(11)} C_a(5)
\end{aligned}
$$

2.2.6 Numerical Result of Arc Criticality

Let $C'_a(i)$ denote the estimation for $C_a(i)$ obtained from the IAC estimator, $C''_a(i)$ denotes the estimation for $C_a(i)$ obtained from the TAC estimator, and $C'''_a(i)$ denotes the estimation for $C_a(i)$ obtained from the CAC estimator, then for N simulation replications:

$$
\begin{aligned}
C'_a(i) &= \frac{\sum_{k=1}^{N} I_i^{(k)}}{N}. \\
C''_a(i) &= \frac{\sum_{k=1}^{N} \bar{F}_i^{(k)}}{N}. \\
C'''_a(i) &= \frac{\sum_{k=1}^{N} C_a^{(k)}(i|T_1, ..., T_n)}{N}
\end{aligned}
$$

where $I_i^{(k)}$ is the IAC estimator of activity i obtained from the k^{th} simulation replication result; $F_i^{(k)}$ is the TAC estimator of activity i obtained from the k^{th} simulation replication result; and $C_a^{(k)}(i|T_1, ..., T_n)$ is the CAC estimator of activity i obtained from the k^{th} simulation replication result.

The following lemma from [10] compares the variance performance of the TAC estimator and the IAC estimator:

Lemma 2. *VAR*$(C_a''(i)) \leq$ *VAR*$(C_a'(i))$, $\forall i$, $\forall N$.

Proof. From the law of total variance, we have that

$$\begin{aligned} Var(I_i^{(k)}) &= \mathbb{E}[Var(I_i^{(k)}|X_j, 1 \leq j \leq m, j \neq i)] + Var(\mathbb{E}[I_i^{(k)}|X_j, 1 \leq j \leq m, j \neq i]) \\ &= \mathbb{E}[Var(I_i^{(k)}|X_j, 1 \leq j \leq m, j \neq i)] + Var(F_i^{(k)}) \end{aligned}$$

since we have that $\mathbb{E}[Var(I_i^{(k)}|X_j, 1 \leq j \leq m, j \neq i)] \geq 0$, we have $Var(F_i^{(k)}) \leq Var(I_i^{(k)})$. Since all simulation replications are independent, we have that

$$Var(C_a''(i)) = Var(\frac{\sum_{k=1}^{N} \bar{F}_i^{(k)}}{N}) = \frac{Var(F_i^{(k)})}{N} \leq \frac{Var(I_i^{(k)})}{N} = Var(\frac{\sum_{k=1}^{N} \bar{I}_i^{(k)}}{N}) = Var(C_a'(i))$$

□

Also, we have that VAR$(C_a'''(i)) \leq$ VAR$(C_a'(i))$, $\forall i$, $\forall N$ from [2]. As for the comparison between VAR$(C_a''(i))$ and VAR$(C_a'''(i))$, no theoretical proof can be shown. Hence, numerical experiments are used for comparing the variance performance between the CAC and TAC estimators.

The CAC and TAC estimators are tested on randomly generated activity networks of different sizes. A network with n nodes and m arcs is firstly randomly generated using the AM algorithm mentioned in Section 1.4. Then the distributional parameters for each arcs in the network are randomly generated. For exponentially and normally distributed arc lengths, their mean μ are uniformly sampled from $[0.5, 15]$. For a normally distributed arc length, its standard deviation σ is generated as $\sigma = 0.25\mu$. For a gamma distribution, its shape parameter k is uniformly sampled from $[0.5, 9]$ and scale parameter θ is uniformly sampled from $[0.5, 2.5]$. For a uniform distribution, its lower bound is uniformly sampled from $[0.5, 5.5]$ and its interval length is uniformly

sampled from $[2, 10]$. For a triangle distribution with parameters $a \leq c \leq b$ and $a < b$, a is uniformly sampled from $[1, 5]$, $b - a$ is uniformly sampled from $[3, 8]$ and c is uniformly sampled from $[a, b]$.

Tables 2.1 - 2.5 provide 95% confidence intervals (C.I.) for a representative randomly selected activity, which indicates that the TAC and CAC estimators have similar accuracy.

Table 2.1: 95% C.I. of Normally Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.30 ± 0.04	0.80 ± 0.05	0.59 ± 0.06	0.65 ± 0.06	0.68 ± 0.05	0.80 ± 0.05
CAC	0.30 ± 0.03	0.80 ± 0.05	0.58 ± 0.05	0.65 ± 0.06	0.69 ± 0.05	0.80 ± 0.05

Table 2.2: 95% C.I. of Exponentially Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.42 ± 0.05	0.46 ± 0.06	0.43 ± 0.05	0.45 ± 0.06	0.78 ± 0.05	0.39 ± 0.06
CAC	0.44 ± 0.04	0.47 ± 0.06	0.43 ± 0.05	0.45 ± 0.06	0.78 ± 0.04	0.38 ± 0.06

Table 2.3: 95% C.I. of Gamma Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.18 ± 0.05	0.89 ± 0.03	0.24 ± 0.05	0.62 ± 0.05	0.36 ± 0.06	0.83 ± 0.05
CAC	0.18 ± 0.05	0.89 ± 0.02	0.24 ± 0.05	0.60 ± 0.06	0.34 ± 0.06	0.86 ± 0.04

Table 2.4: 95% C.I. of Uniformly Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.68 ± 0.06	0.80 ± 0.03	0.74 ± 0.04	0.89 ± 0.03	0.90 ± 0.03	0.84 ± 0.05
CAC	0.67 ± 0.05	0.79 ± 0.03	0.72 ± 0.03	0.87 ± 0.03	0.89 ± 0.03	0.83 ± 0.04

Table 2.5: 95% C.I. of Triangularly Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.76 ± 0.04	0.55 ± 0.06	0.21 ± 0.04	0.56 ± 0.05	0.27 ± 0.05	0.34 ± 0.05
CAC	0.76 ± 0.05	0.56 ± 0.05	0.21 ± 0.04	0.54 ± 0.05	0.27 ± 0.04	0.34 ± 0.05

Table 2.6 is an aggregated version of Tables 2.1 - 2.5. For each of network size settings in Table 2.6, three independent networks are generated. And for each networks, arc criticalities of all arcs are estimated by the CAC and the TAC estimators for 200 simulation replications with common seeds. The sample standard deviation of criticality estimator for each activities is calculated. Then the averaged sample standard deviation (STD) over all activities in the AN is calculated and used as a measure of the estimator's variance performance for all activities in an AN. For each randomly generated networks, the ratio of averaged STD of TAC estimator over CAC estimator is calculated. And for each network size settings in Table 2.6, each element in the table is the averaged ratio value of three independent networks. Table 2.6 shows that for different AN sizes and different arc distributions, the variance performance of the TAC estimator and the CAC estimator are very close.

Notice that by conditioning on node release times, the dimension of integration in Equation

(2.5) is reduced significantly from Equation (2.9) for ANs with more arcs than nodes. However, in Table 2.6, for ANs with 100 nodes and 300 arcs, the sample variance performance for the TAC estimator and CAC estimator is very close to each other, in which case the dimension of integration for the CAC estimator is 100 and that for the TAC estimator is 299, nearly three times as that of the CAC estimator. The reason is that the node release times in an AN are not independent (they are positively correlated), while the activity durations are all independent. Therefore, although the CAC estimator reduces the dimension of integration significantly, the sample variance of the estimator is not reduced significantly compared to that of the TAC estimator.

Table 2.6: Averaged Sample Standard Error of TAC/CAC (based on 200 Independent Replications, elements unit 10^{-4})

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
Arc Distributions	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
Exponential	120/113	83/72	93/84	70/60	71/66	52/46
Normal	60/65	52/51	46/45	26/25	49/45	29/27
Gamma	42/38	70/64	74/72	49/47	50/49	43/42
Uniform	77/74	44/40	85/86	55/52	46/44	15/13
Triangular	71/57	75/75	49/49	33/30	36/35	13/14

Table 2.7: Time Ratio of TAC/CAC (based on 200 Independent Replications, number of paths in parentheses)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
Arc Distributions	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
Exponential	0.29 (42)	1.65 (391)	0.28 (64)	0.94 (247)	0.66 (245)	2.32 (956)
Normal	0.51 (50)	0.99 (162)	0.62 (73)	1.45 (561)	0.58 (123)	1.56 (832)
Gamma	0.44 (40)	0.55 (155)	0.43 (77)	0.71 (436)	0.42 (125)	1.15 (995)
Uniform	0.30 (43)	1.63 (337)	0.28 (65)	2.35 (660)	0.29 (112)	2.44 (983)
Triangular	0.45 (88)	1.04 (220)	0.29 (74)	1.12 (333)	0.38 (144)	2.58 (975)

Since activity networks are randomly generated, the number of paths of the randomly

generated ANs cannot be easily controlled by the AM algorithm. Since we need to estimate the criticality indexes for all activities in an AN, algorithm 2 is used for calculating the TAC estimators. The values in Table 2.7 are the ratio of the computation time of the TAC estimator over that of the CAC estimator. From Table 2.7, it can be concluded that for ANs with number of paths less than 200, the TAC estimator computes faster than the CAC estimator. For networks with number of paths greater than 200, the CAC estimator computes faster than the TAC estimator. Algorithm 2's time complexity is $\mathcal{O}(n_p \log(n_p))$, where n_p is the number of paths in the AN and from Equation (2.6), we can see that the time complexity of the CAC estimator for estimating criticality index of all activities is $\mathcal{O}(n_a)$, where n_a is the number of arcs (activities) in the AN, as a result of which, it takes more time to estimate criticality index of all activities using the TAC estimator than using the CAC estimator when the number of paths is large. Also, since we need to use the density and distribution function several times for each estimation using Equation (2.6), the complexity of the density function and distribution function of arc lengths can affect the computing speed of the CAC estimator. In Table 2.7, the time ratio for gamma distribution of same network size and similar number of paths is smaller than other distributions.

2.3 Distribution of PCT

For projects with a deadline, the distribution function of PCT is of great interest to project managers. Suppose the deadline is t. And we are interested in estimating $\Pr(Y > t)$. A direct Monte Carlo estimation of $\Pr(Y > t)$ is given by:

$$\Pr(Y > t) = \int_{\mathbb{R}^m} \cdots \int I\{Y > t\} \times \prod_{i=1}^{m} f_i(x_i)\, dx_i$$

For each simulation replications, all arcs' lengths are simulated. And the estimator is an indicator function whose value is 1 if PCT value is greater than t and 0 otherwise. After large enough number of simulation replications, the mean of the estimator converges to the true distribution function value almost surely.

2.3.1 Conditional PCT Distribution Estimator

Fu [8] proposed a conditional distribution of the PCT estimator derived from

$$\Pr(Y > t | X_j = x_j, 1 \leq j \leq m, j \neq i) = \begin{cases} 1, \text{ if } \|\mathcal{P}\|^i > t \\ \Pr(\|\mathcal{P}_i\| > t) \text{ otherwise,} \end{cases}$$

where $\Pr(\|\mathcal{P}_i\| > t) = \Pr(X_i + \|\mathcal{P}_i\|^i > t) = \Pr(X_i > t - \|\mathcal{P}_i\|^i) = \bar{F}_i(t - \|\mathcal{P}_i\|^i)$, and the estimator is given by:

$$\bar{F}_i(t - \|\mathcal{P}_i\|^i) I(\|\mathcal{P}\|^i \leq t) + I(\|\mathcal{P}\|^i > t).$$

For each simulation replication, the simulated arc values are $\{X_j\}_{j \neq i}$. Assign 0 value to the length of arc i, together with the simulated other arcs length, the PCT value can be calculated. If the PCT value is greater than t, the estimator equals 0; otherwise, calculate the longest length of all paths that include arc i, $\|\mathcal{P}_i\|^i$, the estimator equals the probability that arc i's length is greater than $t - \|\mathcal{P}_i\|^i$.

2.3.2 UDC PCT Distribution Function Estimator

Sigal [1] and Fu [8] both use UDCs in determining the PCT distributions. Then the conditional PCT distribution function is denoted by $F_Y(t)$:

$$F_Y(t) = \Pr(Y_1 \leq t, ..., Y_{n_p} \leq t | \bar{\chi}^*).$$

And the formulation for $F(t)$ is presented as follows:

$$F_Y(t) = \prod_{j \in \chi^*} F_j(a_j)$$

where, $a_j = t - \|\mathcal{P}_j\|^j$.

2.3.3 PCT Distribution Example

Figure 2.1 is used as the example here, and the realization of arc lengths are given by: $X_1 = 27, X_2 = 6, X_3 = 11, X_4 = 10, X_5 = 12, X_6 = 5, X_7 = 35, X_8 = 9$. And suppose we are interested in estimating the probability that the PCT is greater than 50. Assume the conditioned arcs are all arcs except for arc 1. The conditional PCT distribution estimator is calculated as:

$$\bar{F}_1(t - \|\mathcal{P}_1\|^1) I(\|\mathcal{P}\|^1 \leq t) + I(\|\mathcal{P}\|^1 > t)$$

$$= \bar{F}_1(50 - 35) I(35 \leq 50) + I(35 > 50)$$

As for the UDC approach, we have the UDC set given by $\chi^* = \{X_2, X_3, X_4, X_7\}$, and its

complement $\bar{\chi}^* = \{X_1, X_5, X_6, X_8\}$. Then the UDC PCT distribution estimator is calculated as:

$$F_Y(50) = F_2(50 - 14)F_3(50 - 9)F_4(50 - 41)F_7(50 - 27)$$

Chapter 3: Stochastic Gradient Estimation

The content in this chapter is based on material from [13]. In this chapter, our goal is to estimate

$$\frac{dJ(\theta)}{d\theta},$$

where θ is the parameter of interest and

$$J(\theta) = \mathbb{E}[Y(\mathbf{X})] = \mathbb{E}[Y(X_1, X_2, ..., X_N)].$$

$\{X_i\}$ are input random variables and Y is a measurable output function, $Y : \mathbb{R}^n \to \mathbb{R}$. We will focus on the setting where θ is a distributional parameter of the input r.v.s. The various stochastic gradient estimation techniques are classified into two classes by their treatment of the dependence on θ: sample path-based methods and measure-based methods. The three most popular stochastic gradient estimation techniques are infinitesimal perturbation analysis (IPA), the likelihood ratio (LR) method (also known as the score function method), and the weak derivative (WD) method [14, 15, 16, 17, 18, 19].

3.1 Measure-Based Methods

Assume the joint distribution function of $X_1, X_2, ..., X_N$ has a corresponding density function $f(x_1, x_2, ..., x_N)$. Then express $J(\theta)$ as an integration as follows:

$$\begin{aligned} J(\theta) = \mathbb{E}[Y(\mathbf{X})] &= \int Y(\mathbf{x}) dF_{\mathbf{X}}(\mathbf{x}) \\ &= \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times f(x_1, x_2, ..., x_N; \theta) \prod_{j=1}^{N} dx_j. \end{aligned}$$

Further assume the regularity conditions to exchange derivative and integration is satisfied, we have

$$\frac{dJ(\theta)}{d\theta} = \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times \frac{\partial f(x_1, x_2, ..., x_N; \theta)}{\partial \theta} \prod_{j=1}^{N} dx_j. \tag{3.1}$$

3.1.1 Likelihood Ratio Method

The likelihood ratio (LR) method is also called the score function (SF) method [20, 21, 22]. Since we are using Monte Carlo simulation to estimate the stochastic gradients, we want to express Equation (3.1) in the form of an expectation. Thus, a transformation of the derivative of the density function is applied:

$$\begin{aligned} \frac{dJ(\theta)}{d\theta} &= \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times \frac{\partial f(x_1, x_2, ..., x_N; \theta)}{\partial \theta} \prod_{j=1}^{N} dx_j \\ &= \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times \frac{\partial \ln(f(x_1, x_2, ..., x_N; \theta))}{\partial \theta} f(x_1, x_2, ..., x_N; \theta) \prod_{j=1}^{N} dx_j \\ &= \mathbb{E}\Big[Y(X_1, X_2, ..., X_N) \times \frac{\partial \ln(f(X_1, X_2, ..., X_N; \theta))}{\partial \theta} \Big], \end{aligned}$$

and the LR estimator is given by:

$$Y(X_1, X_2, ..., X_N) \times \frac{\partial \ln(f(X_1, X_2, ..., X_N; \theta))}{\partial \theta}.$$

If $\{X_i\}$ are independent random variables and θ is a distributional parameter of the density function f_1 of X_1 only, then the LR estimator becomes:

$$Y(X_1, X_2, ..., X_N) \times \frac{\partial \ln(f_1(X_1; \theta))}{\partial \theta}.$$

3.1.2 Weak Derivative Method

We assume here that $\{X_i\}$ are independent random variables and θ is the distribution parameter of X_1. Then we have:

$$\frac{\partial f(x_1, x_2, ..., x_N; \theta)}{\partial \theta} = \frac{\partial f_1(x_1; \theta)}{\partial \theta} \prod_{i=2}^{N} f_i(x_i)$$

We can decompose the derivative of the density function as a constant times the difference of two density functions $f_1^{(+)}$ and $f_1^{(-)}$:

$$\frac{\partial f_1(x_1; \theta)}{\partial \theta} = c_1(\theta)\Big(f_1^{(+)}(x_1; \theta) - f_1^{(-)}(x_1; \theta)\Big). \tag{3.2}$$

An example for normally distributed X_1 with mean μ and variance σ^2 is provided here: the density function of X_1 is given by

$$f_1(x_1;\mu) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(- \frac{(x_1-\mu)^2}{2\sigma^2}\Big)$$

and the derivative of the density function of X_1 with respect to μ can be expressed as,

$$\begin{aligned}\frac{\partial f_1(x_1;\mu)}{\partial \mu} =& \frac{(x_1-\mu)}{\sigma^3\sqrt{2\pi}} \exp\Big(- \frac{(x_1-\mu)^2}{2\sigma^2}\Big) \\ =& \frac{1}{\sigma\sqrt{2\pi}}\Big[\frac{(x_1-\mu)}{\sigma^2} \exp\Big(- \frac{(x_1-\mu)^2}{2\sigma^2}\Big) I(x_1>\mu) \\ & - \frac{(\mu-x_1)}{\sigma^2} \exp\Big(- \frac{(x_1-\mu)^2}{2\sigma^2}\Big) I(x_1<\mu)\Big]. \end{aligned} \tag{3.3}$$

Inside the bracket of Equation (3.3), the first part is of the form $\mu + Wei(2, \sqrt{2}\sigma)$ and the second part is of the form $\mu - Wei(2, \sqrt{2}\sigma)$, and the constant term is $\frac{1}{\sqrt{2\pi}}$. The triple $(c_1(\theta), f_1^{(+)}, f_1^{(-)})$ constitutes a weak derivative (WD) for f_1, which is generally not unique. Using the weak

derivative of f_1, we can express the gradient as:

$$\begin{aligned}
\frac{dJ(\theta)}{d\theta} =& \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times \frac{\partial f(x_1, x_2, ..., x_N; \theta)}{\partial \theta} \prod_{j=1}^{N} dx_j \\
=& \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times \frac{\partial f_1(x_1; \theta)}{\partial \theta} dx_1 \prod_{j=2}^{N} f_j(x_j)\, dx_j \\
=& c_1(\theta) \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times (f_1^{(+)}(x_1; \theta) - f_1^{(-)}(x_1; \theta)) dx_1 \prod_{j=2}^{N} f_j(x_j)\, dx_j \\
=& c_1(\theta) \Big(\int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times f_1^{(+)}(x_1; \theta) \prod_{j=2}^{N} f_j(x_j) \prod_{j=1}^{N} dx_j \\
& - \int_{\mathbb{R}^N} \cdots \int Y(x_1, x_2, ..., x_N) \times f_1^{(-)}(x_1; \theta) \prod_{j=2}^{N} f_j(x_j) \prod_{j=1}^{N} dx_j \Big) \\
=& c_1(\theta) \Big(\mathbb{E}\Big[Y(X_1^{(+)}, X_2, ..., X_N) \Big] - \mathbb{E}\Big[Y(X_1^{(-)}, X_2, ..., X_N) \Big] \Big).
\end{aligned}$$

Thus the WD estimator is given by

$$c_1(\theta) \Big(Y(X_1^{(+)}, X_2, ..., X_N) - Y(X_1^{(-)}, X_2, ..., X_N) \Big)$$

where $X_1^{(+)} \sim f_1^{(+)}$ and $X_1^{(-)} \sim f_1^{(-)}$.

3.2 Sample Path-Based Methods

3.2.1 Derivatives of Random Variables

Because the sample path (PA) estimators require the notion of derivatives of random variables, the definition of the derivative of a random variable X is introduced here. We first construct a family of random variables $\{X(\theta)\}$ parameterized by θ on a common probability space. We

construct $X(\theta) \sim F(\cdot;\theta)$ such that $X(\theta)$ is differentiable w.r.t θ $w.p.1$. Then the sample derivative is defined as:

$$\frac{dX(\theta;\omega)}{d\theta} = \lim_{\delta \to 0} \frac{X(\theta+\delta,\omega) - X(\theta,\omega)}{\delta},$$

where ω is a sample point in the sample space Ω. If the distribution function of X is known, we have [23, 24]

$$\frac{dX(\theta)}{d\theta} = -\frac{\partial F(X;\theta)/\partial\theta}{\partial F(X;\theta)/\partial X}. \tag{3.4}$$

3.2.2 Location and Scale Parameters

A distributional parameter θ of a random variable $X \sim F(\cdot;\theta)$ is said to be a *location* parameter if $F(x+\theta;\theta)$ does not depend on θ; θ is said to be a *scale* parameter if $F(x\theta;\theta)$ does not depend on θ; θ is said to be a *generalized scale* parameter if $F(\beta + x\theta;\theta)$ does not depend on θ.

If θ is a location parameter, then $\frac{dX(\theta)}{d\theta} = 1$. If θ is a scale parameter, then $\frac{dX(\theta)}{d\theta} = \frac{X}{\theta}$. If θ is a generalized scale parameter, then $\frac{dX(\theta)}{d\theta} = \frac{X-\beta}{\theta}$.

3.2.3 Infinitesimal Perturbation Analysis

Infinitesimal perturbation analysis (IPA) method is a sample path-based method. IPA may transform the random variables so that the random variable and its distributional parameter are split. For example, if $X \sim \mathcal{N}(\mu,\sigma)$, X can be expressed as $X = \mu + \sigma Z$, where $Z \sim \mathcal{N}(0,1)$. For random variable X with general distributions, an inverse transform method can be applied so that X has the same distribution as $F^{-1}(U;\theta)$, where $U \sim U(0,1)$ [25].

Assuming the regularity condition to exchange derivatives with integration is satisfied and after applying the transformation to change the measure, we have:

$$\frac{dJ(\theta)}{d\theta} = \int_{[0,1]^N} \frac{dY(X(\theta;u))}{d\theta}\, du \tag{3.5}$$

where in the integrand of Equation (3.5), $\frac{dY(X(\theta;u))}{d\theta} = \frac{dY}{dX}\frac{dX}{d\theta}$, the estimation of $\frac{dX(\theta)}{d\theta}$ is needed, which has been introduced in Section 3.2.1.

3.3 Finite Difference Methods

The previous three methods are direct gradient estimation methods, which utilize sample path information and distribution functions to obtain a stochastic gradient estimator. Indirect gradient estimation methods assume the output of the simulation comes out of a black box and take the form of finite difference estimators.

The simplest finite difference (FD) estimator is the one-sided forward difference gradient estimator, given by:

$$\frac{Y(\theta+\delta,\xi_2) - Y(\theta,\xi_1)}{2\delta}.$$

A more accurate estimator is the two-sided symmetric difference estimator is given by:

$$\frac{Y(\theta+\delta,\xi_2) - Y(\theta-\delta,\xi_1)}{2\delta},$$

where $Y(\cdot)$ is the sample performance function, δ is a small perturbation value, ξ_1 and ξ_2 are two sample points in the sample space. For example, in SANs, Y could be the PCT of the network

and θ could be the mean value of arc 1's length. Then, to obtain the FD estimator, all arc lengths are simulated with the mean of X_1 equal to $\theta + \delta$, and the PCT value is calculated based on the first simulation replication, which we denoted by $y^{(1)}$. Then all arc lengths are resimulated with the mean of X_1 equal to $\theta - \delta$ and the PCT value is calculated based on the second simulation replication, denoted by $y^{(2)}$. The FD estimator is given by $\frac{y^{(1)}-y^{(2)}}{2\delta}$.

One of the drawbacks of FD estimators is that they have large variance. In order to reduce the variance of FD estimators, the common random numbers (CRN) variance reduction technique ($\xi_1 = \xi_2$) is applied here. When applying (CRN) technique to SANs, in the first simulation replication, all arc lengths are simulated, whereas in the second simulation replication, only the arc that θ belongs to is resimulated and all the other arc lengths remain the same as the first simulated values.

Chapter 4: Stochastic Gradient in SANs

In this chapter, we are interested in estimating the stochastic gradient of the sample performance measures introduced in Chapter 2 and moments of PCT with respect to the distribution parameters of arc lengths. The gradient of the criticality index of arcs with respect to distribution parameters of activities durations can help us understand how sensitive the criticality index of arc i is to its mean duration; similarly for the gradient of expected PCT. Higher-order gradients of expected PCT can help with functional approximation of the expected PCT as a function of distribution parameters. In Chapter 5, it is shown that the first-order gradient of expected PCT reduces to the arc criticality index if the distribution parameter is a location parameter.

It is obvious that the expected PCT is an increasing function of an activity's mean duration. However, Elmaghraby [26] notes that the variance of expected PCT is not a monotone function of an activity's mean duration. To better understand how variance of PCT changes when the mean duration of an activity changes, higher-order gradient estimation of variance of PCT is important. To estimate the gradient of the variance of PCT, we estimate the gradient of the second moment of PCT.

For optimization problems with a differentiable objective function, gradient-based methods are commonly used for finding the optimal solution and optimal value. Examples of such gradient-based optimization methods are: (1) Interior-Point methods [27, 28, 29], (2) Stochastic Gradient

Descent methods. Thus, a good stochastic gradient estimator with low variance and low complexity is key for gradient-based optimization algorithms.

In this section, we will introduce stochastic gradient estimators for: criticality index of arcs, first moment of PCT, second moment of PCT, and variance of PCT using the gradient estimation techniques mentioned in Chapter 3.

4.1 Stochastic Gradient of Arc Criticalities

In this section, we are interested in estimating the following:

$$\frac{\partial C_a(i)}{\partial \theta_j}$$

where $C_a(i)$ is the criticality index of arc i, and θ_j is the distribution parameter of X_j. $C_a(i)$ can also be expressed as $\mathbb{E}[I\{\text{arc } i \text{ is on the critical path}\}]$. The different arc criticality estimators in Chapter 2 correspond to different stochastic gradient estimators, because they have different sample performance functions and different probability measures: IAC has the sample performance function as an indicator function; TAC has the sample performance function as a complementary distribution function; CAC has the sample performance function as a recursive formula.

4.1.1 IPA gradient of Arc Criticalities

4.1.1.1 IPA of TAC

Assuming the regularity conditions for exchanging derivative and integral in Equation (2.9) is satisfied, we have

$$\frac{\partial C_a(i)}{\partial \theta_j} = \int_{\mathbb{R}^{m-1}} \cdots \int \frac{\partial \bar{F}_i(M_i)}{\partial \theta_j} \times \prod_{\substack{j=1 \\ j \neq i}}^{m} f_j(x_j)\, dx_j. \tag{4.1}$$

The IPA estimator for TAC arc criticality is from [10]:

$$\begin{aligned}
-\frac{\partial F_i(M_i)}{\partial \theta_i} &= -\frac{dF_i(M_i)}{d\theta_i}, \\
-\frac{\partial F_i(M_i)}{\partial \theta_j} &= \frac{dF_i(M_i)}{dM_i}\frac{dM_i}{d\theta_j} \\
&= \frac{dF_i(M_i)}{dM_i}\frac{d\max(\|\mathcal{P}_{i-}\| - \|\mathcal{P}_i\|^i, 0)}{dM_i} \\
&= \frac{dF_i(M_i)}{dM_i} \times I\{\|\mathcal{P}_{i-}\| - \|\mathcal{P}_i\|^i \geq 0\} \times (I\{j \in P^*_{i-}\} - I\{j \in P^*_i\}) \times \left(\frac{\frac{\partial F_j(X_j)}{\partial \theta_j}}{\frac{\partial F_j(X_j)}{\partial X_j}}\right).
\end{aligned}$$

Assuming the regularity conditions for exchanging derivative and integral in Equation (2.9) n times is satisfied, the high-order derivative estimator for $\frac{\partial^n C_a(i)}{\partial \theta_i^n}$ is given by:

$$-\frac{\partial^n F_i(M_i)}{\partial \theta_i^n}. \tag{4.2}$$

4.1.1.2 IPA of CAC

Assuming the regularity conditions for exchanging derivative and integral in Equation (2.9) is satisfied, we have:

$$\frac{\partial C_a(i)}{\partial \theta_j} = \int_{\mathbb{R}^n} \cdots \int \frac{\partial C_a(i|T_1 = t_1, ..., T_n = t_n)}{\partial \theta_j} \times f_{T_1,...,T_n}(t_1, ..., t_n)\, dt_1 \ldots dt_n,$$

so the IPA estimator for CAC is given by:

$$\begin{aligned}
&\frac{\partial C_a(i|T_1, ..., T_n)}{\partial \theta_i} \\
=&\frac{\partial \frac{f_i(T_*-T_i)\prod_{j=1,j\neq i}^{q} F_j(T_*-T_j)}{\sum_{k=1}^{q} f_k(T_*-T_k)\prod_{j=1,j\neq k}^{q} F_j(T_*-T_j)} \times C_n(*|T_1, ..., T_n)}{\partial \theta_i} \\
=&\frac{\partial \frac{f_i(T_*-T_i)\prod_{j=1,j\neq i}^{q} F_j(T_*-T_j)}{\sum_{k=1}^{q} f_k(T_*-T_k)\prod_{j=1,j\neq k}^{q} F_j(T_*-T_j)}}{\partial \theta_i} \times C_n(*|T_1, ..., T_n) \\
&+ \frac{f_i(T_*-T_i)\prod_{j=1,j\neq i}^{q} F_j(T_*-T_j)}{\sum_{k=1}^{q} f_k(T_*-T_k)\prod_{j=1,j\neq k}^{q} F_j(T_*-T_j)} \times \frac{\partial C_n(*|T_1, ..., T_n)}{\partial \theta_i}.
\end{aligned} \tag{4.3}$$

For the derivatives appearing in the first term of Equation (4.3) above, we have ($i \neq j$):

$$\begin{aligned}
\frac{\partial f_i(T_i - T_*)}{\partial \theta_i} &= \frac{df_i(T_i - T_*)}{d(T_i - T_*)} \frac{\partial (T_i - T_*)}{\partial X_i} \frac{\partial X_i}{\partial \theta_i} + \frac{\partial f_i(T_i - T_*)}{\partial \theta_i}, \\
\frac{\partial f_i(T_i - T_*)}{\partial \theta_j} &= \frac{df_i(T_i - T_*)}{d(T_i - T_*)} \frac{\partial (T_i - T_*)}{\partial X_j} \frac{\partial X_j}{\partial \theta_j}, \\
\frac{\partial T_i}{\partial X_j} &= I\{X_j \text{ is on the longest path that ends at node } i\}.
\end{aligned} \tag{4.4}$$

For calculating the derivatives of F_i, substitute f_i with F_i for the above equations, and we

have

$$\frac{\partial F_i(T_i - T_*)}{\partial \theta_i} = \frac{dF_i(T_i - T_*)}{d(T_i - T_*)} \frac{\partial (T_i - T_*)}{\partial X_i} \frac{\partial X_i}{\partial \theta_i} + \frac{\partial F_i(T_i - T_*)}{\partial \theta_i},$$
$$\frac{\partial F_i(T_i - T_*)}{\partial \theta_j} = \frac{dF_i(T_i - T_*)}{d(T_i - T_*)} \frac{\partial (T_i - T_*)}{\partial X_j} \frac{\partial X_j}{\partial \theta_j}.$$

4.1.2 LR Gradient of Arc Criticalities

The TAC estimator is not available for measure-based gradient estimators when the parameter of interest belongs to X_i, i.e., when we are interested in estimating $\frac{\partial C_a(i)}{\partial \theta_i}$. The reason LR is not applicable for TAC is that in Equation (2.9) the joint density function is independent of θ_i.

The CAC estimator is also not available for measure-based gradient estimators, because the joint density function of node release times $f_{T_1,...,T_n}(t_1, ..., t_n)$ in Equation (2.5) is unknown. In order to use measure-based gradient estimation techniques, we need information about $\frac{\partial f_{T_1,...,T_n}(t_1,...,t_n)}{\partial \theta_j}$, which is unknown.

In conclusion, both the TAC and the CAC estimators are not applicable for measure-based gradient estimation techniques. Measure-based gradient estimation techniques are only applicable to the IAC estimator. The LR estimator of IAC is derived from exchanging derivative and integration of Equation (2.4)

$$\begin{aligned}\frac{\partial C_a(i)}{\partial \theta_j} &= \int_{\mathbb{R}^m} \cdots \int I\{\|\mathcal{P}_i\| = \|\mathcal{P}\|\} \times \frac{\partial \prod_{i=1}^m f_i(x_i)}{\partial \theta_j} \, dx_1 \ldots dx_m \\ &= \int_{\mathbb{R}^m} \cdots \int I\{\|\mathcal{P}_i\| = \|\mathcal{P}\|\} \times \frac{\partial \ln(f_j(x_j))}{\partial \theta_j} \prod_{i=1}^m f_i(x_i) \, dx_1 \ldots dx_m\end{aligned}$$

and the LR estimator for IAC is given by:

$$I\{\|\mathcal{P}_i\| = \|\mathcal{P}\|\} \times \frac{\partial \ln(f_j(x_j))}{\partial \theta_j}. \tag{4.5}$$

4.1.3 WD gradient of Arc Criticalities

As mentioned before, the WD gradient estimation technique is only applicable for the IAC estimator. The WD estimator for IAC $\frac{\partial C_a(i)}{\partial \theta_j}$ is given by:

$$c_1(\theta_j)\Big(Y(X_1, ..., X_j^{(+)}, ..., X_N) - Y(X_1, ..., X_j^{(-)}, ..., X_N)\Big) \tag{4.6}$$

where we have $X_j^{(+)} \sim f_j^{(+)}$ and $X_j^{(-)} \sim f_j^{(-)}$. $(c_1(\theta_j), f_j^{(+)}, f_j^{(-)})$ forms a weak derivative of the distribution of X_j. For details about weak derivatives of different distributions, please refer to [13].

The variance of the WD estimator is given by:

$$\begin{aligned}
&\mathrm{Var}\Big[c_1(\theta_j)\Big(Y(X_1, ..., X_j^{(+)}, ..., X_N) - Y(X_1, ..., X_j^{(-)}, ..., X_N)\Big)\Big] \\
=&(c_1(\theta_j))^2\Big(\mathrm{Var}\Big[Y(X_1, ..., X_j^{(+)}, ..., X_N)\Big] + \mathrm{Var}\Big[Y(X_1, ..., X_j^{(-)}, ..., X_N)\Big] \\
&- 2\mathrm{Cov}\Big[Y(X_1, ..., X_j^{(+)}, ..., X_N), Y(X_1, ..., X_j^{(-)}, ..., X_N)\Big]\Big).
\end{aligned}$$

To minimize the variance of the WD estimator, we want $\mathrm{Cov}\Big[Y(X_1, ..., X_j^{(+)}, ..., X_N), Y(X_1, ..., X_j^{(-)}, ..., X_N)\Big]$ to be as large as possible. In [30], it is shown that the WD gradient estimator for expected PCT with all activities' durations normally distributed outperforms IPA and LR gradient estimators in numerical experiments of small ANs. Heidergott et al. [31] apply

coupled phantoms in reducing the variance of the WD gradient estimator for $\frac{\partial \mathbb{E}[Y]}{\partial \sigma}$, where Y is the PCT of an AN as a Gaussian system and σ is the standard deviation of the system.

4.1.4 Numerical Results of Gradient Estimators of Arc Criticalities

In this section, a random activity network with n nodes and m arcs is first generated using the AM algorithm mentioned in Section 1.4. For a normally distributed arc length, its mean value μ is uniformly sampled from [0.5,15] and its standard deviation is generated as $\sigma = 0.333\mu$. For an exponentially distributed arc length, its mean value μ is uniformly sampled from [0.5,15].

With number of nodes n and number of arcs m fixed, an activity network with all arcs either normally distributed or exponentially distributed is randomly generated. The generated network is random in terms of both the network structure and its arcs' distributional parameters. The criticality indexes of all arcs of the randomly generated network is estimated using the CAC estimator with 1000 simulation replications. Then the set of arcs P with top 10 largest criticality indexes of arcs are chosen as the target list of arcs. The arc criticality gradients $\frac{\partial C_a(i)}{\partial \mu_i}, \forall i \in P$ are estimated using different estimators with the same set of 200 simulation replications. For each $i \in P$, the sample standard deviation (STD) of the estimator and the sample mean (gradient estimation) of the gradient estimator are calculated.

For fixed number of nodes n and number of arcs m, the procedure in the previous paragraph is repeated 3 times. Finally, the sample standard deviation (STD) and computing time are averaged over 30 arcs coming from 3 independently generated SANs with same number of nodes and number of arcs. In Tables 4.1 to 4.6, TAC stands for the IPA gradient estimator of arc criticalities in Equation (4.2), CAC stands for the IPA gradient estimator of arc criticalities in

Table 4.1: 95% C.I. of Gradient of Normally Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.032 ± 0.006	0.046 ± 0.010	0.034 ± 0.005	0.065 ± 0.007	0.037 ± 0.005	0.040 ± 0.006
CAC	0.027 ± 0.005	0.042 ± 0.007	0.033 ± 0.006	0.062 ± 0.005	0.041 ± 0.005	0.016 ± 0.006
LR	0.050 ± 0.035	0.041 ± 0.063	0.030 ± 0.030	0.086 ± 0.037	0.030 ± 0.027	0.025 ± 0.039
WDC	0.032 ± 0.007	0.036 ± 0.011	0.035 ± 0.006	0.071 ± 0.009	0.028 ± 0.014	0.036 ± 0.011
WDNC	0.023 ± 0.009	0.046 ± 0.018	0.033 ± 0.009	0.059 ± 0.012	0.031 ± 0.029	0.038 ± 0.017

Table 4.2: 95% C.I. of Gradient of Exponentially Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.009 ± 0.002	0.017 ± 0.002	$0.003 \pm$ 5e-4	0.065 ± 0.007	$0.003 \pm$ 4e-4	0.040 ± 0.006
CAC	0.009 ± 0.004	0.004 ± 0.004	0.006 ± 0.002	0.062 ± 0.005	0.009 ± 0.003	0.016 ± 0.006
LR	0.011 ± 0.016	0.011 ± 0.018	0.008 ± 0.008	0.086 ± 0.037	0.014 ± 0.010	0.025 ± 0.039
WDC	0.015 ± 0.009	0.010 ± 0.006	0.013 ± 0.005	0.071 ± 0.009	0.007 ± 0.004	0.028 ± 0.014
WDNC	0.020 ± 0.017	0.012 ± 0.014	0.013 ± 0.007	0.059 ± 0.012	0.018 ± 0.007	0.031 ± 0.029

Equation (4.3), LR stands for the LR estimator of arc criticalities in Equation (4.5), WDC stands for the WD estimator of arc criticalities with common random numbers applied in Equation (4.6) and WDC is the estimator in Equation (4.6) without common random numbers applied.

In Tables 4.1 and 4.2, for each randomly generated AN with given number of nodes and arcs, an arc with arc criticality between 0.3 and 0.8 is randomly picked and applied with different arc criticality estimators under 200 simulation replications. The 95% C.I. for different arc criticality estimators are presented in Tables 4.1 and 4.2. It can be concluded that: for all activities' durations normally distributed or exponentially distributed, TAC, CAC, WDC and WDNC have very close accuracy, while LR has the worst accuracy.

Tables 4.3 and 4.4 are the aggregated version of Tables 4.1 and 4.2. Instead of focusing

Table 4.3: Averaged STD of Gradient of Normally Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.032	0.049	0.033	0.048	0.034	0.033
CAC	0.038	0.062	0.034	0.064	0.041	0.046
LR	1.024	1.580	1.086	1.369	0.971	1.097
WDC	0.038	0.061	0.041	0.059	0.041	0.041
WDNC	0.081	0.145	0.114	0.147	0.121	0.102

Table 4.4: Averaged STD of Gradient of Exponentially Distributed Arc Criticalities (based on 200 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	0.014	0.022	0.012	0.018	0.015	0.015
CAC	0.018	0.027	0.021	0.023	0.025	0.020
LR	0.104	0.173	0.083	0.125	0.108	0.113
WDC	0.042	0.063	0.037	0.054	0.042	0.040
WDNC	0.081	0.146	0.064	0.106	0.098	0.100

on estimating the gradient of one arc, Tables 4.3 and 4.4 contain the averaged STD among 30 different arcs from 3 different networks with the same number of nodes and arcs, and the sample variance performance of the TAC arc criticality gradient estimator is the lowest among the gradient estimators across randomly generated activity networks of different sizes. The variance of the TAC and CAC estimators are close, whereas LR has the highest variance. WDC is close to TAC and CAC when all arcs are normally distributed, but when all arcs are exponentially distributed, the sample variance performance of WDC is worse than that of TAC and CAC. The averaged STD value of WDNC is about twice as that of WDC, which indicates that common random numbers cuts the sample variance of the WD gradient estimator in half.

In Tables 4.5 to 4.6, MCT is short for the mean computation time, which is the corresponding

Table 4.5: MCT of Gradient of Normally Distributed Arc Criticalities

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	2.11	2.22	5.29	5.35	17.26	17.17
CAC	9.71	13.48	22.55	29.19	70.43	84.11
LR	0.41	2.79	2.52	11.38	10.61	69.78
WDC	0.82	5.51	4.32	22.28	21.20	140.24
WDNC	1.78	6.73	6.86	25.33	29.75	149.36

Table 4.6: MCT of Gradient of Exponentially Distributed Arc Criticalities

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TAC	2.10	2.24	5.31	5.32	17.22	17.30
CAC	10.03	13.75	23.59	30.01	74.37	88.48
LR	0.42	2.77	2.49	11.13	10.30	71.15
WDC	0.79	5.52	4.25	21.62	20.38	142.90
WDNC	1.72	6.65	6.69	24.41	28.63	151.75

average computing time for Tables 4.3 to 4.4. Tables 4.5 to 4.6 indicate that CAC has the longest computing time in most cases. WDC and WDNC have the longest computing time in the case of 100 nodes and 300 arcs. The performance function of LR, WDC and WDNC are both PCT, and the time complexity for calculating PCT is $\mathcal{O}(n_p)$, where n_p is the number of paths in the SAN. As a result, the time complexity of LR, WDC and WDNC are $\mathcal{O}(n_p)$. And the computing time of WDC and WDNC is about twice as LR, because weak derivative estimators in Equation (4.6) requires two estimates of the performance function Y.

4.2 Stochastic Gradient of First Moment of PCT

In this section, we are interested in estimating the following:

$$\frac{\partial \mathbb{E}(Y)}{\partial \theta_j}$$

where Y is the project completion time (PCT), which can be expressed as:

$$Y(X_1, ..., X_m) = \sum_{i \in P^*} X_i. \tag{4.7}$$

4.2.1 IPA Conditioned on Node Release Time

The IPA gradient estimator of $\frac{d\mathbb{E}(Y)}{d\theta_j}$ is given by:

$$I\{j \in P^*\} \frac{\partial X_j}{\partial \theta_j}.$$

Bowman [11] proposed a conditional IPA estimator for PCT conditioned on node release times, derived from the following:

$$\begin{aligned}
\frac{\partial \mathbb{E}(Y)}{\partial \theta_j} &= \mathbb{E}\Big[I\{j \in P^*\} \frac{dX_j}{d\theta_j} \Big] \\
&= \mathbb{E}\Big[\mathbb{E}\Big[I\{j \in P^*\} \frac{dX_j}{d\theta_j} \Big| T_1, ..., T_n \Big] \Big] \qquad (4.8) \\
&= \mathbb{E}\Big[C_a(i|T_1, ..., T_n) \frac{dX_j}{d\theta_j} \Big|_{X_j = T_{h(j)} - T_{t(j)}} \Big] \qquad (4.9)
\end{aligned}$$

where $\frac{dX_j}{d\theta_j}$ is the gradient of random variable X_j mentioned in Equation (3.4). And from Section 3.2.2 we have that $\frac{dX_j}{d\theta_j} = 1$ for θ_j being a location parameter and $\frac{dX_j}{d\theta_j} = \frac{X_j}{\theta_j}$ for θ_j being a scale parameter. Because the mean value μ is a location parameter for normally distributed random variables and a scale parameter for exponentially distributed random variables, Equation (4.9) reduces to $\frac{\partial \mathbb{E}(Y)}{\partial \theta_j} = C_a(i)$ for normally distributed arc j's length. In Equation (4.9), $C_a(i|T_1, ..., T_n)$ is the expression in Equation (2.6), $T_{h(j)}$ is the node release time of the head node (where the arrow points at) of arc j and $T_{t(j)}$ is the tail node of arc j. Thus, the IPA gradient estimator conditioned on node release times is named NIPA and is given by:

$$C_a(i|T_1, ..., T_n)\frac{dX_j}{d\theta_j}\Big|_{X_j=T_{h(j)}-T_{t(j)}}. \tag{4.10}$$

In Equation (4.10), $\frac{dX_j}{d\theta_j}\Big|_{X_j=T_{h(j)}-T_{t(j)}}$ stands for the value of the derivative of random variable X_j as a function of X_j when $X_j = T_{h(j)} - T_{t(j)}$. For example, when θ_j is a scale parameter of X_j, $\frac{dX_j}{d\theta_j}\Big|_{X_j=T_{h(j)}-T_{t(j)}} = \frac{T_{h(j)}-T_{t(j)}}{\theta_j}$.

4.2.2 IPA Conditioned on Threshold

We proposed a conditional IPA estimator for PCT conditioned on all arc lengths expect for the one of interest in [32]. The estimator is derived from the following:

$$\begin{aligned}\frac{\partial \mathbb{E}(Y)}{\partial \theta_j} &= \mathbb{E}\Big[I\{j \in P^*\}\frac{dX_j}{d\theta_j}\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[I\{j \in P^*\}\frac{dX_j}{d\theta_j}\Big| X_i, i \neq j\Big]\Big] \\ &= \begin{cases} \mathbb{E}[\Pr(X_j \geq m_j)], & \theta_j \text{ is a location parameter} \\ \mathbb{E}[\frac{1}{\theta_j}\int_{m_j}^{+\infty} x f_j(x)\, dx], & \theta_j \text{ is a scale parameter} \end{cases}\end{aligned} \tag{4.11}$$

The IPA estimator for PCT using TAC condition is named TIPA and is given by:

$$\begin{cases} \Pr(X_j \geq m_j), & \theta_j \text{ is a location parameter} \\ \frac{1}{\theta_j}\int_{m_j}^{+\infty} x f_j(x)\, dx, & \theta_j \text{ is a scale parameter} \end{cases} \tag{4.12}$$

$f_j(x)$ in Equations (4.11) and (4.12) is the density function of the duration of activity j, X_j. And for most of commonly used random variables, the integral part of Equation (4.12) is a closed-form function.

4.2.3 Higher-Order Stochastic Gradients of PCT

Since the IPA gradient estimator has better variance performance and faster convergence speed compared to the LR gradient estimator, we will not use LR for estimating the higher-order

stochastic gradient of PCT, that is estimating:

$$\frac{\partial^n \mathbb{E}(Y)}{\partial \theta_j^n}$$

where $n > 1$. Since estimating $\frac{\partial^n \mathbb{E}(Y)}{\partial \theta_j^n}$ using CIPA requests taking higher-order derivatives of Equation (2.6), which is very complicated, NIPA is not appropriate for estimating $\frac{\partial^n \mathbb{E}(Y)}{\partial \theta_j^n}$. Higher-order derivative using WD estimators require additional simulation replications, so is also not considered.

For estimating $\frac{\partial^n \mathbb{E}(Y)}{\partial \theta_j^n}$, the TIPA estimator for $\frac{\partial^n \mathbb{E}(Y)}{\partial \theta_j^n}$ is given by [32]:

$$\begin{cases} \frac{d^n \Pr(X_j \geq m_j)}{d\theta_j^n}, & \theta_j \text{ is a location parameter} \\ \frac{d^n \frac{1}{\theta_j} \int_{m_j}^{+\infty} x f_j(x)}{d\theta_j^n} dx, & \theta_j \text{ is a scale parameter} \end{cases} \tag{4.13}$$

Notice that for most commonly used continuous random variables, Equation (4.13) is a closed-form function of the threshold m_j defined in Equation (2.8).

4.3 Stochastic Gradient of Second Moment of PCT

In this section, we are interested in estimating the following:

$$\frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j}$$

where Y is the project completion time (PCT) given by Equation (4.7).

Once we have the estimator for $\frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j}$, the stochastic gradient of the variance of PCT with

respect to arc j's length distribution parameter θ_j can be calculated as:

$$\begin{aligned}\frac{\partial \text{Var}(Y)}{\partial \theta_j} &= \frac{\partial(\mathbb{E}(Y^2) - (\mathbb{E}(Y))^2)}{\partial \theta_j} \\ &= \frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j} - \frac{\partial(\mathbb{E}(Y))^2}{\partial \theta_j} \\ &= \frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j} - 2\mathbb{E}(Y)\frac{\partial \mathbb{E}(Y)}{\partial \theta_j}.\end{aligned}$$

Similarly, the higher-order derivatives are given by:

$$\begin{aligned}\frac{\partial^2 \text{Var}(Y)}{\partial \theta_j^2} &= \frac{\partial^2 \mathbb{E}(Y^2)}{\partial \theta_j^2} - 2\Big(\frac{\partial \mathbb{E}(Y)}{\partial \theta_j}\Big)^2 - 2\mathbb{E}(Y)\frac{\partial^2 \mathbb{E}(Y)}{\partial \theta_j^2} \\ \frac{\partial^3 \text{Var}(Y)}{\partial \theta_j^3} &= \frac{\partial^3 \mathbb{E}(Y^2)}{\partial \theta_j^3} - 6\frac{\partial \mathbb{E}(Y)}{\partial \theta_j}\frac{\partial^2 \mathbb{E}(Y)}{\partial \theta_j^2} - 2\mathbb{E}(Y)\frac{\partial^3 \mathbb{E}(Y)}{\partial \theta_j^3} \\ \frac{\partial^4 \text{Var}(Y)}{\partial \theta_j^4} &= \frac{\partial^4 \mathbb{E}(Y^2)}{\partial \theta_j^4} - 6\Big(\frac{\partial^2 \mathbb{E}(Y)}{\partial \theta_j^2}\Big)^2 - 8\frac{\partial \mathbb{E}(Y)}{\partial \theta_j}\frac{\partial^3 \mathbb{E}(Y)}{\partial \theta_j^3} - 2\mathbb{E}(Y)\frac{\partial^4 \mathbb{E}(Y)}{\partial \theta_j^4}.\end{aligned}$$

4.3.1 IPA Conditioned on Node Release Time

The IPA gradient estimator of $\frac{d\mathbb{E}(Y^2)}{d\theta_j}$ is given by:

$$2I\{j \in P^*\}Y\frac{\partial X_j}{\partial \theta_j}.$$

Bowman [11] proposed a conditional IPA estimator for PCT conditioned on node release times, the IPA gradient estimator of second moment of PCT conditioned on node release times is derived

from the following:

$$\begin{aligned}\frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j} &= \mathbb{E}\Big[2I\{j \in P^*\}Y\frac{dX_j}{d\theta_j}\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[2I\{j \in P^*\}Y\frac{dX_j}{d\theta_j}\Big|T_1,...,T_n\Big]\Big] \\ &= \mathbb{E}\Big[2C_a(i|T_1,...,T_n)T_n\frac{dX_j}{d\theta_j}\Big|_{X_j=T_{h(j)}-T_{t(j)}}\Big],\end{aligned} \tag{4.14}$$

where $\frac{dX_j}{d\theta_j}$ is the gradient of random variable X_j mentioned in Equation (3.4). Thus, the IPA gradient estimator for second moment of PCT conditioned on node release times is named NIPA2 and is given by:

$$2T_nC_a(i|T_1,...,T_n)\frac{dX_j}{d\theta_j}\Big|_{X_j=T_{h(j)}-T_{t(j)}}. \tag{4.15}$$

The difference between the gradient estimator of first moment of PCT in Equation (5.1) and that of second moment of PCT in Equation (4.15) is that Equation (4.15) has one more term $2T_n$. Assume that the regularity condition for exchanging derivative and expectation n times is satisfied in Equation (5.1), then the higher order IPA gradient estimator for $\frac{\partial^n \mathbb{E}(Y^2)}{\partial \theta_j^n}$ is given by taking derivative of Equation (4.15) $n-1$ times, which requires $n-1$ order derivative of the recursive formula $C_a(i|T_1,...,T_n)$, that is very complicated and time consuming. In conclusion, the NIPA2 estimator is not suitable for higher-order gradient estimation of the second moment of PCT.

4.3.2 IPA Conditioned on Threshold

The IPA estimator for second moment of PCT conditioned on all arc lengths expect for the one of interest is similar to that of the first-moment estimator. The estimator is derived from the

following:

$$\begin{aligned}\frac{\partial \mathbb{E}(Y^2)}{\partial \theta_j} &= \mathbb{E}\Big[2I\{j \in P^*\}Y\frac{dX_j}{d\theta_j}\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[2I\{j \in P^*\}Y\frac{dX_j}{d\theta_j}\Big|X_i, i \neq j\Big]\Big] \\ &= \begin{cases} 2\mathbb{E}[\int_{m_j}^{+\infty}(x+a_j)f_j(x)\,dx], & \theta_j \text{ is a location parameter} \\ 2\mathbb{E}[\frac{1}{\theta_j}\int_{m_j}^{+\infty}x(x+a_j)f_j(x)\,dx], & \theta_j \text{ is a scale parameter} \end{cases}\end{aligned}$$

where $a_j = \|\mathcal{P}\|^j$, which is the length of the critical path given all arc lengths under the condition that the length of arc j is 0, and $f_j(x)$ is the density function of the duration of activity j, X_j. The IPA estimator for the second moment of PCT using TAC condition is named TIPA2 and is given by:

$$\begin{cases} 2\int_{m_j}^{+\infty}(x+a_j)f_j(x)\,dx, & \theta_j \text{ is a location parameter,} \\ \frac{2}{\theta_j}\int_{m_j}^{+\infty}x(x+a_j)f_j(x)\,dx, & \theta_j \text{ is a scale parameter.} \end{cases} \tag{4.16}$$

Assume the estimator in Equation (4.16) is $n-1$ times continuous differentiable. Then the IPA gradient estimator conditioned on TAC condition for $\frac{\partial^n \mathbb{E}(Y^2)}{\partial \theta_j^n}$ is given by:

$$\begin{cases} 2\int_{m_j}^{+\infty}(x+a_j)\frac{\partial^{n-1}f_j(x)}{\partial \theta_j^{n-1}}\,dx, & \theta_j \text{ is a location parameter,} \\ \frac{2}{\theta_j}\int_{m_j}^{+\infty}x(x+a_j)\frac{\partial^{n-1}f_j(x)}{\partial \theta_j^{n-1}}\,dx, & \theta_j \text{ is a scale parameter.} \end{cases} \tag{4.17}$$

Notice that for most commonly used continuous random variables, Equation (4.17) is a closed-form function of the threshold m_j defined in Equation (2.8).

Chapter 5: Applications of Gradient Estimators in SANs

In this chapter, we first introduce how the gradient estimator in Section 4.2.3 can be applied in estimating the change in expected PCT when the mean duration of one activity is changed by a given amount. Then, using the estimation of change of expected PCT, an algorithm is proposed for solving an optimization problem in SANs called time-cost tradeoffs problem.

Bowman [11] proposed a simulation-based optimization problem of crashing the mean duration of activities called the time-cost trade-off problem. Kim et al. [4] concentrate on the problem of minimizing a quantile of the PCT subject to constraints on circuit design variables. Goh [33] proposed a robust programming algorithm for minimizing the expected PCT when the distributions of the activity durations are not known. We improved the heuristic optimization algorithm in [11] using the SPA estimator of expected PCT. The content in this chapter is based on material from [32].

5.1 Estimating the Change of Expected Project Completion Time

Cho and Yum [34] claimed that $\frac{\partial \mathbb{E}(Y)}{\partial \mu_i}$ cannot serve as an accurate estimate for change in $\mathbb{E}(Y)$ due to a discrete change in the mean duration of activity i, μ_i. For example, if the original value of activity 1's mean duration μ_1 is 25, then using $\frac{\partial \mathbb{E}(Y)}{\partial \mu_1}$ to estimate the amount of change of $\mathbb{E}(Y)$ when μ_1 is decreased by 5, keeping all the other activity mean durations unchanged, is

not accurate.

The goal is to estimate:

$$\Delta_i \mathbb{E}[Y](\delta) = \mathbb{E}[Y]\Big|_{\mu_i=\mu} - \mathbb{E}[Y]\Big|_{\mu_i=\mu-\delta}, \text{ for } \delta \geq 0.$$

Cho and Yum [34] consider:

$$\Delta_i \mathbb{E}[Y](\delta) = \int_{\mu-\delta}^{\mu} C_i(x)dx, \tag{5.1}$$

where $C_i(x)$ is a function with input of arc i's mean duration and output of criticality index of arc i. In [34], it is assumed that all activities are normally distributed and the parameter of interest is the mean duration of activity i. An explanation of Equation (5.1) is given by:

$$\begin{aligned}\Delta_i \mathbb{E}[Y](\delta) &= \mathbb{E}[Y]\Big|_{\mu_i=\mu} - \mathbb{E}[Y]\Big|_{\mu_i=\mu-\delta} \\ &= \int_{\mu-\delta}^{\mu} \frac{\partial \mathbb{E}[Y]}{\partial \mu_i} dx = \int_{\mu-\delta}^{\mu} C_i(x)dx. \end{aligned} \tag{5.2}$$

Equation (4.15) follows from the mean duration μ_i being a location parameter for normal distributions, and Bowman [11] proved that for location parameters, $\frac{\partial \mathbb{E}[Y]}{\partial \mu_i} = C_i(x)$. Then, to find a functional approximation of $C_i(x)$, Cho and Yum [34] proposed using logistic regression to fit $C_i(x)$, because $C_i(x)$ is a S shaped curve. Numerical experiments indicated that their logit fitting approach underestimates $\Delta_i \mathbb{E}[Y](\delta)$ when σ_i/μ_i is large, where μ_i and σ_i are the mean and standard deviation of normally distributed X_i.

We propose using a Taylor series approximation to fit $C_i(x)$ locally, assuming $C_i(x)$ is

N-times continuously differentiable,

$$C_i(x) \approx \sum_{k=0}^{N} \frac{1}{k!} C_i^{(k)}(\mu)(x-\mu)^k, \tag{5.3}$$

where $C_i^{(k)}(\mu)$ is the k^{th} order derivative of function $C_i(x)$ at activity i's original mean duration μ, which can be estimated using the IPA estimator in Equation (4.2). As for the case θ_i is a scale parameter rather than a location parameter, we have the expectation version of Equation (5.3) given by:

$$\frac{\partial \mathbb{E}[Y]}{\partial \mu_i} \approx \sum_{k=0}^{N} \frac{1}{k!} \frac{\partial^{k+1} \mathbb{E}[Y]}{\partial \mu_i^{k+1}}(\mu)(x-\mu)^k. \tag{5.4}$$

Using Equation (5.4), we have that Equation (5.2) becomes:

$$\begin{aligned}
\Delta_i \mathbb{E}[Y](\delta) &= \mathbb{E}[Y]\Big|_{\mu_i=\mu} - \mathbb{E}[Y]\Big|_{\mu_i=\mu-\delta} \\
&= \int_{\mu-\delta}^{\mu} \frac{\partial \mathbb{E}[Y]}{\partial \mu_i} dx \\
&\approx \int_{\mu-\delta}^{\mu} \sum_{k=0}^{N} \frac{1}{k!} \frac{\partial^{k+1} \mathbb{E}[Y]}{\partial \mu_i^{k+1}}(\mu)(x-\mu)^k dx \\
&= \sum_{k=0}^{N} \frac{1}{(k+1)!} \frac{\partial^{k+1} \mathbb{E}[Y]}{\partial \mu_i^{k+1}}(\mu)(x-\mu)^{k+1}\Big|_{\mu-\delta}^{\mu} \\
&= \sum_{k=0}^{N} \frac{(-1)^{k+1}}{(k+1)!} \frac{\partial^{k+1} \mathbb{E}[Y]}{\partial \mu_i^{k+1}}(\mu)\delta^{k+1}.
\end{aligned} \tag{5.5}$$

Notice that Equation (5.5) works not only for the distribution parameter being the mean, but also works for other distributional parameters. In [34], they also claimed that using direct Monte Carlo simulation (MCS) to fit a logistic regression requires several thousands of runs to estimate $C_i(x)$ at different x values. Hence, they proposed using Taguchi Orthogonal Array experiment to

reduce the number of simulation replications. Using the TAC estimator, we can solve this issue with an easy and efficient approach. Since m_i in Equation (2.8) does not depend on X_i, and therefore not on μ_i, no new simulation replications are needed for estimating $C_i(x)$ at a new x value.

5.2 Estimating the Criticality Curve

As mentioned in Section 5.1, to fit a logistic regression model for $C_i(x)$, we need to estimate the $C_a(i)$ value when the mean duration of activity i takes different discrete values while the parameters of other arc length remain unchanged. For estimating arc criticality using the TAC estimator, calculating the threshold m_i is important. The following algorithm presents an efficient way of calculating the TAC threshold for one activity of interest.

When estimating the criticality curve using TAC, Algorithm 1 is helpful. Suppose we are interested in estimating the criticality curve at 10 different values. The following procedure explains the advantage of the TAC estimator:

1. Simulate all arc lengths.
2. Calculate threshold m_i.
3. Calculate estimator for $C_i(x)$, $EST \leftarrow \bar{F}_i(m_i)$.

When estimating $C_i(x)$ at different x values, steps 1 and 2 only need to be run once, and for different x values, redo step 3 with the distribution parameter of $F_i(x)$ changed. This property is called Sample Performance Invariance (SPI), because estimating the gradient at different parameter values does not depend on the simulated samples. This property is advantageous for all measure-

based gradient estimation methods, such as the Likelihood Ratio (LR) method and the weak derivatives method. Sample path-based gradient estimation methods such as IPA do not generally possess this property, but for the TAC estimator, the IPA estimator also possesses the SPI property, so we can estimate the function of $C_i(x)$ more efficiently.

5.3 Optimization of Time-cost tradeoff Problem

Bowman [11] formulated a nonlinear programming problem called the time-cost tradeoff optimization problem, where the objective function is the expected PCT of a stochastic activity network. The cost of reducing one unit of mean duration of activity i is b_i, and the total budget is B. The upper and lower bounds of the mean duration of activity i are given by u_i and l_i, i.e., $l_i \leq \mu_i \leq u_i$. Assume $\sum_i b_i(u_i - l_i) > B$ and there are m arcs in the activity network. Then we have an optimization problem with nonlinear objective function and linear constraints given by:

$$\min_{\beta} \quad \mathbb{E}[Y] \tag{5.6}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} b_i \beta_i \leq B \tag{5.6a}$$

$$\beta_i \leq u_i - l_i, \quad \forall i \in \{1, ..., m\} \tag{5.6b}$$

$$-\beta_i \leq 0, \quad \forall i \in \{1, ..., m\} \tag{5.6c}$$

where β_i is the amount of decreasing of θ_i (distribution parameter of the duration of activity i, usually is the mean) and Y is the PCT. In [11], Bowman claimed that the KKT condition [29] for Problem (5.6) is a necessary and sufficient condition for a local optimum solution and derived a heuristic algorithm for finding the local optimum solution satisfying the local KKT

condition. Here, we will show that the KKT condition of Problem (5.6) is indeed a necessary and sufficient condition for a global optimum solution by proving that the expected PCT as the objective function in Problem (5.6) is a convex function of β. For proving the convexity of the objective function in in Problem (5.6), the following Theorem is presented:

Theorem 1. *Let $X = (X_1, X_2, ..., X_m)$ be a random vector with X_i independent random variables with known invertible distribution functions $F_i(\cdot; \theta_i)$, $f : \mathbb{R}^m \to \mathbb{R}$ be a convex function, A a real matrix and b a real vector, $\theta = (\theta_1, \theta_2, ..., \theta_m)$, where θ_i is the distribution parameter of X_i. If θ_i is either a location or scale parameter of X_i for all i, then $\mathbb{E}[f(AX + b)]$ is a convex function of θ.*

Proof. Proof of Theorem 1. First consider location parameter θ_i of X_i, where i is fixed. The distribution function of X_i is given by $F_i(x; \theta_i)$, for x_0 fixed, by definition of location parameter θ_i, we have, $F_i(x_0 + \theta'; \theta') = F_i(x_0 + \theta''; \theta'') = y$, where $\theta' \neq \theta''$ are two different values of θ_i. Then we have $F_i^{-1}(y; \theta') = x_0 + \theta'$ and $F_i^{-1}(y; \theta'') = x_0 + \theta''$. Hence, $F_i^{-1}(y; \theta'') = F_i^{-1}(y; \theta') - \theta' + \theta''$.

By the probability integral transform theorem [35], we have $X_i(\theta_i) \overset{d}{=} F_i^{-1}(Y_i; \theta_i)$, where $Y_i \overset{\text{i.i.d}}{\sim} U(0, 1)$, let $\theta' = \theta_0$ fixed, for any value of θ_i, we have $X_i(\theta_i) \overset{d}{=} Z_i(\theta_i) = F_i^{-1}(Y_i; \theta_i) = F_i^{-1}(Y_i; \theta_0) - \theta_0 + \theta_i$, then $X_i(\theta_i)$ is equal in distribution to a new random variable $Z_i(\theta_i)$ that can be expressed as a linear function of θ_i.

Similarly, if θ_i is a scale parameter of X_i, by definition of a scale parameter, we have $F_i(x_0\theta'; \theta') = F_i(x_0\theta''; \theta'') = y$, and $F_i^{-1}(y; \theta') = x_0\theta'$, $F_i^{-1}(y; \theta'') = x_0\theta''$. Hence, $F_i^{-1}(y; \theta'') = \frac{F_i^{-1}(y;\theta')}{\theta'}\theta''$. Therefore, $X_i(\theta_i) \overset{d}{=} Z_i(\theta_i) = \frac{F_i^{-1}(Y_i;\theta_0)}{\theta_0}\theta_i$, where θ_0 is a constant and $Y_i \overset{\text{i.i.d}}{\sim} U(0, 1)$. Hence, we have that $X_i(\theta_i)$ is equal in distribution to a new random variable $Z_i(\theta_i)$ that can be

expressed as a linear function of θ_i.

In conclusion, if θ_i is either a location or scale parameter of X_i for all i, then $X = (X_1, X_2, ..., X_m) \stackrel{d}{=} Z = (Z_1, Z_2, ..., Z_m)$, where Z is a linear mapping of θ. Since the composition of convex and linear mappings is convex, and expectation preserves convexity [29], $\mathbb{E}_X[f(AX + b)] = \mathbb{E}_Z[f(AZ + b)]$ is a convex function of θ.

□

Bowman [11] states that constraint (5.6a) can be replaced by equality constraint $\sum_{i=1}^{m} b_i\beta_i = B$, because $\mathbb{E}[Y]$ is a decreasing function of β_i for all i. Hence, we replace the constraint (5.6a) with $\sum_{i=1}^{m} b_i\beta_i = B$ and derive the KKT conditions as follows:

$$\begin{aligned}
\sum_{i=1}^{m} b_i\beta_i &= B \\
\beta_i &\geq 0, \quad \forall i \in \{1, 2, ..., m\} \\
\lambda_i^b &\geq 0, \quad \forall i \in \{1, 2, ..., m\} \\
\lambda_i^c &\geq 0, \quad \forall i \in \{1, 2, ..., m\} \\
\lambda_i^b(\beta_i - (u_i - l_i)) &= 0, \quad \forall i \in \{1, 2, ..., m\} \\
\lambda_i^c\beta_i &= 0, \quad \forall i \in \{1, 2, ..., m\} \\
\nabla\mathbb{E}[Y] + \nu\nabla(\sum_{i=1}^{m} b_i\beta_i - B) &= 0
\end{aligned}$$

From the KKT conditions, we have that the necessary and sufficient condition for β to be the

global minimum solution of Problem (5.6) is given by:

$$\sum_{i=1}^{m} b_i \beta_i = B$$

$$\beta_i \geq 0, \quad \forall i \in \{1, 2, ..., m\}$$

$$\frac{\partial \mathbb{E}[Y]}{\partial \beta_i} = \frac{\partial \mathbb{E}[Y]}{\partial \beta_j}, \quad \forall i, j \in \{k | 0 < \beta_k < u_k - l_k, k = 1, 2, ..., m\} \tag{5.7}$$

The heuristic algorithm proposed in [11] has two stages: in stage 1, at each step, the algorithm decreases the mean duration time of the activity that has the largest $\frac{\partial \mathbb{E}[T]}{\partial \beta_i}/b_i$ by aB/b_i, where a is the fraction of budget to be used, e.g., $a = 0.01$; in stage 2, the algorithm redistributes the budget between activity i that has the largest $\frac{\partial \mathbb{E}[T]}{\partial \beta_i}/b_i$ value and activity j that has the smallest $\frac{\partial \mathbb{E}[T]}{\partial \beta_i}/b_i$ value among activities whose decision variables are not at their boundaries (strictly between 0 and upperbound).

Using the proposed Taylor series approximation method in Equation (5.5), we propose a new heuristic algorithm called the Knapsack Ratio (KR) algorithm [32] for solving the time-cost tradeoff optimization problem. In the following algorithm, θ_i^0 stands for the original mean duration of activity i. The KR algorithm is given below:

In Algorithm 3, N is the number of simulation replications, α and t are real numbers strictly between 0 and 1, θ is the vector of distribution parameters of all activities' duration in the AN. $C_a(i)$ is the criticality index of activity i and is estimated using the TAC estimator in Equation (2.8). $\Delta_i \mathbb{E}(Y)(\delta_i)$ is defined and estimated using the Taylor series approximation method in Equation (5.5). In Algorithm 3, we assumes that at each step, a chosen activity's mean duration will be decreased by a fixed amount. The optimal solution of the optimization problem

Algorithm 3: Knapsack Ratio (KR) Algorithm for Time-cost tradeoffs Problem

Input : N, α, t, θ
Output: θ

$\mathbf{G} \leftarrow \{i | C_a(i) \geq t, i \in \{1, 2, ..., n_a\}\}$
$\theta^0 \leftarrow \theta$
while $B > 0$ **do**
  $\delta_i \leftarrow \alpha * (\theta_i^0 - l_i)$
  $E \leftarrow \{i | l_i < \theta_i < u_i, i \in \mathbf{G}\}$
  $j \leftarrow \underset{i \in E}{\text{argmax}} \{\frac{\Delta_i \mathbb{E}(Y)(\delta_i)}{\delta_i} / b_i\}$
  $\theta_i \leftarrow \theta_i - \min(b_i \delta_i, B)/b_i$
  $B \leftarrow B - \min(b_i \delta_i, B)$
end

is approximated by the optimal solution of a knapsack integer programming problem when α is small enough, e.g., $\alpha = 0.01$.

5.4 Numerical Experiments Results

In this section, stochastic activity networks with fixed numbers of nodes and arcs are randomly generated using the algorithm presented in [5]. All arc lengths in the randomly generated network are either normally distributed or exponentially distributed. For normally distributed activities, their mean durations are generated uniformly between 0.5 and 50, and their standard deviations are generated uniformly between 0.1 and 1 times their mean durations. For exponentially distributed activities, their mean durations are generated uniformly between 0.5 and 50.

For following experiments, we will use direct Monte Carlo simulation with common random numbers (DMCCR). Suppose we use Monte Carlo simulation with N simulation replications to estimate $\mathbb{E}(Y)$ or $C_a(i)$ when a stochastic network and all activities' distribution parameters are given. Then one of the activity's distribution parameter θ_i is changed and we need to re-estimate performance functions like $\mathbb{E}(Y)$ and $C_a(i)$. Instead of simulating all activities' lengths again,

DMCCR only re-simulates activity i's length N times and replaces the old simulated X_i values while keeping all other simulated $X_j, j \neq i$, values unchanged. DMCCR can save computational time and reduce variance when only a small set of activities' parameters are changed.

5.4.1 Numerical Results of Estimation of Criticality Curve

With the number of arcs and number of nodes fixed, a random network with all arcs independently distributed is generated. The criticality index of a given activity i is mainly affected by two factors: the number of paths that includes activity i and the mean duration of activity i. In extreme cases, if activity i is on all paths, then $C_i(x) = 1, \forall x$, and if the mean duration of activity i is sufficiently large, then $C_a(i)$ is very close to 1. For activities with very small criticality index values, we are less interested in fitting their criticality curves, because decreasing their mean duration has negligible effect on the expected project completion time. For activities that are on most of the paths, we are also less interested in fitting their criticality curves, because changes in their mean duration has negligible effect on their criticality indexes. Therefore, for each SAN, we randomly choose an activity i such that $C_a(i) > 0.5$ and $\mathcal{R}_i < 0.6$.

After the network structure and distribution parameters are randomly generated, and the arc of interest is chosen, for a chosen arc i with original mean duration μ, 30 mean duration values ranging from 0.1μ to 1.5μ with stepsize $1.4\mu/30$ are considered. For each mean duration value x, $N = 1000$ simulation replications are run for estimating $C_i(x)$ and its corresponding sample standard deviation is computed. The mean of the sample standard deviations across the 30 different x values is also calculated, called the Mean Standard Deviation (MSD). To compare the TAC and CAC estimators [2], ratios of their MSD and computation time are computed.

Algorithm 1 is used for calculating the TAC estimator, and DMCCR is applied for estimating the CAC estimator. From Table 5.1, we can see that the variance performance of the TAC estimator and the CAC estimator in estimating the criticality curve for both normally distributed and exponentially distributed activity times are close to each other. From Table 5.2, we can see that the computational time for the CAC estimator is about 100 times that of using the TAC estimator. Thus, considering variance performance and computing speed, the TAC estimator is preferred over the CAC estimator in estimating the criticality curve.

Table 5.1: Criticality Curve Estimation MSD Ratio of CAC/TAC (unit 10^{-3})

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
Arc Distributions	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
Normal	9/12	10/11	12/14	10/13	12/11	12/13
Exponential	10/13	11/14	12/13	9/13	11/13	10/11

Table 5.2: Criticality Curve Estimation Time Ratio of CAC/TAC

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
Arc Distributions	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
Normal	73.8	93.1	95.9	123.7	108.0	126.7
Exponential	88.2	102.6	85.9	96.9	86.2	90.1

Figure 5.1 depicts a representative criticality curve plot of a chosen activity i in a randomly generated SAN with normally distributed activities of size 20 nodes and 50 arcs, where 30 different criticality values are obtained using the same method as in the experiments in Tables 5.1 - 5.2. For logit curve fitting, we first do a logistic regression on 30 sample points estimated by the TAC estimator and have the estimation for coefficients of the Logit function, then plot the curve of the Logit function. For TAC and CAC, we first estimate 30 different criticality values,

then join the 30 points with a smooth curve. Figure 5.1 indicates that the Logit curve fit of the criticality curve deviates from the other two, especially for lower values of μ_i.

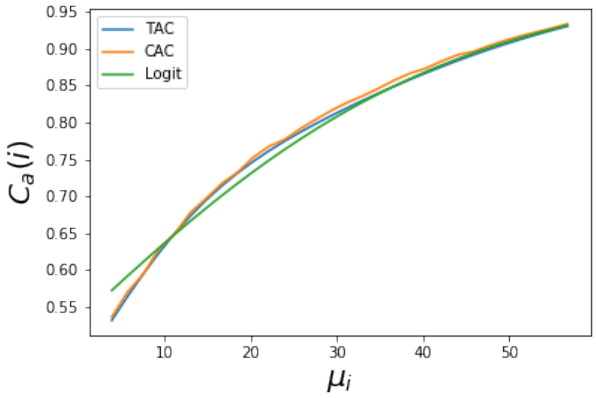


Figure 5.1: Criticality Curve of Normally Distributed SAN with 20 Nodes and 50 Arcs.

5.4.2 Estimation of Change of Mean Completion Time

We consider three approaches for estimating $\Delta_i \mathbb{E}[Y](\delta)$: direct Monte Carlo simulation with common random numbers (DMCCR); Logit model approximation (LGT); Taylor series approximation (TSA). The experiment first generates a random network with given numbers of arcs and nodes, and then chooses an activity as in Section 6.1, i.e., an activity i such that $C_a(i) > 0.5$ and $\mathcal{R}_i < 0.6$. In the following experiments, $\delta = \alpha\mu$, where μ is the original mean duration of the activity of interest and α takes two values: $\alpha = 10\%$ and $\alpha = 20\%$. For the TSA method, $N = 3$ in Equation (5.5). For each method, $N = 1000$ simulation replications are run for estimating $\Delta_i \mathbb{E}[Y](\delta)$ once. For each methods and network, $\Delta_i \mathbb{E}[T](\delta)$ is estimated 100 times to compute the sample mean and sample standard error.

In Tables 5.3 - 5.6, the first three rows are the 95% confidence intervals (C.I.) of the three methods on the 100 macro replications. And the last three rows are the total computation time of the three methods. For each column, an activity network with given number of nodes, number

of arcs, and arc distributions is first generated. From Tables 5.3 - 5.6 we can conclude that both TSA and LGT method have better variance performance than the DMCCR method. The variance performance of TSA and LGT are indistinguishable. In Tables 5.3 - 5.4, all three methods have close estimated mean values. But in Tables 5.5 - 5.6, LGT underestimates $\Delta_i \mathbb{E}[Y](\delta)$, as is mentioned in [34]. As for computing speed, TSA and DMCCR are faster than LGT, and TSA and DMCCR have similar computing speed. In conclusion, in terms of variance performance and computing speed, the TSA method is the best among all three different methods.

Table 5.3: Normal Distribution Mean Completion Time with 10% Change (95% C.I. based on 100 Macro Replications, STD in parentheses)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TSA C.I.	2.22 $\pm$ 0.006	3.85 $\pm$ 0.006	4.15 $\pm$ 0.004	3.39 $\pm$ 0.004	4.77 $\pm$ 0.004	3.52 $\pm$ 0.006
LGT C.I.	2.22 $\pm$ 0.006	3.86 $\pm$ 0.006	4.15 $\pm$ 0.004	3.4 $\pm$ 0.004	4.78 $\pm$ 0.004	3.52 $\pm$ 0.006
DMCCR C.I.	2.19 $\pm$ 0.19	3.91 $\pm$ 0.22	4.25 $\pm$ 0.18	3.35 $\pm$ 0.27	4.82 $\pm$ 0.16	3.68 $\pm$ 0.24
TSA Time	1.32 (0.06)	1.33 (0.07)	2.87 (0.18)	2.87 (0.13)	9.21 (0.33)	9.30 (0.39)
LGT Time	3.20 (0.13)	3.20 (0.14)	4.69 (0.22)	4.71 (0.23)	10.87 (0.35)	10.97 (0.41)
DMCCR Time	1.21 (0.06)	1.22 (0.07)	2.76 (0.17)	2.74 (0.14)	9.06 (0.33)	9.09 (0.37)

Table 5.4: Normal Distribution Mean Completion Time with 20% Change (95% C.I. based on 100 Macro Replications, STD in parentheses)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TSA C.I.	4.25 $\pm$ 0.01	7.6 $\pm$ 0.014	8.27 $\pm$ 0.01	6.74 $\pm$ 0.006	9.5 $\pm$ 0.008	6.98 $\pm$ 0.01
LGT C.I.	4.24 $\pm$ 0.01	7.63 $\pm$ 0.014	8.28 $\pm$ 0.01	6.75 $\pm$ 0.006	9.52 $\pm$ 0.008	6.99 $\pm$ 0.01
DMCCR C.I.	4.36 $\pm$ 0.19	7.78 $\pm$ 0.24	8.31 $\pm$ 0.18	6.77 $\pm$ 0.31	9.5 $\pm$ 0.16	7.11 $\pm$ 0.19
TSA Time	1.27 (0.04)	1.25 (0.05)	2.76 (0.21)	2.80 (0.11)	9.00 (0.51)	9.04 (0.66)
LGT Time	3.04 (0.07)	3.02 (0.11)	4.51 (0.29)	4.61 (0.25)	10.68 (0.65)	10.73 (0.80)
DMCCR Time	1.15 (0.02)	1.15 (0.05)	2.66 (0.19)	2.68 (0.13)	8.89 (0.52)	8.93 (0.68)

Table 5.5: Exponential Distribution Mean Completion Time with 10% Change (95% C.I. based on 100 Macro Replications, STD in parentheses)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TSA C.I.	2.42 ± 0.008	3.73 ± 0.008	4.05 ± 0.006	2.88 ± 0.008	4.63 ± 0.006	3.0 ± 0.004
LGT C.I.	1.28 ± 0.008	3.19 ± 0.01	3.81 ± 0.008	2.41 ± 0.008	4.3 ± 0.008	2.46 ± 0.01
DMCCR C.I.	2.6 ± 0.26	3.66 ± 0.34	3.95 ± 0.36	2.96 ± 0.29	4.34 ± 0.34	3.09 ± 0.27
TSA Time	1.19 (0.08)	1.21 (0.12)	2.73 (0.22)	2.76 (0.15)	9.08 (0.74)	9.15 (0.62)
LGT Time	1.26 (0.09)	1.27 (0.13)	2.80 (0.24)	2.83 (0.13)	9.14 (0.68)	9.18 (0.62)
DMCCR Time	1.17 (0.08)	1.19 (0.13)	2.71 (0.22)	2.73 (0.13)	9.05 (0.68)	9.16 (0.63)

Table 5.6: Exponential Distribution Mean Completion Time with 20% Change (95% C.I. based on 100 Macro Replications, STD in parentheses)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
TSA C.I.	4.68 ± 0.018	7.4 ± 0.018	8.07 ± 0.012	5.71 ± 0.014	9.21 ± 0.012	5.91 ± 0.016
LGT C.I.	2.42 ± 0.018	6.33 ± 0.024	7.6 ± 0.018	4.79 ± 0.018	8.58 ± 0.016	4.85 ± 0.02
DMCCR C.I.	4.75 ± 0.23	7.31 ± 0.31	8.24 ± 0.35	5.58 ± 0.25	9.15 ± 0.38	5.78 ± 0.27
TSA Time	1.16 (0.02)	1.16 (0.10)	2.64 (0.15)	2.68 (0.18)	8.87 (0.65)	8.87 (0.15)
LGT Time	1.23 (0.02)	1.23 (0.09)	2.70 (0.16)	2.75 (0.18)	8.95 (0.66)	8.95 (0.15)
DMCCR Time	1.14 (0.01)	1.15 (0.09)	2.63 (0.15)	2.66 (0.18)	8.85 (0.66)	8.87 (0.13)

5.4.3 Numerical Experiments of Optimization of Time-Cost Tradeoffs

In this section, the DMCCR is not applied, i.e., whenever μ_i is changed, all arc lengths are re-simulated. The DMCCR is not applied here for two reasons: (1) the number of simulation replications $N = 1000$ is too small to get an accurate estimate using DMCCR for the case when several μ_is are changed one by one; (2) for large enough N, e.g., $N = 10000$, the time complexity of evaluating the ratio is relatively large compared to that of extra simulation runs without using the DMCCR. For each randomly generated SAN, the KR algorithm and Bowman's algorithm [11] are compared for finding the optimal solution of time-cost tradeoff problems described in

Problem (5.6). The parameter settings are: μ_i^0 is the original mean duration of activity i, costs a_i are uniformly generated between 1 and 20, $u_i = \mu_i^0, l_i = 0.2 * \mu_i^0$, criticality lower bound $t = 0.001$, budget $B = 0.2 \sum a_i(u_i - l_i)$. The parameter settings for the KR algorithm are: number of simulation replications $N = 1000$, decreasing step $\alpha = 0.2$, and criticality lower bound $t = 0.001$, Taylor series approach is used in step 6 of the KR algorithm with $N = 2$ in Equation (5.5). The parameter setting for Bowman's algorithm are: simulation replication for phase 1 and phase 2 are $N1 = N2 = 1000$, step of decreasing for phase 1 and phase 2 are $A1 = A2 = 0.1$*Budget, upper limit of iteration times for phase 2 is $M = 20$. After finding the solutions using the two different algorithms, 10,000 simulation replications are run for estimating the mean and standard deviation of project completion time under the parameter settings obtained through two algorithms. In Tables 5.7 - 5.8, the first two rows are the 95% confidence interval (C.I.) of project completion time under the optimized parameters obtained by the KR algorithm and Bowman's algorithm. The third row is the 95% C.I. for the project completion time before decreasing any mean duration. Rows 4 and 5 are the computation times for the optimization using the two methods.

Table 5.7: Normal Distribution Optimal Project Completion Time Estimation (95% C.I. based on 10,000 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
KR	181.7 $\pm$ 0.74	239 $\pm$ 0.78	211.1 $\pm$ 0.8	232.3 $\pm$ 0.71	237.8 $\pm$ 0.74	253.4 $\pm$ 0.74
Bowman	183.1 $\pm$ 0.74	239.5 $\pm$ 0.78	210.7 $\pm$ 0.82	233 $\pm$ 0.68	238.1 $\pm$ 0.76	257 $\pm$ 0.74
Original	301.7 $\pm$ 0.84	391.7 $\pm$ 0.92	409.3 $\pm$ 1.02	405.9 $\pm$ 0.91	427.2 $\pm$ 0.98	474.8 $\pm$ 0.84
KR Time	65	199	127	371	692	2806
Bowman Time	128	286	543	795	1949	3942

Table 5.8: Exponential Distribution Optimal Project Completion Time Estimation (95% C.I. based on 10,000 Independent Replications)

	Network Size					
	30 Nodes		50 Nodes		100 Nodes	
	60 Arcs	90 Arcs	100 Arcs	150 Arcs	200 Arcs	300 Arcs
KR	209.8 ± 1.08	278.3 ± 1.32	235.9 ± 1.19	291.1 ± 1.25	287.8 ± 1.2	318.9 ± 1.08
Bowman	208.6 ± 1.07	275.8 ± 1.29	232.3 ± 1.13	291.4 ± 1.22	288.4 ± 1.18	320.1 ± 1.07
Original	347.3 ± 1.7	460.8 ± 2.03	458.8 ± 2.02	499.9 ± 2.09	500.6 ± 1.95	595.9 ± 1.7
KR Time	22	177	82	248	567	2760
Bowman Time	70	209	315	450	2549	5148

From Tables 5.7 - 5.8, we can see that the KR algorithm is better than Bowman's algorithm in terms of computing speed, especially for large complex networks. However, in some cases, the KR algorithm's solution has a worse (larger) objective value compared with Bowman's algorithm's solution, although in most cases the difference is relatively small compared to the original objective function value. By decreasing α and t, we believe that the KR algorithm will converge to the global optimal solution.

5.5 Future Research

Future research will focus on the local functional estimation of the variance of PCT using the higher-order gradient estimator in Section 4.3. The higher-order gradient estimator in Section 4.3 and its application in estimating the change of variance of PCT due to large perturbation of distribution parameters in the SAN will be tested on randomly generated ANs. By changing the objective function of the time-cost tradeoffs problem to a linear combination of expected PCT and variance of PCT, we can decrease uncertainty of the PCT by choosing the optimal decision variables.

The PERT distribution [36] has been widely used in project management. Sensitivity

analysis and time-cost tradeoffs optimization problems will be tested on SANs with activity durations PERT distributed.

Theorem 1 only consider the case where the distribution parameters in SANs are either location or scale parameters. Future work will focus on expanding Theorem 1 to the case that the parameter of interest is neither a location nor a scale parameter, for example, characterizing tail behavior in heavy-tailed distributions.

Chapter 6: Optimal Crashing AN with Single Disruption

6.1 Problem Background

6.1.1 Crashing ANs

Crashing an activity means reducing the duration of the activity at some cost. Crashing an activity network means reducing the PCT by crashing some activities in the network. Generally, there are multiple crashing options. For example, let activity 1 be drilling a tunnel that has a fixed duration of 1000 hours, and suppose there are two options for crashing activity 1: option 1 is to have overtime work to reduce the duration by 0.05 percent for every extra working hour; and option 2 is to use new materials that can reduce the duration by 1 percent for every ton of new material used. The cost of overtime work is $\$5K/hr$ and the cost of new material is $\$200K/ton$. Assume a limit of 200 overtime working hours by law and 15 tons of new material. Let d_1 to denote the duration of activity 1; $e_1^1 = 0.0005$ denotes the ratio of duration deduction per overtime working hour applied; $e_1^2 = 0.01$ denotes the ratio of duration deduction per ton of new material applied; $b_1^1 = 5$ denotes the cost of one overtime working hour; $b_1^2 = 200$ denotes per ton price of new material; θ_1^1 denotes the amount of overtime hour spent and θ_1^2 denotes the tons of new material used. We also define the upper bound for overtime hours and new materials usage given by $\bar{\theta}_1^1 = 200$ and $\bar{\theta}_1^2 = 15$.

Now the project manager has $\$500K$ available for reducing the duration of activity 1, among which $\$300K$ is allocated to option 1 (overtime work) and $\$200K$ is allocated to option 2 (new material). Then, the amount of overtime working hour is $\theta_1^1 = 300/5 = 60$ hours and the tons of new material used is $\theta_1^2 = 200/200 = 1$ ton, which result in a total reduction for activity 1 of $1000(\theta_1^1 e_1^1 + \theta_1^2 e_1^2) = 40$ hours, so that the duration of activity 1 after crashing is $1000(1 - (\theta_1^1 e_1^1 + \theta_1^2 e_1^2)) = 960$ hours.

6.1.2 Disruptions

A stochastic disruption is an event that may occur at any time during the project and results in a change—typically a significant change—in the system's parameters [37]. Examples of disruption are: Hurricane, Earthquake, Electrical Outage, Pandemic, Manual Strike, etc. When a disruption happens before the completion of a project, it will affect (delay) some of the activities in the project. We assume that a disruption can delay the duration of activities that start after the disruption and does not change the duration of activities that start on and before the disruption.

For example, a project has 5 activities and their starting times are given by $t_1 = t_2 = 0$, $t_3 = t_4 = 5$, $t_5 = 11$, their durations are given by $d_1 = 5$, $d_2 = 10$, $d_3 = 6$, $d_4 = 3$, $d_5 = 1$. If a disruption occurs at time 3, it will delay activities 3, 4, and 5 by $X_3 = 50$, $X_4 = 30$, and $X_5 = 150$ respectively, so that the new durations for activities 3, 4, and 5 that have not started becomes $d_3 + X_3 = 56$, $d_4 + X_4 = 34$, and $d_5 + X_5 = 151$. The disruption occurrence time denoted by H is assumed to be continuous random variable with known distribution and support $[0, +\infty)$. The delays associated with each activity caused by the disruption are denoted by $\{X_i\}$ and are independent continuous random variables with known distributions and support $[0, +\infty)$.

It is also assumed that H and $\{X_i\}$ are mutually independent. But when a disruption occurs, the realizations of all delays $\{X_i\}$ are known. For example, we estimate the damage of a hurricane after its occurrence.

6.1.3 Crashing of ANs with a Single Disruption

Crashing an activity network with a single disruption is addressed by Yang and Morton in [37]. For the example provided in Section 6.1.1, if activity 1 started after the occurrence of the disruption (say, a hurricane), its duration is no longer 1000 hours. Due to the effect of the hurricane, its duration is delayed by $X_1 = 2000$ hours, so that the new duration of activity 1 is $d_1 + X_1 = 3000$ hours. For the same crashing budget allocation plan provided in Section 6.1.1, we have the duration of activity 1 after crashing given by $(d_1 + X_1)(1 - (\theta_1^1 e_1^1 + \theta_1^2 e_1^2)) = 3000 * 0.94 = 2820$.

For the project provided in Figures 1.1 and 1.2, we have the durations of activities given by $d_1 = 5$, $d_2 = 10$, $d_3 = 6$, $d_4 = 3$, $d_5 = 1$, and we have the starting time of activities given by $t_1 = t_2 = 0$, $t_3 = t_4 = 5$, $t_5 = 11$ (here we assume all activities start immediately whenever feasible). Assume there is only one crashing option for each activity in Figure 1.2. For the crashing and cost parameters, assume that $e_i^1 = b_i^1 = 1, \bar{\theta}_i^1 = 0.5, \forall i \in \{1, 2, 3, 4, 5\}$. The budget is 2 and the crashing decision variables are $\theta_1^1 = 0.5$, $\theta_2^1 = 0.5$, $\theta_3^1 = 0.5$, $\theta_4^1 = 0$, $\theta_5^1 = 0.5$, and the PCT after crashing is 6. When the disruption occurred at time 3, the durations of activities 1 and 2 are unchanged, while the durations of activities 3, 4, and 5 are delayed by $X_3 = 50$, $X_4 = 30$, and $X_5 = 150$ such that their new durations are given by $d_3 + X_3 = 56$, $d_4 + X_4 = 34$, and $d_5 + X_5 = 151$. Due to contract and other issues, the budget and resources

are spent at the beginning of the activities and cannot be recalled once the activity started. Under such restriction, when the disruption happen at 3, the value of θ_1^1 and θ_2^1 cannot be changed, and we can only reallocate the rest of the budget (2-0.5-0.5=1) among activities 3, 4, and 5, which have not yet started. The original values of θ_3^1, θ_4^1 and θ_5^1 are not optimal, given the new durations of activities 3, 4 and 5. Thus, we change the crashing decision variable of activities 3, 4, and 5 to $\tilde{\theta}_3^1 = 0.5$, $\tilde{\theta}_4^1 = 0$ and $\tilde{\theta}_5^1 = 0.5$ to minimize the PCT.

The above example shows the minimization of PCT based on the remaining budget under one scenario of H and random delays in the sample space. Our goal is to find a set of initial crashing decision variables that minimize the expected PCT when the random disruption and delays are taken into account.

6.2 Problem Formulation

We now present three different formulations for the problem in Section 6.1: a Stochastic Programming (SP) formulation [38], a Mixed Integer Formulation proposed by Yang and Morton [37], and a Piecewise Linear formulation [38]. We will show in Sections 6.3 and 6.4 that our SP formulation can be used for a gradient descent method and in Section 6.6 that our SP formulation can handle more situations.

6.2.1 Stochastic Programming Formulation

Before introducing the SP formulation of the optimal crashing of AN with single disruption, the linear programming (LP) formulation of crashing ANs without disruption is first introduced here. Materials of this section are from [38].

6.2.1.1 Linear Programming Formulation of Optimal Crashing of ANs

In the A-on-N representation mode, the starting time of activities must satisfy the following condition: for any directed arcs in the AN, the starting time of the head node of the arc is greater than or equal to the summation of the starting time of the tail node of the arc and the duration of the tail node. The LP formulation of the project in Figure 1.2 is provided below:

$$
\begin{aligned}
z^* = \min_{\boldsymbol{\theta}} \quad & t_6 && && (6.1)\\
\text{s.t.} \quad & t_6 - t_5 \geq (1-\theta_5)d_5 && \\
& t_6 - t_4 \geq (1-\theta_4)d_4 && \\
& t_5 - t_3 \geq (1-\theta_3)d_3 && \\
& t_5 - t_2 \geq (1-\theta_2)d_2 && \\
& t_4 - t_1 \geq (1-\theta_1)d_1 && \\
& t_3 - t_1 \geq (1-\theta_1)d_1 && \\
& \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 \leq B && \\
& 0 \leq \theta_i \leq 0.5 && \forall i \in \mathcal{N} \\
& t_i \geq 0 && \forall i \in \mathcal{N}
\end{aligned}
$$

In Problem (6.1), t_i denotes the starting time of activity i, d_i denotes the duration of activity i and θ_i denotes the amount of crashing option applied to activity i. The decision variables in Problem (6.1) are $\{t_1, t_2, t_3, t_4, t_5, t_6, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$. For illustrative simplicity, we assume that there is only one option for each activity and $e_i = b_i = 1, \forall i \in \mathcal{N} = \{1, 2, 3, 4, 5\}$. The first 6 constraints

indicate that an activity cannot start until all of the activities it precedes are completed. The objective function is the starting time of the dummy sink node, which is the PCT. Notice that the starting time of the source dummy node t_0 does not appear in the constraints, because constraints including t_0 are included in the last constraint.

6.2.1.2 Example of LP Formulation of Crashing AN with Disruption

Recall the example provided in Section 6.1.3, when a disruption happens at time 3, it changed the duration of some activities and caused a redistribution of the remainder of the budget based on the new durations. Thus, the LP formulation of the example provided in Section 6.1.3 is given below:

$$
\begin{aligned}
\min \quad & \tilde{t}_6 && (6.2)\\
\text{s.t.} \quad & \tilde{t}_6 - \tilde{t}_5 \geq (d_5 + X_5)(1 - e_5\tilde{\theta}_5) && \text{(Type I)}\\
& \tilde{t}_6 - t_4 \geq (d_4 + X_4)(1 - e_4\tilde{\theta}_4) && \text{(Type II)}\\
& \tilde{t}_5 - t_3 \geq (d_3 + X_3)(1 - e_3\tilde{\theta}_3) && \text{(Type II)}\\
& \tilde{t}_5 - t_2 \geq d_2(1 - e_2\theta_2) && \text{(Type III)}\\
& \theta_1 + \theta_2 + \tilde{\theta}_3 + \tilde{\theta}_4 + \tilde{\theta}_5 \leq 1\\
& 0 \leq \tilde{\theta}_3, \tilde{\theta}_4, \tilde{\theta}_5 \leq 1\\
& \tilde{t}_3, \tilde{t}_4, \tilde{t}_5, \tilde{t}_6 \geq 0
\end{aligned}
$$

In Problem (6.2), the decision variables are $\{\tilde{t}_5, \tilde{t}_6, \tilde{\theta}_3, \tilde{\theta}_4, \tilde{\theta}_5\}$, and $\{t_2, t_3, t_4, \theta_1, \theta_2\}$ are not decision variables in Problem (6.2), but are feasible solutions from Problem (6.1). Notice that the starting

times of activities 3, 4, 5, and 6 are uncertain, because they depend on the choice of the redistributed crashing decision variables $\tilde{\theta}_3$, $\tilde{\theta}_4$, and $\tilde{\theta}_5$.

6.2.1.3 Stochastic Programming Formulation

From Sections 6.2.1.1 and 6.2.1.2, it is clear that our problem is a two-stage stochastic programming problem. The general formulation is provided below. The first-stage problem is given by:

$$z^* = \min_{\mathbf{t},\boldsymbol{\theta}} \quad g(\mathbf{t}, \boldsymbol{\theta}) \tag{6.3}$$

$$\text{s.t.} \quad t_k - t_i \geq d_i(1 - \sum_{j \in J_i} e_i^j \theta_i^j) \qquad \forall (i,k) \in \mathcal{A} \tag{6.3a}$$

$$\sum_{i \in I} \sum_{j \in J_i} b_i^j \theta_i^j \leq B \tag{6.3b}$$

$$\sum_{j \in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \tag{6.3c}$$

$$0 \leq \theta_i^j \leq \bar{\theta}_i^j \qquad \forall j \in J_i, i \in \mathcal{N} \tag{6.3d}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{6.3e}$$

In Problem (6.3), $\mathbf{t}$ and $\boldsymbol{\theta}$ are the vector form of activity starting times and crashing decision variables for all options of all activities. $(\mathbf{t}, \boldsymbol{\theta})$ are the decision variables of Problem (6.3). $\mathcal{A}$ is the set of all arcs in the AN, $\mathcal{N}$ is the set of nodes in the AN, and J_i is the set of crashing options for activity i. Also $g(\mathbf{t}, \boldsymbol{\theta})$ is defined as:

$$g(\mathbf{t}, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H)] \tag{6.4}$$

where $f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H)$ is a function of activity starting times and crashing decision variables from the first-stage problem, and the realization of the random delay and disruption occurrence, defined by the output as the optimal value of the second-stage problem given below:

$$
\begin{aligned}
f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H) = \min \quad & t_{N_D} I\{q_{N_D} \leq H\} + \tilde{t}_{N_D} I\{q_{N_D} > H\} && (6.5)\\
\text{s.t.} \quad & t_k I\{q_k \leq H\} + \tilde{t}_k I\{q_k > H\} - (t_i I\{q_i \leq H\} + \tilde{t}_i I\{q_i > H\}) \\
& \geq (X_i I\{t_i > H\} + d_i) \\
& \qquad (1 - \sum_{j \in J_i} e_i^j (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\})) && \forall (i,k) \in \mathcal{A} \\
& && (6.5a)\\
& \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\}) \leq B && (6.5b)\\
& \sum_{j \in J_i} (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\}) \leq 1 && \forall i \in \mathcal{N} \\
& && (6.5c)\\
& 0 \leq \theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\} \leq \bar{\theta}_i^j && \forall i \in \mathcal{N}, j \in J_i \\
& && (6.5d)\\
& t_i I\{q_i \leq H\} + \tilde{t}_i I\{q_i > H\} \geq 0 && \forall i \in \mathcal{N} \\
& && (6.5e)
\end{aligned}
$$

In Problem (6.5), $q_k = \max_{i \in \textbf{pred}(k)} \{t_i\}$, q_k is the node starting time the predecessor nodes of node k that has the largest node (activity) starting time, where predecessor is defined in Section 1.2.6. In the second-stage Problem (6.5), all variables with a tilde are decision variables for the second-stage problem and for variables without a tilde, they are either input of the function (first-

stage decision variables) or parameters. In the second-stage Problem (6.5), the set of constraints

Table 6.1: Four Types of Second-stage Constraints

Types	Constraint	Range of H
Type I	$\tilde{t}_k - \tilde{t}_i \geq (X_i + d_i)(1 - \sum_{j \in J_i} e_i^j \tilde{\theta}_i^j)$	$H < q_i$
Type II	$\tilde{t}_k - t_i \geq (X_i + d_i)(1 - \sum_{j \in J_i} e_i^j \tilde{\theta}_i^j)$	$q_i \leq H < t_i$
Type III	$\tilde{t}_k - t_i \geq d_i(1 - \sum_{j \in J_i} e_i^j \theta_i^j)$	$t_i \leq H < q_k$
Type IV	$t_k - t_i \geq d_i(1 - \sum_{j \in J_i} e_i^j \theta_i^j)$	$q_k \leq H$

in (6.5a) are classified into four mutually exclusive classes listed in Table 6.1. Notice that Type IV is not included in the second-stage Problem (6.5a), because all variables in Type IV are first-stage variables. Thus, for H fixed, each constraint in (6.5a) only belongs to one of the first three types in Table 6.1. Given $(\mathbf{t}, \boldsymbol{\theta})$ and H, we denote the set of activities in the second-stage Problem (6.5) that belongs to Type i in Table 6.1 as $\mathcal{A}_i^H$, the set of activities with stochastic delayed durations as $\mathcal{N}_1^H$, and the set of activities with fixed durations as $\mathcal{N}_2^H$. Then, we have that $\bigcup_{i=1}^{4} \mathcal{A}_i^H = \mathcal{A}$, $\mathcal{A}_i^H \cap \mathcal{A}_j^H = \emptyset, \forall i \neq j$ and $\mathcal{N}_1^H \cup \mathcal{N}_2^H = \mathcal{N}$, $\mathcal{N}_i^H \cap \mathcal{N}_j^H = \emptyset$. Examples of three types of constraints are provided in Problem (6.2).

6.2.1.4 Further Formulation of the Two-stage SP Problem

Recall that the formulation in Section 6.2.1.3 has two sets of decision variables: the starting time of activities $\mathbf{t}$ and the amount of crashing for each options of all activities $\boldsymbol{\theta}$. From the constraints in first-stage Problem (6.3), we know that $\mathbf{t}$ and $\boldsymbol{\theta}$ are not independent. Because we want to apply gradient based optimization, it is desirable to have the set of decision variables independent. As a result, the decision variable $\mathbf{t}$ is expressed as a function of $\boldsymbol{\theta}$ and an independent non-negative auxiliary variable $\boldsymbol{\epsilon}$. We express two equivalent formulas for calculating the activity

starting time $\mathbf{t}$:

$$\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]} = \max_{i \in \mathbf{pred}(k)} \Big\{ d_i (1 - \sum_{j \in J_i} e_i^j \theta_i^j) + \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} \Big\} + \epsilon_k, \quad \forall k \in \mathcal{N}, \tag{6.6a}$$

$$\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]} = \max_{P \in \mathcal{P}_k} \Big\{ \sum_{i \in P - \{k\}} d_i (1 - \sum_{j \in J_i} e_i^j \theta_i^j) + \epsilon_i \Big\} + \epsilon_k, \quad \forall k \in \mathcal{N}. \tag{6.6b}$$

In Equations (6.6a) and (6.6b), $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ is a vector whose k^{th} element is the starting time of activity k, denoted by $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]}$. We always have that $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[0]} = 0$, $d_0 = 0$ and $\epsilon_0 = 0$. Equation (6.6a) is a recursive formula for calculating activity starting times, while Equation (6.6b) is another formula for calculating activity starting times. For any feasible solution $(\mathbf{t}_1, \boldsymbol{\theta}_1)$, there exists $\boldsymbol{\epsilon}_1$ such that $\mathbf{t}(\boldsymbol{\theta}_1, \boldsymbol{\epsilon}_1) = \mathbf{t}_1$, and vice versa. Hence, using the formulation of $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, we can transform Problem (6.3) into an equivalent problem:

$$z^* = \min_{\boldsymbol{\theta}, \boldsymbol{\epsilon}} \quad g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) \tag{6.7}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^j \leq B \tag{6.7a}$$

$$\sum_{j \in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \tag{6.7b}$$

$$0 \leq \theta_i^j \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \tag{6.7c}$$

$$\epsilon_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{6.7d}$$

Notice that in Problem (6.7) all constraints involving $\mathbf{t}$ are eliminated, because $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ automatically satisfies constraints (6.3a) and (6.3e). As a consequence, Equation (6.4) becomes:

$$g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, H}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)] \tag{6.4b}$$

and the second-stage Problem (6.5) becomes:

$$
\begin{aligned}
f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H) = \min \quad & \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[N_D]} I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[N'_D]} \le H\} + \tilde{t}_{N_D} I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[N'_D]} > H\} && (6.8)\\
\text{s.t.} \quad & \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[k]} I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[k']} \le H\} + \tilde{t}_k I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[k']} > H\} \\
& - (\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} \le H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} > H\}) \\
& \ge (X_i I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} > H\} + d_i)\Big(1 - \sum_{j\in J_i} e_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} \le H\} \\
& \quad + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} > H\})\Big) \qquad \forall (i,k) \in \mathcal{A} && (6.8a)\\
& \sum_{i\in\mathcal{N}} \sum_{j\in J_i} b_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} \le H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} > H\}) \le B \\
& && (6.8b)\\
& \sum_{j\in J_i} (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} \le H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} > H\}) \le 1 \qquad \forall i \in \mathcal{N} \\
& && (6.8c)\\
& 0 \le \theta_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} \le H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} > H\} \le \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \\
& && (6.8d)\\
& \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} \le H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} > H\} \ge 0 \qquad \forall i \in \mathcal{N} \\
& && (6.8e)
\end{aligned}
$$

In Problem (6.8), $k' = \underset{i\in\mathbf{pred}(k)}{\arg\max} \; t_i$, i.e., k' is the index of the predecessor node of node k that has the largest starting time, similarly for i'. $(\boldsymbol{\theta},\boldsymbol{\epsilon})$ are the first-stage decision variables from Problem (6.8), and $\mathbf{X}$ is a realization of the random delays of durations of activities and H is the occurrence time of the disruption. The SP problem formulated by (6.7) and (6.8) is equivalent to the formulation (6.3) and (6.5). Notice that the two-stage stochastic programming problem is

different from the commonly used two-stage SP in [39, 40], in the sense that indicator functions are included.

Recall that in Section 5.3, we mentioned that the inequality constraint on the budget can be replaced an equality constraint. The same can be done for constraint (6.7a) in Problem (6.7). However, it is not trivial to check the validity for that reason that $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ is not a decreasing function of θ_i for all i. Thus, the following Theorem is provided to validate the exchange of inequality with equality in constraint (6.7a).

We define a new first-stage problem with its optimal value as a function of budget B:

$$
\begin{aligned}
z(B) = \min_{\boldsymbol{\theta}, \boldsymbol{\epsilon}} \quad & g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) && && \text{(6.7S)} \\
\text{s.t.} \quad & \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^j = B && && \text{(6.7S.a)} \\
& \sum_{j \in J_i} \theta_i^j \leq 1 && \forall i \in \mathcal{N} && \text{(6.7S.b)} \\
& 0 \leq \theta_i^j \leq \bar{\theta}_i^j && \forall i \in \mathcal{N}, j \in J_i && \text{(6.7S.c)} \\
& \epsilon_i \geq 0 && \forall i \in \mathcal{N} && \text{(6.7S.d)}
\end{aligned}
$$

Theorem 2. *$z(B)$ is a decreasing function of B.*

Proof. Proof of Theorem 2. We first present Algorithm 4 for allocating extra budget $\delta > 0$.

Notice that in Problem (6.8), the index sets $\mathcal{A}_i^H, i = 1, 2, 3, 4$ and $\mathcal{N}_i^H, i = 1, 2$ all depend on $(\boldsymbol{\theta}, \boldsymbol{\epsilon}, H)$. Let Ω be the sample space of $(\mathbf{X}, H)$, $\mathcal{O}_k^H = \{(i, j) | i \in \mathcal{N}_k^H, j \in J_i\}$. Let $\bar{B} = \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \bar{\theta}_i^j$, for any $B' \in [0, \bar{B}]$ and $\delta \in [0, \bar{B} - B']$ fixed, we want to show that $z(B') \leq z(B' + \delta)$. For $B = B'$, its associated optimal solution is $(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)$. Notice the difference

Algorithm 4: Allocate Extra Budget

Input : $\boldsymbol{\theta}^*$,$\boldsymbol{\epsilon}^*$,δ

Output: $\boldsymbol{\theta}$

$n \leftarrow 0$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$

$\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}^*$

while $\delta > 0$ **do**

  $i_n \leftarrow \arg\max_i \{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} | \exists j \in J_i, \theta_i^j < \bar{\theta}_i^j \text{ and } \sum_{j \in J_i} \theta_i^j < 1\}$

  $j_n \leftarrow \arg\max_j \{b_{i_n}^j | \theta_{i_n}^j < \bar{\theta}_{i_n}^j\}$

  $\delta_n \leftarrow \min\{\bar{\theta}_{i_n}^{j_n} - \theta_{i_n}^{j_n}, \frac{\delta}{b_{i_n}^{j_n}}, 1 - \sum_{j \in J_{i_n}, j \neq j_n} \theta_i^j\}$

  $\theta_{i_n}^{j_n} \leftarrow \theta_{i_n}^{j_n} + \delta_n$

  $\epsilon_i \leftarrow 0, \forall i \in \{i | \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} > \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i_n]}\}$

  $\boldsymbol{\theta}^{(n)} \leftarrow \boldsymbol{\theta}$; $\boldsymbol{\epsilon}^{(n)} \leftarrow \boldsymbol{\epsilon}$; $\delta \leftarrow \delta - \delta_n$

  $n \leftarrow n + 1$

end

between Problem (6.7S) and Problem (5.6): the objective function is (5.6) is a decreasing function of each coordinate decision variables, which is not true for Problem (6.7S); increasing a crashing decision variable in (6.7S) may increase the objective function. As a result, we need a way to allocate the extra budget so that the objective function decreases, and Algorithm 4 provides a way of allocating the extra budget. Assume we have δ extra budget to allocate, we allocate δ based on the Algorithm 4:

Assume when Algorithm 4 stops, $n = m \geq 1$ and $\boldsymbol{\theta}^{(k)}$ is the $\boldsymbol{\theta}$ value after k^{th} iteration of Algorithm 4. Next we will show that the above algorithm will generate a new crashing decision variable $\boldsymbol{\theta}^{(m)}$ such that $g(\mathbf{t}(\boldsymbol{\theta}^{(m)}, \boldsymbol{\epsilon}^*), \boldsymbol{\theta}^{(m)}) \leq g(\mathbf{t}(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*), \boldsymbol{\theta}^*)$ and $\sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^{j(m)} = B + \delta$, where $\{\theta_i^{j(m)}\}$ are elements of $\boldsymbol{\theta}^{(m)}$.

Let $k = 0$, for each $(\mathbf{X}, H) \in \Omega$, we have

$f(\mathbf{t}(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}), \boldsymbol{\theta}^{(0)}, \mathbf{X}, H)$

$$= \inf_{\substack{0\le\tilde{\theta}_i^j\le\bar{\theta}_i^j,\forall(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{j\in J_i}\tilde{\theta}_i^j\le 1,\forall i\in\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\tilde{\theta}_i^j \le B+\delta_0-\sum_{(i,j)\in\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\theta_i^{j(0)}}} \max_{P\in\mathcal{P}_{N_D}}\Bigg\{\sum_{i\in P\cap\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j(0)})+\epsilon_i^{(0)}\Big)$$

$$+\sum_{i\in P\cap\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)\Bigg\} \tag{T2.a}$$

$$= \inf_{\substack{0\le\tilde{\theta}_i^j\le\bar{\theta}_i^j,\forall(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{j\in J_i}\tilde{\theta}_i^j\le 1,\forall i\in\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\tilde{\theta}_i^j \le B-\sum_{(i,j)\in\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\theta_i^{j*}}} \max_{P\in\mathcal{P}_{N_D}}\Bigg\{\sum_{i\in P\cap\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j(0)})+\epsilon_i^{(0)}\Big)+$$

$$\sum_{i\in P\cap\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)+\sum_{i\in P\cap(\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*))}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j(0)})+\epsilon_i^{(0)}\Big)\Bigg\}$$

(T2.b)

$$= \inf_{\substack{0\le\tilde{\theta}_i^j\le\bar{\theta}_i^j,\forall(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{j\in J_i}\tilde{\theta}_i^j\le 1,\forall i\in\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\\ \sum_{(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\tilde{\theta}_i^j \le B-\sum_{(i,j)\in\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}\theta_i^{j*}}} \max_{P\in\mathcal{P}_{N_D}}\Bigg\{\sum_{i\in P\cap\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j(0)})+\epsilon_i^{(0)}\Big)+$$

$$\sum_{i\in P\cap\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)+\sum_{i\in P\cap(\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*))}d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j(0)})\Bigg\}$$

(T2.c)

$$\le \inf_{\substack{0\le\tilde{\theta}_i^j\le\bar{\theta}_i^j,\forall(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\ \sum_{j\in J_i}\tilde{\theta}_i^j\le 1,\forall i\in\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\ \sum_{(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\tilde{\theta}_i^j \le B-\sum_{(i,j)\in\mathcal{O}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\theta_i^{j*}}} \max_{P\in\mathcal{P}_{N_D}}\Bigg\{\sum_{i\in P\cap\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j*})+\epsilon_i^*\Big)+$$

$$\sum_{i\in P\cap\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)+\sum_{i\in P\cap(\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*))}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)\Bigg\}$$

(T2.d)

$$= \inf_{\substack{0\le\tilde{\theta}_i^j\le\bar{\theta}_i^j,\forall(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\ \sum_{j\in J_i}\tilde{\theta}_i^j\le 1,\forall i\in\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\ \sum_{(i,j)\in\mathcal{O}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\tilde{\theta}_i^j \le B-\sum_{(i,j)\in\mathcal{O}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\theta_i^{j*}}} \max_{P\in\mathcal{P}_{N_D}}\Bigg\{\sum_{i\in P\cap\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}\Big(d_i(1-\sum_{j\in J_i}e_i^j\theta_i^{j*})+\epsilon_i^*\Big)+$$

$$\sum_{i\in P\cap\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)}(d_i+X_i)(1-\sum_{j\in J_i}e_i^j\tilde{\theta}_i^j)\Big\} \tag{T2.e}$$

$$=f(\mathbf{t}(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*),\boldsymbol{\theta}^*,\mathbf{X},H).$$

Equation (T2.a) is the piecewise linear form of $f(\mathbf{t}(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)}),\boldsymbol{\theta}^{(0)},\mathbf{X},H)$. At each iteration of Algorithm 4, one of θ_i^j is increased to its upper bound. By increasing θ_i^j, the duration of activity i is decreased, as a result reducing the starting time of some of the activities that originally start after activity i. Hence, after the first iteration of Algorithm 4, for a fixed disruption occurrence time H, the set of activities starts after H is getting smaller, i.e., $\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})$ is smaller than $\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)$. We also have the following:

$$\begin{aligned}
&\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\cup\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})=\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\cup\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)=\mathcal{N},\\
&\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\subset\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\
\Rightarrow&\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\subset\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)}),\\
&\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)-\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})=\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\\
\Rightarrow&I_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})=\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\cup(\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)),\\
&\mathcal{N}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)=\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\cup(\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{N}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)),
\end{aligned} \tag{T2.f}$$

$$\begin{aligned}
\Rightarrow&\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})=\mathcal{O}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)\cup(\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{O}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)),\\
&\mathcal{O}_1^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)=\mathcal{O}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})\cup(\mathcal{O}_2^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})-\mathcal{O}_2^H(\boldsymbol{\theta}^*,\boldsymbol{\epsilon}^*)).
\end{aligned} \tag{T2.g}$$

From (T2.a) to (T2.b) is a result of decomposing $\mathcal{N}_1^H(\boldsymbol{\theta}^{(0)},\boldsymbol{\epsilon}^{(0)})$ into two parts using Equation (T2.f). Based on the choice of i_n in Algorithm 4, for all activities starting after activity i_n, their corresponding ϵ_i all equal to 0 and their crashing parameters $\{\theta_i^j\}$ all at their maximum. Also,

we have

$$t_i(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*) > t_{i_0}(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*), \quad \forall i \in P \cap (\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{N}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*))$$

$$\Rightarrow \epsilon_i^{(0)} = 0, \tilde{\theta}_i^j \leq \theta_i^{j(0)}, \quad \forall (i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*) \tag{T2.h}$$

$$\Rightarrow d_i(1 - \sum_{j \in J_i} e_i^j \theta_i^{j(0)}) \leq (d_i + X_i)(1 - \sum_{j \in J_i} e_i^j \tilde{\theta}_i^j), \quad \forall i \in P \cap (\mathcal{N}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{N}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)), \tag{T2.i}$$

it is clear that (T2.b) equals to (T2.c). For every feasible solution of the minimization problem (T2.d), $\{\tilde{\theta}_i^j\}_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)}$, we have

$$\begin{aligned}
&\sum_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \tilde{\theta}_i^j \leq B - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j*} \\
\Rightarrow &\sum_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)})} \tilde{\theta}_i^j \leq B - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j*} - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \tilde{\theta}_i^j \\
\Rightarrow &\sum_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)})} \tilde{\theta}_i^j \leq B - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j*} - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j(0)} \\
\Rightarrow &\sum_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)})} \tilde{\theta}_i^j \leq B - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j*} - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}) - \mathcal{O}_2^H(\boldsymbol{\theta}^*, \boldsymbol{\epsilon}^*)} \theta_i^{j*} \\
\Rightarrow &\sum_{(i,j) \in \mathcal{O}_1^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)})} \tilde{\theta}_i^j \leq B - \sum_{(i,j) \in \mathcal{O}_2^H(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)})} \theta_i^{j(0)},
\end{aligned} \tag{T2.j}$$

(T2.j) shows that for each feasible solutions of the minimization Problem (T2.d), it is also feasible in Problem (T2.c), i.e., the feasible region of Problem (T2.d) is smaller than or equal to that of Problem (T2.c). From (T2.i), we also have that for each element in the Max function (for each path in the network), its corresponding value (its path length) of (T2.d) is smaller than or equal to that of (T2.c). In conclusion, we have checked the validity from (T2.c) to (T2.d).

Repeat (T2.a) to (T2.e) for $\{(\boldsymbol{\theta}^{(k)}, \boldsymbol{\epsilon}^{(k)})\}_{k=1}^{m}$, then we have

$$\forall(\mathbf{X}, H) \in \Omega, \forall 1 \leq k \leq m, f(\mathbf{t}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\epsilon}^{(k)}), \boldsymbol{\theta}^{(k)}, \mathbf{X}, H) \leq f(\mathbf{t}(\boldsymbol{\theta}^{(k-1)}, \boldsymbol{\epsilon}^{(k-1)}), \boldsymbol{\theta}^{(k-1)}, \mathbf{X}, H)$$

$$\forall(\mathbf{X}, H) \in \Omega, f(\mathbf{t}(\boldsymbol{\theta}^{(0)}, \boldsymbol{\epsilon}^{(0)}), \boldsymbol{\theta}^{(0)}, \mathbf{X}, H) \leq f(\mathbf{t}(\boldsymbol{\theta}^{*}, \boldsymbol{\epsilon}^{*}), \boldsymbol{\theta}^{*}, \mathbf{X}, H)$$

$$\Rightarrow \forall(\mathbf{X}, H) \in \Omega, f(\mathbf{t}(\boldsymbol{\theta}^{(m)}, \boldsymbol{\epsilon}^{(m)}), \boldsymbol{\theta}^{(m)}, \mathbf{X}, H) \leq f(\mathbf{t}(\boldsymbol{\theta}^{*}, \boldsymbol{\epsilon}^{*}), \boldsymbol{\theta}^{*}, \mathbf{X}, H)$$

$$\Rightarrow \mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}(\boldsymbol{\theta}^{(m)}, \boldsymbol{\epsilon}^{(m)}), \boldsymbol{\theta}^{(m)}, \mathbf{X}, H)] \leq \mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}(\boldsymbol{\theta}^{*}, \boldsymbol{\epsilon}^{*}), \boldsymbol{\theta}^{*}, \mathbf{X}, H)]$$

$$\Rightarrow g(\mathbf{t}(\boldsymbol{\theta}^{(m)}, \boldsymbol{\epsilon}^{(m)}), \boldsymbol{\theta}) \leq g(\mathbf{t}(\boldsymbol{\theta}^{*}, \boldsymbol{\epsilon}^{*}), \boldsymbol{\theta})$$

$$\Rightarrow z(B + \delta) \leq z(B)$$

In conclusion, $z(B)$ is a decreasing function of B.

□

Applying Theorem 2, Problems (6.7S) and (6.7) are equivalent. Then, our final two-stage stochastic programming formulation with indicator functions included is given by Problem (6.7S) and (6.8).

6.2.2 Mixed Integer Formulation

Yang and Morton provided a discrete mixed integer version of formulation of Problem (6.3) and (6.5) in [37]. The problem is given below:

$$z^* = \min_{\mathbf{t},\boldsymbol{\theta}} \quad p^0 t_{N_D} + \sum_{\omega\in\Omega} p^\omega f^\omega(\mathbf{t},\boldsymbol{\theta}) \tag{6.9}$$

$$\text{s.t.} \quad t_k - t_i \geq d_i(1 - \sum_{j\in J_i} e_i^j \theta_i^j) \qquad \forall (i,k) \in \mathcal{A} \tag{6.9a}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in J_i} b_i^j \theta_i^j \leq B \tag{6.9b}$$

$$\sum_{j\in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \tag{6.9c}$$

$$0 \leq \theta_i^j \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \tag{6.9d}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{6.9e}$$

And the second-stage problem corresponding to Problem (6.5) is given by:

$$f^\omega(\hat{\mathbf{t}},\hat{\boldsymbol{\theta}}) = \min_{\hat{\mathbf{t}},\hat{\boldsymbol{\theta}}} \quad t_{N_D} \tag{6.10}$$

$$\text{s.t.} \quad H^\omega + MG_i \geq t_i \qquad \forall i \in \mathcal{N} \tag{6.10a}$$

$$H^\omega - M(1 - G_i) \leq t_i \qquad \forall i \in \mathcal{N} \tag{6.10b}$$

$$t_i + M'G_i \geq \hat{t}_i \qquad \forall i \in \mathcal{N} \tag{6.10c}$$

$$t_i - M'G_i \geq \hat{t}_i \qquad \forall i \in \mathcal{N} \tag{6.10d}$$

$$\theta_i^j + \bar{\theta}_i^j G_i \geq \hat{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \tag{6.10e}$$

$$\theta_i^j - \bar{\theta}_i^j G_i \leq \hat{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \tag{6.10f}$$

$$t_k - t_i \geq d_i + X_i^{\omega} G_i$$

$$- \sum_{j \in J_i} d_i e_i^j \theta_i^j - \sum_{j \in J_i} X_i^{\omega} e_i^j z_i^j \qquad \forall (i,k) \in \mathcal{A} \qquad (6.10g)$$

$$\sum_{i \in I} \sum_{j \in J_i} b_i^j \theta_i^j \leq B \qquad (6.10h)$$

$$\sum_{j \in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \qquad (6.10i)$$

$$z_i^j \leq \bar{\theta}_i^j G_i \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.10j)$$

$$z_i^j \leq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.10k)$$

$$z_i^j \leq \theta_i^j + \bar{\theta}_i^j (G_i - 1) \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.10l)$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \qquad (6.10m)$$

$$t_i \geq H^{\omega} G_i \qquad \forall i \in \mathcal{N} \qquad (6.10n)$$

$$0 \leq \theta_i^j \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.10o)$$

$$0 \leq z_i^j \leq 1 \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.10p)$$

$$G_i \in \{0,1\} \qquad \forall i \in \mathcal{N}, \qquad (6.10q)$$

where in Problem (6.10), p^0 is the probability that disruption will not happen, ω is a scenario in the sample space of stochastic duration delays $\mathbf{X}$ and disruption occurrence H. In Problem (6.10), $\{\hat{t}_i\}$ are first stage variables and $\{t_i\}$ are second stage variables, G_i equals 1 if activity i starts after the disruption occurrence and equals 0 otherwise. M and M' are constants, sufficiently large positive numbers. Constraints (6.10c) - (6.10f) guarantee that for activities not affected by the disruption (start before the disruption), their starting time and crashing decision equal the first-stage decision values in Problem (6.8). Constraint (6.10n) restricts the second-stage starting time

of activities to be no later than the disruption occurrence time. For each activity i, its duration becomes $(d_i + X_i G_i)(1 - \sum_{j \in J_i} e_i^j \theta_i^j)$, which expands to the right hand side of constraint (6.10g). Notice that $(d_i + X_i G_i)(1 - \sum_{j \in J_i} e_i^j \theta_i^j)$ contains bilinear terms $\{G_i \theta_i^j\}$, which are linearized using variable z_i^j, and constraints (6.10j) - (6.10l).

Yang and Morton [37] also provide an extensive formulation of the two-stage stochastic mixed integer programming, which is presented below:

$$z^* = \min_{\mathbf{t}, \boldsymbol{\theta}} \quad p^0 t_{N_D} + \sum_{\omega \in \Omega} p^\omega f^\omega(\mathbf{t}, \boldsymbol{\theta}) \tag{6.11}$$

$$\text{s.t.} \quad t_k - t_i \geq d_i (1 - \sum_{j \in J_i} e_i^j \theta_i^j) \qquad \forall (i,k) \in \mathcal{A} \tag{6.11a}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^j \leq B \tag{6.11b}$$

$$\sum_{j \in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \tag{6.11c}$$

$$H^\omega + M G_i^\omega \geq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.11d}$$

$$H^\omega - M(1 - G_i^\omega) \leq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.10e}$$

$$t_i^\omega + M' G_i^\omega \geq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.10f}$$

$$t_i^\omega - M' G_i^\omega \leq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.10g}$$

$$\theta_i^{j\omega} + \bar{\theta}_i^j G_i^\omega \geq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \tag{6.10h}$$

$$\theta_i^{j\omega} - \bar{\theta}_i^j G_i^\omega \geq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \tag{6.10i}$$

$$t_k^\omega - t_i^\omega \geq d_i + X_i^\omega G_i^\omega - \sum_{j \in J_i} d_i e_i^j \theta_i^{j\omega} - \sum_{j \in J_i} X_i^\omega e_i^j z_i^{j\omega} \qquad \forall (i,k) \in \mathcal{A}, \omega \in \Omega \tag{6.11j}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^{j\omega} \leq B \qquad \forall \omega \in \Omega \tag{6.11k}$$

$$\sum_{j \in J_i} \theta_i^{j\omega} \leq 1 \qquad \forall i \in \mathcal{N}, \omega \in \Omega \qquad (6.11l)$$

$$z_i^{j\omega} \leq \bar{\theta}_i^j G_i^\omega \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.11m)$$

$$z_i^{j\omega} \leq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.11n)$$

$$z_i^{j\omega} \leq \theta_i^j + \bar{\theta}_i^j (G_i^\omega - 1) \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.11o)$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \qquad (6.11p)$$

$$t_i^\omega \geq H^\omega G_i^\omega \qquad \forall i \in \mathcal{N}, \omega \in \Omega \qquad (6.11q)$$

$$0 \leq \theta_i^j \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.11r)$$

$$0 \leq \theta_i^{j\omega} \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.11s)$$

$$0 \leq z_i^{j\omega} \leq 1 \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.11t)$$

$$G_i^\omega \in \{0, 1\} \qquad \forall i \in \mathcal{N}, \omega \in \Omega. \qquad (6.11u)$$

Notice that in the formulation (6.11), it is assumed that the sample space of stochastic delay of activities' durations and disruption occurrence has a finite number of elements, where ω in Problem (6.11) stands for a scenario in the sample space. Then Problem (6.11) can be solved using Sample Average Approximation (SAA). Yang and Morton proved that Problem (6.11) is NP-hard even with a single disruption scenario that occurs with probability one at time zero [37]. As a result, they formulate the problem as a two-stage stochastic mixed integer programming problem using (6.9) and (6.10) and develop a branch-and-cut decomposition SAA based algorithm to solve (6.9) and (6.10) [37].

6.2.3 Difference between SP and Mixed Integer Formulations

The two-stage stochastic programming formulation in (6.3) and (6.5) is different from the two-stage mixed integer formulation in (6.9) and (6.10) in two ways: (1) Yang and Morton's [37] formulation assumes the event that disruption will not happen has a positive probability, not just that the support of the density function of disruption occurrence time is from 0 to positive infinity; our formulation in Section 6.2.1 does not have this assumption; (2) The criterion determining the starting time of activity i a first-stage or second-stage decision variable in two formulations are different. In Yang and Morton's [37] two-stage mixed integer formulation, the criterion is whether or not: activity i starts before the disruption. While in our formulation, the criterion is whether or not: all of activity i's predecessor nodes (activities) start before the disruption.

For the first difference between the two formulations, the objective function of Problem (6.7S) can be adjusted by adding one more term to accommodate the difference. That is, the objective function in Problem (6.3) is adjusted to:

$$p_0 t_{N_D} + (1 - p_0)\mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H)]$$

For the second difference, our formulation in Problem (6.5) can be changed to accommodate

Problem (6.10) and is provided by

$$f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H) = \min \quad t_{N_D} I\{t_{N_D} \leq H\} + \tilde{t}_{N_D} I\{t_{N_D} > H\} \tag{6.5A}$$

$$\text{s.t.} \quad t_k I\{t_k \leq H\} + \tilde{t}_k I\{t_k > H\} - (t_i I\{t_i \leq H\} + \tilde{t}_i I\{t_i > H\})$$

$$\geq (X_i I\{t_i > H\} + d_i)$$

$$(1 - \sum_{j \in J_i} e_i^j (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\})) \qquad \forall (i,k) \in \mathcal{A}$$

(6.5A.a)

$$\sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\}) \leq B \tag{6.5A.b}$$

$$\sum_{j \in J_i} (\theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\}) \leq 1 \qquad \forall i \in \mathcal{N}$$

(6.5A.c)

$$0 \leq \theta_i^j I\{t_i \leq H\} + \tilde{\theta}_i^j I\{t_i > H\} \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i$$

(6.5A.d)

$$t_i I\{t_i \leq H\} + \tilde{t}_i I\{t_i > H\} \geq I\{t_i > H\} H \qquad \forall i \in \mathcal{N}.$$

(6.5A.e)

The difference between Problem (6.5A) and Problem (6.5) is that in (6.5A), all $\{q_i\}$ are replaced by $\{t_i\}$, and the RHS of constraint (6.5e) becomes $I\{t_i > H\}H$ in constraint (6.5A.e). Although Problem (6.5A) is equivalent to Problem (6.5), we prefer Problem (6.5) over Problem (6.5A), because in the next section, we need $f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, H)$ to not depend on H once the interval in which H falls is known, and the term $I\{t_i > H\}H$ in constraint (6.5A.e) precludes this.

6.2.4 Piecewise Linear Formulation

The piecewise linear formulation takes advantage of the fact that PCT is the maximum of all path lengths in the AN. Notice that there are also constraints about the budget, crashing decision variables bounds, etc. We have seen the piecewise linear formulation in the proof of Theorem 2. The piecewise linear formulation is given by:

$$
\begin{aligned}
& g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) \\
=& \inf_{E_1} \quad \mathbb{E}_{\mathbf{X},H}\Bigg[\inf_{E_2} \max_{P \in \mathcal{P}_{N_D}} \Bigg\{ \sum_{i \in P} (X_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} > H\} + d_i)\Big(1 - \sum_{j \in J_i} e_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} \leq H\} \\
& \qquad + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} > H\})\Big)\Bigg\}\Bigg]
\end{aligned} \tag{6.12}
$$

where E_1 denotes constraints (6.7a) - (6.7d) and E_2 denotes constraints (6.8b) - (6.8d).

6.3 Stochastic Gradient Estimation

The materials in Sections 6.3 - 6.7 are from [38]. In the next section, we develop a simulation optimization algorithm that is based on a stochastic gradient estimated by Monte Carlo simulation. A simulation-based approach to two-stage stochastic programming with recourse was first proposed by [41]. In this section, we need to derive an unbiased stochastic gradient estimator for the gradient of $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$, the objective function in Problem (6.7S). The difference between two-stage stochastic programming with recourse [39, 40] and the problem setting in this paper is that there are indicator functions of the first-stage variables in the second-stage problem in our problem formulation. In our formulation, whether a variable belongs to the first stage or second

stage depends on the independent random event called the disruption. The indicator function increases the complexity of the problem and also creates new technical challenges in deriving a stochastic gradient estimator. There are two challenges in deriving an unbiased gradient estimator for $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$: (1) function inside the expectation $f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)$ does not have a closed form; moreover, if we express $f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)$ in the piecewise linear form in Section 6.2.4, the derivative variable appears both in the feasible region and objective function; (2) There are indicator functions in $f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)$ in all formulations. As for the first challenge, Shipario et al. [41] proposed a gradient-based algorithm for solving stochastic programming problems. As for the second challenge, in Section 6.3.1, we [38] propose an unbiased estimator of the stochastic gradient by pushing the first-stage variables out of the indicator function and moving it into the measure of the disruption. Using the unbiased stochastic gradient estimator and KKT condition, we propose a gradient-based simulation optimization algorithm for the two-stage stochastic mixed integer programming problem with constraints. Our estimator and algorithm can be extended to more generalized multi-stage stochastic programming problems with indicator functions included in both the objective function and constraints.

6.3.1 Conditional Expectation and Gradient

The key to estimate the stochastic gradient $\nabla g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ is to estimate the stochastic gradient of Equation (6.4b): $\nabla \mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)]$. Notice that the constraints in Problem (6.8) all include indicator functions of disruption happening time H and first stage variable $(\boldsymbol{\theta}, \boldsymbol{\epsilon})$. Thus, we will estimate $\nabla \mathbb{E}_{\mathbf{X},H}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)]$ by conditioning on H. For a given fixed set of crashing decision $\boldsymbol{\theta}$ and $\boldsymbol{\epsilon}$, the activity starting times of all activities without disruption can be

calculated using Equations (6.6a) or (6.6b). The activity starting times are denoted by (vector) $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) = [t_0, t_1, t_2, ..., t_{N_D}]$. Reorder the element of $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ in ascending order so that $t^{(k)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ is the k^{th} smallest element of $\mathbf{t}$. We define the event $E^{(m)} = \{t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})\}$ for $1 \leq m \leq N_D - 1$, and $E^{(N_D)} = \{t^{(N_D-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H\}$. By conditioning on $E^{(m)}$ and unconditioning, we rewrite Equation (6.4b) as:

$$g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) = \sum_{m=0}^{N_D} \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]\mathbb{P}(E^{(m)}) \tag{6.4c}$$

and for θ_i^j and ϵ_i such that $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ is differentiable, the gradient of $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ is calculated as:

$$\begin{aligned}
&\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \theta_i^j} \\
&= \sum_{m=1}^{N_D} \Bigg\{ \frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]}{\partial \theta_i^j} \mathbb{P}(E^{(m)}) \\
&\quad + \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}] \Big(f_H(t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})) \frac{\partial t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})}{\partial \theta_i^j} - f_H(t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})) \frac{\partial t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})}{\partial \theta_i^j} \Big) \Bigg\},
\end{aligned} \tag{6.13a}$$

$$\begin{aligned}
&\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \epsilon_i} \\
&= \sum_{m=1}^{N_D} \Bigg\{ \frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]}{\partial \epsilon_i} \mathbb{P}(E^{(m)}) \\
&\quad + \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}] \Big(f_H(t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})) \frac{\partial t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})}{\partial \epsilon_i} - f_H(t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})) \frac{\partial t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})}{\partial \epsilon_i} \Big) \Bigg\}.
\end{aligned} \tag{6.13b}$$

Notice that Equations (6.13a) and (6.13b) are valid for a given feasible θ_i^j and ϵ_i if small perturbation of θ_i^j and ϵ_i does not change the order of elements of $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, which is true if there are no ties

in $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$. If there are ties in $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, only a one-sided gradient exists for Equation (6.13a) and (6.13b). To make Equation (6.13a) valid for estimating the one sided gradient $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \theta_i^{j-}}$, the elements in $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ that have the same value need to be ordered in a way such that $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[n]} \leq \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[m]}$, if $\frac{\partial \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[n]}}{\theta_i^{j-}} \leq \frac{\partial \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[m]}}{\theta_i^{j-}}$, the same for $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \theta_i^{j+}}$, $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \epsilon_i^{+}}$, and $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \epsilon_i^{-}}$. In Equations (6.13a) and (6.13b), there are three parts requiring Monte Carlo simulation to estimate:

$$\frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) | t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})]}{\partial \theta_i^j}, \tag{6.14a}$$

$$\frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) | t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})]}{\partial \epsilon_i}, \tag{6.14b}$$

and

$$\mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) | t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})]. \tag{6.15}$$

6.3.2 Stochastic Gradient Estimator

Conditioning on $t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, we have that Problem (6.8) becomes:

$$f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) = \min \quad \tilde{t}_{N_D} \tag{P.0}$$

$$\text{s.t.} \quad \tilde{t}_k - \tilde{t}_i \geq (d_i + X_i)(1 - \sum_{j \in J_i} e_i^j \tilde{\theta}_i^j) \qquad \forall (i,k) \in \mathcal{A}_1^H \tag{P.0a}$$

$$\tilde{t}_k - \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} \geq (d_i + X_i)(1 - \sum_{j \in J_i} e_i^j \tilde{\theta}_i^j) \quad \forall (i,k) \in \mathcal{A}_2^H \tag{P.0b}$$

$$\tilde{t}_k - \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} \geq d_i(1 - \sum_{j \in J_i} e_i^j \theta_i^j) \qquad \forall (i,k) \in \mathcal{A}_3^H \tag{P.0c}$$

$$\sum_{i \in I_1^H} \sum_{j \in J_i} b_i^j \tilde{\theta}_i^j + \sum_{i \in \mathcal{N}_2^H} \sum_{j \in J_i} b_i^j \theta_i^j \leq B \tag{P.0d}$$

$$\sum_{j\in J_i}\tilde{\theta}_i^j \le 1 \qquad \forall i \in \mathcal{N}_1^H \qquad \text{(P.0e)}$$

$$\tilde{\theta}_i^j \le \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}_1^H, \forall j \in J_i \qquad \text{(P.0f)}$$

$$\tilde{\theta}_i^j \ge 0 \qquad \forall i \in \mathcal{N}_1^H, \forall j \in J_i \qquad \text{(P.0g)}$$

$$\tilde{t}_i \ge 0 \qquad \forall i \in \mathcal{N}_1^H \qquad \text{(P.0h)}$$

Transforming Problem (P.0) into canonical form, we have:

$$-\max \quad -\tilde{t}_{N_D} \qquad \text{(P.1)}$$

$$\text{s.t.} \quad -\tilde{t}_k + \tilde{t}_i - (d_i + X_i)\sum_{j\in J_i} e_i^j \tilde{\theta}_i^j \le -d_i - X_i \qquad \forall (i,k) \in \mathcal{A}_1^H \qquad \text{(P.1a)}$$

$$-\tilde{t}_k - (d_i + X_i)\sum_{j\in J_i} e_i^j \tilde{\theta}_i^j \le -\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} - d_i - X_i \qquad \forall (i,k) \in \mathcal{A}_2^H \qquad \text{(P.1b)}$$

$$-\tilde{t}_k \le -\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i \sum_{j\in J_i} e_i^j \theta_i^j - d_i \qquad \forall (i,k) \in \mathcal{A}_3^H \qquad \text{(P.1c)}$$

$$\sum_{i\in\mathcal{N}_1^H}\sum_{j\in J_i} b_i^j \tilde{\theta}_i^j \le B - \sum_{i\in\mathcal{N}_2^H}\sum_{j\in J_i} b_i^j \theta_i^j \qquad \text{(P.1d)}$$

$$\sum_{j\in J_i}\tilde{\theta}_i^j \le 1 \qquad \forall i \in \mathcal{N}_1^H \qquad \text{(P.1e)}$$

$$\tilde{\theta}_i^j \le \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}_1^H, \forall j \in J_i \qquad \text{(P.1f)}$$

$$\tilde{\theta}_i^j \ge 0 \qquad \forall i \in \mathcal{N}_1^H, \forall j \in J_i \qquad \text{(P.1g)}$$

$$\tilde{t}_i \ge 0 \qquad \forall i \in \mathcal{N}_1^H \qquad \text{(P.1h)}$$

Assuming the regularity conditions for exchanging derivative and expectation are satisfied for Equations (6.14a) and (6.14b), our next goal is to estimate:

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \theta_i^j} \quad \text{and} \quad \frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \epsilon_i} \tag{6.16}$$

given that $t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})$. Although under the condition that $t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})$, we do not have a closed form for the function $f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)$, we can still estimate its derivative. Notice that $f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)$ in Problem (P.1) is a function whose output is the optimal value of a linear programming and whose input are the parameters of the RHS of the constraints of the LP. From [42], the sensitivity analysis of RHS of LP constraints, we have that for a small perturbation of the RHS of a constraint of in an LP problem, its optimal solution's change equals the perturbation times its corresponding shadow price.

Notice that $\frac{\partial \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[k]}}{\partial \theta_i^j}$ may not exist if there are ties of the length of paths that start from source node and end at node k, i.e., two paths in $\mathcal{P}_k$ have the same length and activity i only appears on one path. As a result of that, only the following two one-sided gradient always exist

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \theta_i^{j\pm}} \quad \text{and} \quad \frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \epsilon_i^{\pm}}. \tag{6.17}$$

Notice that for a given $m \geq 1$, when $t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})$, $f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)$ does not depend on H. Also notice that there is only one element in $\mathcal{A}_2^H$. For m fixed and $t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) <$

$H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, we have the gradient estimators of (6.17) given by:

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)}{\partial \theta_i^{j\pm}} = \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']}}{\partial \theta_i^{j\pm}} y^*_{n_t(i',k')} - \sum_{\substack{(i',k') \in \mathcal{A}_3^H \\ i' = i}} d_i e_i^j y^*_{n_t(i',k')} + b_i^j y^*_{N_c}, \tag{6.18A}$$

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)}{\partial \epsilon_i^{\pm}} = \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']}}{\partial \epsilon_i^{\pm}} y^*_{n_t(i',k')}, \tag{6.18B}$$

where n_t is a mapping such that $n_t(i,j)$ is the constraint number in (P.1a) and (P.1b) that corresponds to arc (i,j), N_c is the constraint numbers associated with (P.1c) in Problem (P.1), and $y^*_{n_t(i,j)}$ and $y^*_{N_c}$ are the optimal solutions of the dual problem of (P.1) corresponding to the $n_t(i,j)^{th}$ and N_c^{th} constraints. Next we provide a theorem showing the unbiasedness of the above gradient estimators.

Theorem 3.

$$\begin{aligned}
&\frac{\partial \mathbb{E}[f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H) | t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})]}{\partial \theta_i^{j\pm}} \\
=&\mathbb{E}\Big[\frac{\partial f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H)}{\partial \theta_i^{j\pm}} \Big| t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})\Big], \\
&\frac{\partial \mathbb{E}[f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H) | t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})]}{\partial \epsilon_i^{\pm}} \\
=&\mathbb{E}\Big[\frac{\partial f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H)}{\partial \epsilon_i^{\pm}} \Big| t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})\Big],
\end{aligned}$$

where $\frac{\partial f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H)}{\partial \theta_i^{j\pm}}$ *and* $\frac{\partial f(\boldsymbol{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \boldsymbol{X}, H)}{\partial \epsilon_i^{\pm}}$ *are given by Equations (6.18A) and (6.18B).*

6.3.3 Proof of Unbiased Gradient Estimator

Two lemmas are first presented and prepared for the proof of Theorem 3. Recall from the previous section that y_i is the dual variable that corresponds to the i^{th} constraint of Problem (P.1). In Problem (P.1), let $N_1 = |\mathcal{A}_1^H|$, $N_2 = |\mathcal{A}_2^H|$ and $N_3 = |\mathcal{A}_3^H|$. Also create a mapping n_t such that $n_t(i,j)$ is the constraint number in (P.1a) and (P.1b) that corresponds to arc (i,j). Before the proof of unbiasedness, we first present two lemmas, Lemma 3 and Lemma 4. Lemma 3 states that for any feasible solutions of the dual problem of (P.1), the first $N_1 + N_2 + N_3$ dual variables are all less than or equal to 1.

Lemma 3. $y_i \leq 1, \quad \forall 1 \leq i \leq N_1 + N_2 + N_3$.

Proof. Proof of Lemma 3. The dual constraints for all columns of the LHS of Problem (P.1) that includes $\{\tilde{t}_i\}$ are given by:

$$\sum_{(i,N_D)\in\cup_{m=1}^{3}\mathcal{A}_m^H} y_{n_t(i,N_D)} \leq 1, \tag{L.2a}$$

$$\sum_{(i,j)\in\cup_{m=2}^{3}\mathcal{A}_m^H} y_{n_t(i,j)} \leq \sum_{(j,k)\in\mathcal{A}_1^H} y_{n_t(j,k)}, \quad \forall j \in \{i|(i,k)\in\mathcal{A}_1^H\}, \tag{L.2b}$$

Constraint (L.2b) indicates that for each activity in the network whose starting time is affected (delayed) by the disruption, the sum of the dual variable of their incoming arcs are no larger than the sum of outcoming arcs. Constraint (L.2a) indicates that the sum of dual variables of the incoming arcs of the sink node is no larger than 1. Since all dual variables are non-negative, we have that the first $N_1 + N_2 + N_3$ dual variables of problem (P.1) are all between 0 and 1.

□

Next we provide Lemma 4 stating that the optimal $N_c = (N_1+N_2+N_3+1)^{th}$ dual variable (the dual variable corresponding to the budget constraint) is bounded by an integrable function of $\mathbf{X}$. Before the proof of Lemma 4, we first create two mappings n_d and n_e such that activity i is the $n_d(i)^{th}$ constraint in (P.1d) and option j of activity i is the $n_e(i,j)^{th}$ constraint in (P.1e).

Lemma 4. $y^*_{N_c} \leq h_c(\boldsymbol{X})$, $\mathbb{E}[h_c(\boldsymbol{X})] < +\infty$.

Proof. Proof of Lemma 4. Let $\mathcal{O}_1^H = \{(i,j)|i \in \mathcal{N}_1^H, j \in J_i\}$ and $N_d = |\mathcal{N}_1^H|$. The constraints of the dual problem of (P.1) that corresponding to $\{\tilde{\theta}_i^j\}$ are given by:

$$
\begin{aligned}
&-(d_i+X_i)e_i^j \sum_{(i,k)\in\mathcal{A}_1^H\cup\mathcal{A}_2^H} y_{n_t(i,k)} + b_i^j y_{N_c} + y_{N_c+n_d(i)} + y_{N_c+N_d+n_e(i,j)} \geq 0, \quad \forall (i,j) \in \mathcal{O}_1^H \\
\Leftrightarrow & y_{N_c} \geq \frac{1}{b_i^j}\Big((d_i+X_i)e_i^j \sum_{(i,k)\in\mathcal{A}_1^H\cup\mathcal{A}_2^H} y_{n_t(i,k)} - y_{N_c+n_d(i)} - y_{N_c+N_d+n_e(i,j)}\Big), \quad \forall (i,j) \in \mathcal{O}_1^H \\
\Leftrightarrow & y_{N_c} \geq \max_{(i,j)\in\mathcal{O}_1^H}\Big\{\frac{1}{b_i^j}\Big((d_i+X_i)e_i^j \sum_{(i,k)\in\mathcal{A}_1^H\cup\mathcal{A}_2^H} y_{n_t(i,k)} - y_{N_c+n_d(i)} - y_{N_c+N_d+n_e(i,j)}\Big)\Big\}
\end{aligned}
$$

since the RHS of constraint (P.1d) is positive, the optimal dual solution $y^*_{n_c}$ that minimized the dual objective function satisfies the following inequalities:

$$
\begin{aligned}
y^*_{n_c} &= \max\Big(\max_{(i,j)\in\mathcal{O}_1^H}\Big\{\frac{1}{b_i^j}\Big((d_i+X_i)e_i^j \sum_{(i,k)\in\mathcal{A}_1^H\cup\mathcal{A}_2^H} y^*_{n_t(i,k)} - y^*_{N_c+2+n_d(i)} - y^*_{N_c+2+N_d+n_e(i,j)}\Big)\Big\}, 0\Big) \\
&\leq \max_{(i,j)\in\mathcal{O}_1^H}\Big\{\frac{1}{b_i^j}(d_i+X_i)e_i^j \sum_{(i,k)\in\mathcal{A}_1^H\cup\mathcal{A}_2^H} y^*_{n_t(i,k)}\Big\} \\
&\leq \max_{(i,j)\in\mathcal{O}_1^H}\Big\{\frac{|\mathcal{A}_1^H|+|\mathcal{A}_2^H|}{b_i^j}(d_i+X_i)e_i^j\Big\} \\
&\leq \sum_{(i,j)\in\mathcal{O}_1^H}\Big\{\frac{|\mathcal{A}_1^H|+|\mathcal{A}_2^H|}{b_i^j}(d_i+X_i)e_i^j\Big\} = h_c(\mathbf{X})
\end{aligned}
$$

It is obvious that $\mathbb{E}(h_c(\mathbf{X})) < +\infty$ under the assumption that $\{X_i\}$ have finite first moments.

□

Next is the proof for Theorem 3.

Proof. Proof of Theorem 3. Without loss of generality, we only prove the first case of Theorem 3. For (i, j) and $\mathbf{X}$ fixed, and $t^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, there exists $N > 0$, such that when $n > N$, we have:

$$
\begin{aligned}
&n|f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) - f(\mathbf{t}(\boldsymbol{\theta} - \Delta_1, \boldsymbol{\epsilon}), \boldsymbol{\theta} - \Delta_1, \mathbf{X}, H)| \\
=&n\Big| \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} (\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} - \mathbf{t}(\boldsymbol{\theta} - \Delta_1, \boldsymbol{\epsilon})_{[i']}) y^*_{n_t(i',k')} - \sum_{(i',k') \in \mathcal{A}_3^H} \frac{1}{n} I\{i' = i\} d_i e_i^j y^*_{n_t(i',k')} + \frac{1}{n} b_i^j y^*_{n_c} \Big| \\
\leq& \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \left| e_i^j y^*_{n_t(i',k')} \right| + \sum_{(i',k') \in \mathcal{A}_3^H} |I\{i' = i\} d_i e_i^j y^*_{n_t(i',k')}| + |b_i^j y^*_{n_c}| \\
\leq& (|\mathcal{A}_1^H| + |\mathcal{A}_2^H| + |\mathcal{A}_3^H| d_i) e_i^j + b_i^j h_c(\mathbf{X}) = h_P(\mathbf{X}),
\end{aligned}
$$

and similarly, we also have

$$
\begin{aligned}
&n|f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) - f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon} - \Delta_2), \boldsymbol{\theta}, \mathbf{X}, H)| \\
=&n \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \left| (\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} - \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon} - \Delta_2)_{[i']}) y^*_{n_t(i',k')} \right| \\
\leq& (|\mathcal{A}_1^H| + |\mathcal{A}_2^H|) \left| e_i^j y^*_{n_t(i',k')} \right| = (|\mathcal{A}_1^H| + |\mathcal{A}_2^H|) e_i^j,
\end{aligned}
$$

where $\Delta_1 = [0, ..., \frac{1}{n}, ..., 0]$, a vector whose elements are all zero except for the element corresponding to θ_i^j and $\Delta_2 = [0, ..., \frac{1}{n}, ..., 0]$, a vector whose elements are all zero except for i^{th} element.

Let $\mathbb{E}_{X|H^m}[\cdot] = \mathbb{E}[\cdot|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})]$. Since $\mathbb{E}[|h_P(\mathbf{X})|] < +\infty$, as a result of Lebesgue Dominated Convergence theorem [43], we have

$$\begin{aligned}
&\lim_{n\to+\infty} n\mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H) - f(\mathbf{t}(\boldsymbol{\theta}-\Delta_1,\boldsymbol{\epsilon}),\boldsymbol{\theta}-\Delta_1,\mathbf{X},H)|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})] \\
=&\mathbb{E}_{X|H^m}\Big[\lim_{n\to+\infty} n\Big(\sum_{\substack{(i',k')\in\mathcal{A}_2^H\cup\mathcal{A}_3^H \\ i'>i}} (\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} - \mathbf{t}(\boldsymbol{\theta}-\Delta_1,\boldsymbol{\epsilon})_{[i']})y^*_{n_t(i',k')} - \\
&\qquad \sum_{(i',k')\in\mathcal{A}_3^H} \frac{1}{n} I\{i'=i\} d_i e_i^j y^*_{n_t(i',k')} + \frac{1}{n} b_i^j y^*_{n_c}\Big)\Big] \\
=&\mathbb{E}_{X|H^m}\Big[\sum_{\substack{(i',k')\in\mathcal{A}_2^H\cup\mathcal{A}_3^H \\ i'>i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']}}{\partial \theta_i^{j-}} y^*_{n_t(i',k')} - \sum_{(i',k')\in\mathcal{A}_3^H} I\{i'=i\} d_i e_i^j y^*_{n_t(i',k')} + b_i^j y^*_{N_c}\Big],
\end{aligned}$$

and

$$\begin{aligned}
&\lim_{n\to+\infty} n\mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H) - f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}-\Delta_2),\boldsymbol{\theta},\mathbf{X},H)|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})] \\
=&\mathbb{E}_{X|H^m}\Big[\lim_{n\to+\infty} n \sum_{\substack{(i',k')\in\mathcal{A}_2^H\cup\mathcal{A}_3^H \\ i'>i}} \Big((\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']} - \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}-\Delta_2)_{[i']})y^*_{n_t(i',k')}\Big)\Big] \\
=&\mathbb{E}_{X|H^m}\Big[\sum_{\substack{(i',k')\in\mathcal{A}_2^H\cup\mathcal{A}_3^H \\ i'>i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon})_{[i']}}{\partial \theta_i^{j-}} y^*_{n_b}\Big],
\end{aligned}$$

which are equivalent to

$$\begin{aligned}
&\frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})]}{\partial \theta_i^{j-}} \\
=&\mathbb{E}\Big[\frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \theta_i^{j-}}\Big|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})\Big], \\
&\frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})]}{\partial \epsilon_i^{-}} \\
=&\mathbb{E}\Big[\frac{\partial f(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta},\mathbf{X},H)}{\partial \epsilon_i^{-}}\Big|t^{(m-1)}(\boldsymbol{\theta},\boldsymbol{\epsilon}) < H \leq t^{(m)}(\boldsymbol{\theta},\boldsymbol{\epsilon})\Big],
\end{aligned}$$

and similarly, we also have the same results for right hand side derivatives.

□

6.4 Stochastic Gradient-Based Optimization

Notice that Problem (6.7S) is an optimization problem with objective function differentiable almost everywhere whose domain is open and with linear constraints. Let $N_D = |\mathcal{N}|$, $P = \{(i,j)|i \in \mathcal{N}, j \in J_i\}$, $N_P = |\{(i,j)|i \in \mathcal{N}, j \in J_i\}|$, and $n_S(i,j)$ is the number of ordered element in P. The KKT condition for Problem (6.7S) is given by

$$\begin{aligned}
\sum_{j\in J_i} \theta_i^j - 1 \leq 0, \qquad & \forall i \in \mathcal{N} \\
\theta_i^j - \bar{\theta}_i^j \leq 0, \qquad & \forall i \in \mathcal{N}, j \in J_i \\
-\theta_i^j \leq 0, \qquad & \forall i \in \mathcal{N}, j \in J_i \\
-\epsilon_i \leq 0, \qquad & \forall i \in \mathcal{N} \\
\sum_{i\in\mathcal{N}}\sum_{j\in J_i} b_i^j \theta_i^j = B &
\end{aligned}$$

$$\lambda_i \geq 0, \qquad i = 1, ..., 2(N_D + N_P)$$

$$\lambda_i(\sum_{j \in J_i} \theta_i^j - 1) = 0, \qquad i = 1, 2, ..., N_D$$

$$(\theta_i^j - \bar{\theta}_i^j)\lambda_{N_D + n_S(i,j)} = 0, \qquad \forall i \in \mathcal{N}, j \in J_i$$

$$\theta_i^j \lambda_{N_D + N_P + n_S(i,j)} = 0, \qquad \forall i \in \mathcal{N}, j \in J_i$$

$$\epsilon_i \lambda_{2N_D + N_P + i} = 0, \qquad i = 1, ..., N_D$$

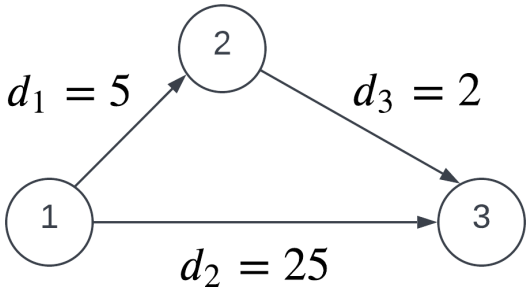
together with the gradient condition

$$\nabla g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) + \sum_{i=1}^{N_D} \lambda_i \nabla(\sum_{j \in J_i} \theta_i^j - 1) + \sum_{(i,j) \in P} \lambda_{N_D + n_S(i,j)} \nabla(\theta_i^j - \bar{\theta}_i^j)$$
$$- \sum_{(i,j) \in P} \lambda_{N_D + N_P + n_S(i,j)} \nabla \theta_i^j - \sum_{i=1}^{N_D} \lambda_{2N_D + N_P + i} \nabla \epsilon_i + \nu \nabla(\sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j \theta_i^j - B) = 0.$$

From the KKT condition, we have that a feasible and differentiable $(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ and its dual optimal solution $(\boldsymbol{\lambda}, \nu)$ satisfy the KKT condition if and only if: (1) For all $\theta_{i_1}^{j_1}$ and $\theta_{i_2}^{j_2}$ nonzero and can be increased, $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \theta_{i_1}^{j_1}} = \frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \theta_{i_2}^{j_2}}$; (2) For all $\epsilon > 0$, $\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \epsilon_i} = \lambda_{2N_D + N_P + i}$. The KKT condition is a necessary and sufficient condition for a stationary point. And if the objective is convex, every stationary point is a global optimum, and as a result the KKT condition is a necessary and sufficient condition for an optimal solution [29] . However, the objective function $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ in Problem (6.7S) is not convex. A nonconvex example is provided in Figure 6.1.

Figure 6.1 is an activity network with three activities and three nodes. The duration of each activities are provided on the graph as $d_1 = 5$, $d_2 = 25$ and $d_3 = 2$. The disruption occurrence time H follows uniform distribution $H \sim U(3 - 0.001, 3 + 0.001)$. The delay of activity 3 is $X_3 = 10,000$ with probability 1. The per unit crashing all equal 1, i.e., $e_1 = e_2 = e_3 = 1$,

Figure 6.1: Nonconvex Activity Network



and the unit cost are all equal to 1, $b_1 = b_2 = b_3 = 1$. The crashing upper bounds are given by $\bar{\theta}_1 = \bar{\theta}_2 = \bar{\theta}_3 = 0.5$. The total budget is $B = 0.5$. The probability that disruption will not happen is $p = 0.1$ and $\alpha = 0.9$. We have two crashing decisions $(\boldsymbol{\theta}^{(1)}, \boldsymbol{\epsilon}^{(1)})$ and $(\boldsymbol{\theta}^{(2)}, \boldsymbol{\epsilon}^{(2)})$ given by:

$$\boldsymbol{\theta}^{(1)} = [0.41, 0.02, 0.07]$$

$$\boldsymbol{\theta}^{(2)} = [0.3, 0.01, 0.19]$$

$$\alpha\boldsymbol{\theta}^{(1)} + (1-\alpha)\boldsymbol{\theta}^{(2)} = [0.399, 0.019, 0.082]$$

$$\boldsymbol{\epsilon}^{(1)} = \boldsymbol{\epsilon}^{(2)} = \mathbf{0}$$

and the objective function for this example is given by:

$$\begin{aligned} g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) =& p * \max\{(1-\theta_1)d_1 + (1-\theta_2)d_2, (1-\theta_3)d_3\} + \\ & (1-p)\Big(\max\{(1-\theta_1)d_1 + (1-\theta_2)d_2, (1-\theta_3)d_3\}\mathbb{P}(H \geq (1-\theta_1)d_1) \\ & + \inf_{\theta_1+\theta_2+\tilde{\theta}_3=B} \{\max\{(1-\theta_1)d_1 + (1-\theta_2)d_2, (1-\tilde{\theta}_3)(d_3+X_3)\}\}\mathbb{P}(H < (1-\theta_1)d_1)\Big) \end{aligned}$$

Then we have

$$
\begin{aligned}
& g(\mathbf{t}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\epsilon}^{(1)}), \boldsymbol{\theta}^{(1)}) \\
=& 0.01 * 24.5 + 0.99 * 24.5 = 24.5 \\
& g(\mathbf{t}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\epsilon}^{(2)}), \boldsymbol{\theta}^{(2)}) \\
=& 0.01 * 24.75 + 0.99 * 9304.81 = 9212.0094 \\
& g(\mathbf{t}(\alpha\boldsymbol{\theta}^{(1)} + (1-\alpha)\boldsymbol{\theta}^{(2)}, \alpha\boldsymbol{\epsilon}^{(1)} + (1-\alpha)\boldsymbol{\epsilon}^{(2)}), \alpha\boldsymbol{\theta}^{(1)} + (1-\alpha)\boldsymbol{\theta}^{(2)}) \\
=& 0.01 * 24.525 + 0.99 * 9184.841 = 9093.23784 \\
& \alpha g(\mathbf{t}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\epsilon}^{(1)}), \boldsymbol{\theta}^{(1)}) + (1-\alpha) g(\mathbf{t}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\epsilon}^{(2)}), \boldsymbol{\theta}^{(2)}) \\
=& 0.9 * 24.5 + 0.1 * 9212.0094 = 943.25094,
\end{aligned}
$$

as a result of which we have

$$
\begin{aligned}
& g(\mathbf{t}(\alpha\boldsymbol{\theta}^{(1)} + (1-\alpha)\boldsymbol{\theta}^{(2)}, \alpha\boldsymbol{\epsilon}^{(1)} + (1-\alpha)\boldsymbol{\epsilon}^{(2)}), \alpha\boldsymbol{\theta}^{(1)} + (1-\alpha)\boldsymbol{\theta}^{(2)}) \\
>& \alpha g(\mathbf{t}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\epsilon}^{(1)}), \boldsymbol{\theta}^{(1)}) + (1-\alpha) g(\mathbf{t}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\epsilon}^{(2)}), \boldsymbol{\theta}^{(2)}).
\end{aligned}
$$

In conclusion, $g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})$ is not convex. Since we have unbiased estimators for one-sided gradients of the objective function of Problem (6.7S), a heuristic gradient-based algorithm for solving Problem (6.7S) is proposed in Algorithms 5 and 6. In Algorithms 5 and 6, $\mathcal{O} = \{(i,j) | i \in \mathcal{N}, j \in J_i\}$. The gradients in Algorithm 5 are estimated by Monte Carlo simulation with N_1 simulation replications and the gradients in Algorithm 6 are estimated by Monte Carlo simulation with N_2 simulation. For phase 1, at each step, we choose the option with largest gradient descend

and increase the crashing option parameter by a given amount. Phase 1 stops when the budget is exhausted. For phase 2, at each step, we choose the option that decreases the objective function the most and the option that decreases the objective function the least among all the options that are not at their boundaries. And we redistribute the budget among the previous two options by a given amount. At each step of phase 2, a decreasing of the objective function is guaranteed. Phase 2 stops after M iterations, where M is a positive integer inputed as a hyper-parameter.

Algorithm 5: Phase 1 Optimization

Input : N_1, α_1, λ
Output: $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow \mathbf{0}$
$\boldsymbol{\epsilon} \leftarrow \mathbf{0}$
while $B > 0$ **do**
  (i,j) $\leftarrow \underset{\substack{(i,j)\in\mathcal{O} \\ \theta_i^j < \bar{\theta}_i^j}}{\arg\min}\{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \theta_i^{j+}}/b_i^j\}$
  $\delta_i^j \leftarrow \alpha_1 \bar{\theta}_i^j$
  $\theta_i^j \leftarrow \theta_i^j + \min(b_i^j \delta_i^j, b_i^j(1 - \sum_{k\in J_i, k\neq j} \theta_i^k), B)/b_i^j$
  B $\leftarrow B - \min(b_i^j \delta_i^j, b_i^j(1 - \sum_{k\in J_i, k\neq j} \theta_i^k), B)$
  $\epsilon_i \leftarrow \max\{0, \epsilon_i - \lambda \frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \epsilon_i^+}\}$
end

Algorithm 6: Phase 2 Optimization

Input : N_2, N_d, M, $\boldsymbol{\theta}$
Output: $\boldsymbol{\theta}$

$\delta_d \leftarrow \frac{B}{N_d}$
n $\leftarrow 0$
while $n < M$ **do**
  $(i_1, j_1) \leftarrow \underset{\substack{(i,j)\in\mathcal{O}\\ 0<\theta_i^j<\bar{\theta}_i^j}}{\arg\min}\{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \theta_i^{j-}}/b_i^j\}$
  $(i_2, j_2) \leftarrow \underset{\substack{(i,j)\in\mathcal{O}\\ 0<\theta_i^j<\bar{\theta}_i^j}}{\arg\max}\{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \theta_i^{j-}}/b_i^j\}$
  $\delta_b \leftarrow \min(\delta_d, b_{i_2}^{j_2}\theta_{i_2}^{j_2}, b_{i_1}^{j_1}(1 - \sum_{k\in J_{i_1}, l\neq j_1}\theta_{i_1}^k))$
  $\theta_{i_1}^{j_1} \leftarrow \theta_{i_1}^{j_1} + \delta_b/b_{i_1}^{j_1}$
  $\theta_{i_2}^{j_2} \leftarrow \theta_{i_2}^{j_2} - \delta_b/b_{i_2}^{j_2}$
  $\epsilon_i \leftarrow \max\{0, \epsilon_i - \lambda\frac{\partial g(\mathbf{t}(\boldsymbol{\theta},\boldsymbol{\epsilon}),\boldsymbol{\theta})}{\partial \epsilon_i^+}\}$
  $n \leftarrow n + 1$
end

6.5 Numerical Experiments

Both algorithm2 in [37], called SAA, and our gradient-based algorithm, called SGD, are tested on five acvitity networks named case 11, 14, 19, 19a and 35, where all except case 19a are from [37]. Experimental data and AN structures of the five ANs are provided in Appendix B. The disruption occurrence time follows a log-normal distribution. All activity random delays are independent exponentially distributed with known mean values. The probability that $H = +\infty$ is 0.1, where $H = +\infty$ means the disruption will not happen.

For the SGD algorithm, in order to find the appropriate batch sizes N_1 and N_2 (number of simulation replications) for estimating the gradient, the SGD algorithm is tested on four ANs named case 11, 14, 19 and 19a with the value of $N_1 = N_2$ equal to 1, 5, 10, 20, 50 and 100. For each batch size $N_1 = N_2$, the SGD algorithm is run 20 times to give 20 independent

Figure 6.2: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 11 Activities (based on 20 runs) of different SGD batch sizes

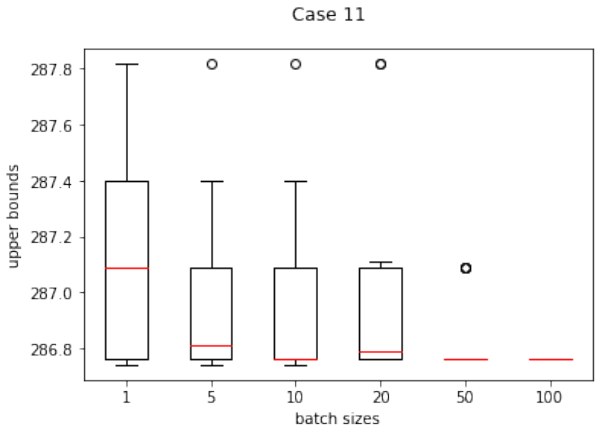
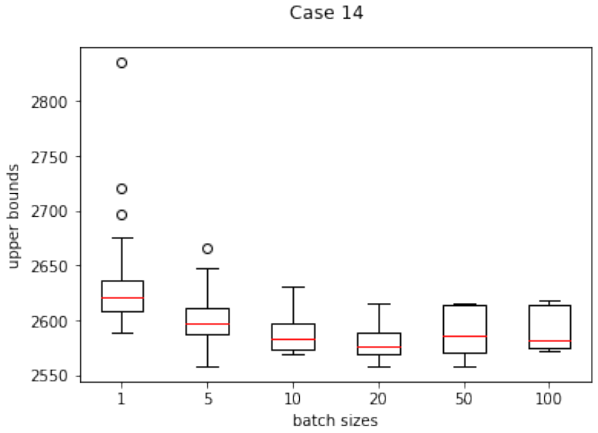


Figure 6.3: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 14 Activities (based on 20 runs) of different SGD batch sizes



optimal solutions that are tested on 5000 fixed randomly generated samples and calculated by Equation (6.4c). For the AN with 35 activities, the choices of $N_1 = N_2$ are 1, 5, 10 and 20. Boxplots of the estimated objective function values are provided in Figures 6.2 - 6.6, from which we can conclude that increasing the batch size can make the estimated objective solution more stable (corresponding minimum objective values close to each other for different runs). For case 19b, increasing the batch size decreases the sample variance and range of the upper bound but increases the median of it. It is believed that for case 19b the SGD algorithm finally converge to

Figure 6.4: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 19 Activities (based on 20 runs) of different SGD batch sizes

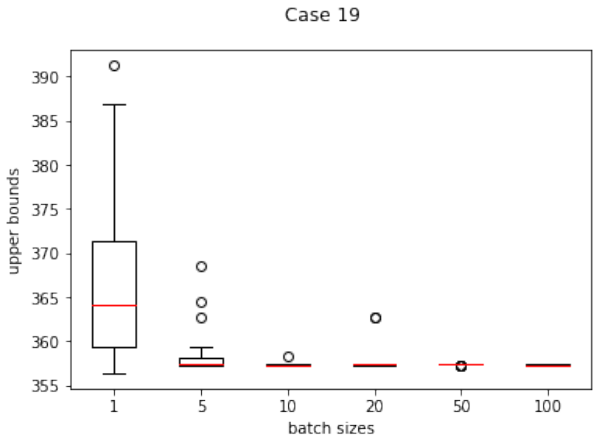
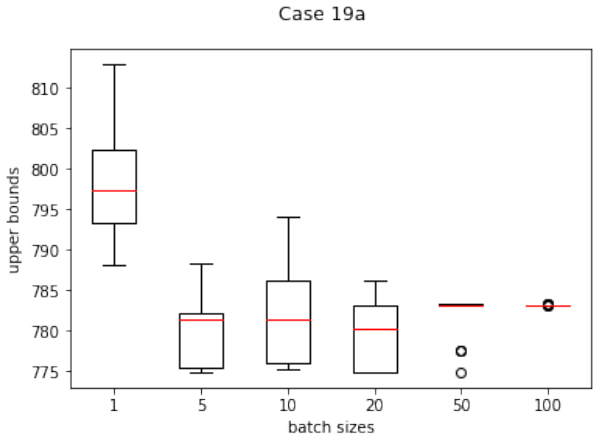
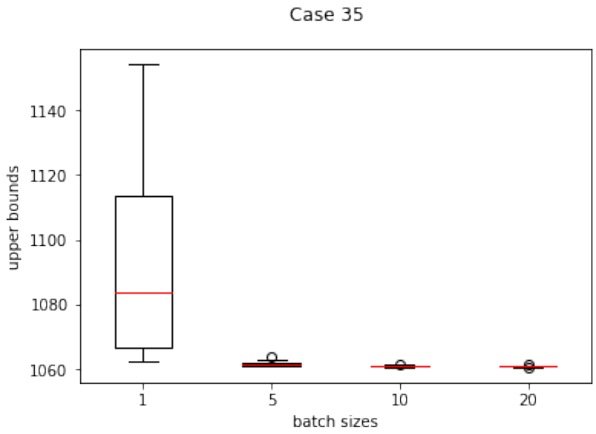


Figure 6.5: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 19a Activities (based on 20 runs) of different SGD batch sizes



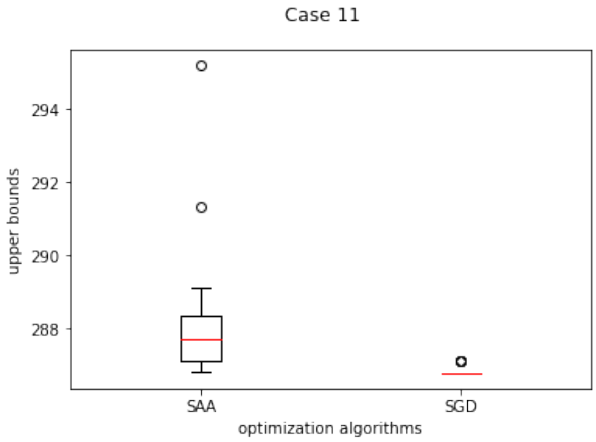
a local minimum instead of the global minimum. Thus, we conclude that for the SGD algorithm, larger batch size is not necessarily better, as small batch size allows more randomness of the descent direction and can avoid converging to the local optimum. Also, for both SAA and SGD algorithm, in order to achieve a better solution, it is necessary to have multiple independent runs of the same algorithm and have their output solutions tested on the simulated objective function value performance instead of having a single run. From Figures 6.2 - 6.6 we can also conclude that a batch size of 10 is good enough for case 35, a batch size of 20 is good enough for case 14

Figure 6.6: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 35 Activities (based on 20 runs) of different SGD batch sizes



and 19b, and a batch size of 50 is good enough for case 11 and 19.

Figure 6.7: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 11 Activities (based on 20 runs with 50 batch size)



Figures 6.7 - 6.11 are the comparison between the SAA and SGD algorithm in finding the optimal solution for the five AN examples. For each network, the SAA algorithm is run 20 times with 500 simulation replications each used to estimate the optimal solution. The hyper-parameters are the same as in [37]. The computation hardware for SAA is Intel Xeon CPU with 30 cores (each core 3.1 Ghz) and 120GB RAM. The computation hardware for SGD is Intel Core i5-8279U CPU with 4 cores (each core 2.4 Ghz) and 8GB RAM. The SAA algorithm is

Figure 6.8: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 14 Activities (based on 20 runs with 20 batch size)

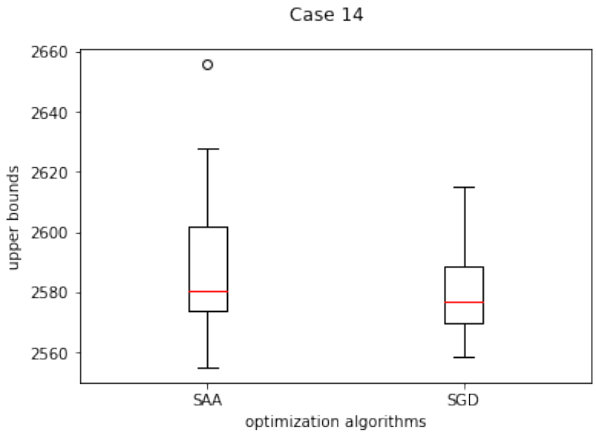
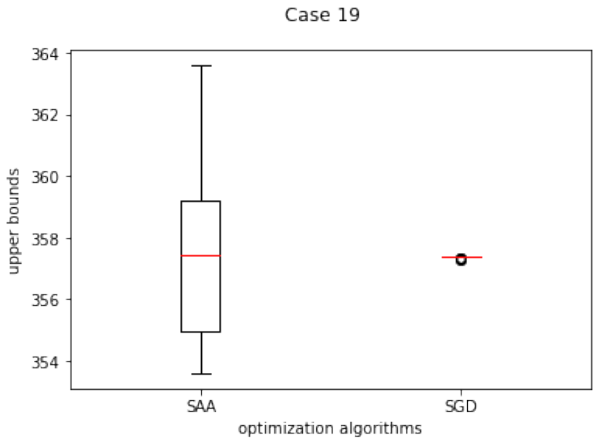


Figure 6.9: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 19 Activities (based on 20 runs with 50 batch size)



constructed using version 0.18.0 of the JuMP package [44] on the Julia platform using the code provided by Yang [37]. The SGD algorithm is constructed using version 3.6 of Python. All linear programming problems are solved by version 8.0.1 of Gurobi [45]. For the SAA algorithm, each node is solved by 6 cores and we allow at most 5 nodes to be solved simultaneously, so that the maximum of cores used at any time is 30 [37]. For the SGD algorithm, only a single core is used, and no parallel computing is applied. SGD is run 20 times with $N_1 = 100, \alpha_1 = 0.5, N_2 = 100, N_d = 10 * B, M = 20$. For both SAA and SGD, we have 20 independent optimal solutions.

Figure 6.10: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 19a Activities (based on 20 runs with 20 batch size)

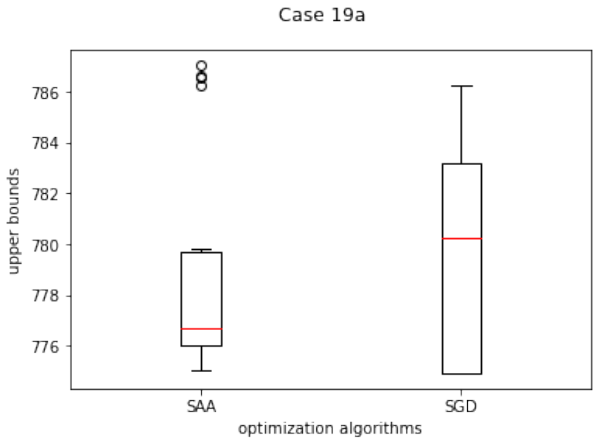
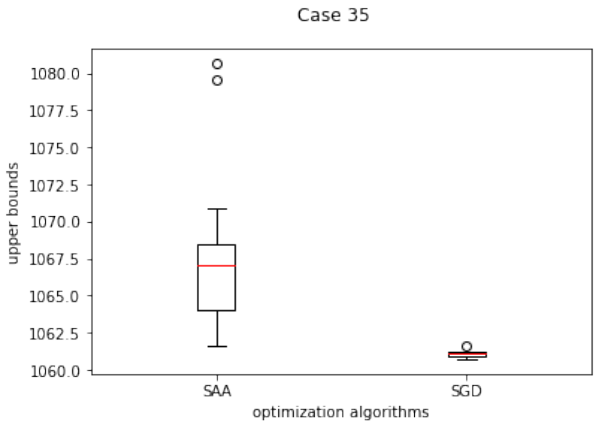


Figure 6.11: Boxplots (with red line the median) of Estimated Minimum Objective Function Value for Activity Network with 35 Activities (based on 20 runs with 10 batch size)



For each method, the optimal solutions are tested on 5000 fixed randomly generated samples and calculated by Equation (6.4c), then 20 objective function sample means are calculated and used as an approximation of the true expected project completion time (PCT). In Figures 6.7 to 6.11, the left-hand side boxplot is the boxplot of 20 optimal values of SAA algorithm and right-hand side boxplot is the boxplot of 20 optimal values of the SGD algorithm. Except for Figure 6.10, SGD produces a tighter boxplot with lower mean compared with SAA. As for Figure 6.10 of case 19a, SAA produces a tighter boxplot with lower mean.

Table 6.2: 95% Confidence Interval for Upper Bound of Optimal Value (based on 20 runs)

Algorithm	11 Activities	14 Activities	19 Activities	19a Activities	35 Activities
SGD	286.8 ± 0.25	2580.7 ± 29.84	357.4 ± 0.07	779.7 ± 8.37	1061.1 ± 0.37
SAA	288.2 ± 3.74	2588.7 ± 49.30	357.6 ± 5.72	778.8 ± 8.08	1067.4 ± 9.64

Table 6.3: Paired-t test p-value (based on 20 runs)

	11 Activities	14 Activities	19 Activities	19a Activities	35 Activities
p-value	5.6e-3	0.16	0.69	0.50	2.23e-5

Table 6.2 is the 95 % confidence interval of the heuristic optimal value (upper bound of the optimal value) estimated by the SGD and SAA algorithms. Table 6.3 contains the p-values of the paired t-test between the 20 upper bounds produced by the SGD and SAA algorithms. From Tables 6.2 and 6.3, we can conclude that the SGD algorithm produces an upper bound of the optimal value with lower mean and tighter C.I. compared to the SAA algorithm.

Table 6.4: Averaged Computing Time (in seconds, over 20 runs, standard deviation in parentheses)

Algorithm	11 Activities	14 Activities	19 Activities	19a Activities	35 Activities
SGD	12 (1.3)	14 (0.2)	60 (1.3)	142 (2.3)	580 (23.8)
SAA	53 (25.2)	685 (221.9)	850 (218.3)	116 (36.5)	230 (83.8)

Table 6.4 provides the averaged computing time of SAA and SGD algorithms. The computing times are measured in seconds. Notice that the SGD algorithm has two versions, Activity-on-Node (A-on-N) version (Algorithms 5 and 6) and Activity-on-Arc (A-on-A) version (Algorithms 7 and 8). The A-on-A algorithm is based on the A-on-A representation of ANs, whose details are presented in Appendix A. The A-on-A algorithm is applied for cases 11, 14 and 19, and the A-on-N algorithm is applied for cases 19b and 35. For an AN, the A-on-A algorithm is preferred if its A-on-A representation is simpler (takes less number of nodes and arcs to present) than its

A-on-N representation, vice versa.

Table 6.5: Memory Consumption (over 10 runs, in GB)

Algorithm	11 Activities	14 Activities	19 Activities	19a Activities	35 Activities
SGD	0.08	0.08	0.08	0.08	0.09
SAA	22	46	45	27	27

Notice that the AN structure of cases 19a and 35 in Figures B.7 and B.8 are different from cases 11, 14, and 19 in Figures B.2 - B.6. An AN is called dense if the ratio of number of arcs and number of nodes is large [5]. For an AN with A-on-N representation, we define the depth of an AN to be the number of nodes on the path with the most nodes and define the breadth of an AN to be the largest number of activities that are in progress at the same time during the course of the project. Cases 19a and 35 are very dense networks. Moreover, cases 19a and 35 have small depth and large breadth. Cases 19a and 35 are neural network-like ANs. Due to the parallel nature of the SAA algorithm, the SAA algorithm is good at dense ANs with small depth and large breadth in terms of computing time. Also notice that the SAA algorithm's computing speed depends on the fixed durations and distributional parameters of the random delays a lot. When a permutation between the data within Tables B.4 and B.5 is applied, with the structure of the ANs unchanged, the averaged computing time for case 19a and 35 is about 7 times of the time in Table 6.4 whereas the computing time for the SGD algorithm varies little for different duration and distribution parameters. It can be concluded that the SGD algorithm outperforms the SAA algorithm in computing speed for cases 11, 14 and 19 whose AN's structure have large depth and small breadth. And the SAA algorithm outperforms the SGD algorithm in computing speed for cases 19a and 35, whose AN's structure are dense and have small depth and large breadth.

Table 6.5 is the RAM usage for 10 runs of the SGD and SAA algorithms. Due to the

parallel computing nature of the SAA algorithm, its RAM usage is proportional to the computing time (more runs more RAM used). As for the SGD algorithm, its RAM usage is independent of the number of runs. Since the SGD algorithm only use single core, its RAM usage is significantly smaller than that of the SAA algorithm, which can be concluded from Table 6.5.

In conclusion, the SGD algorithm produces an upper bound of the optimal value with lower mean and tight C.I. with little RAM usage. The SGD algorithm is faster than the SAA algorithm for non-dense ANs with large depth and small breadth. The SAA algorithm is faster than the SGD algorithm for dense ANs with small depth and large breadth. The SGD may converge to a local optimum if the batch size is too large; choosing a smaller batch size and have multiple runs can alleviate this issue. Both the SAA and SGD algorithms require multiple runs to find the best solution.

6.6 Extensions

In this section, we extend our formulation and gradient-based algorithm to more general situations:

- The disruption affects activities that have yet to complete.
- When a disruption occurs prior to the complete of an activity, the delay it causes is proportional to the remaining time for completing the activity based on its original duration.
- Multiple disruptions occur during the course of the project.
- The nominal duration of activities are random variables.

6.6.1 Unfinished Activities

The problem formulation in Section 6.2 assumes that activities that have already started are not affected by the disruption, even if they haven't finished, i.e., the disruption only affects activities that have not started [37]. In this section, we assume that the disruption affects *all* unfinished activities. Yang and Morton [37] proposed the following formulation to relax this assumption:

$$z^* = \min_{\mathbf{t},\boldsymbol{\theta}} \quad p^0 t_{N_D} + \sum_{\omega\in\Omega} p^\omega f^\omega(\mathbf{t},\boldsymbol{\theta}) \tag{6.19}$$

$$\text{s.t.} \quad t_k - t_i \geq d_k(1 - \sum_{j\in J_i} e_i^j \theta_i^j) \qquad \forall (i,k) \in \mathcal{A} \tag{6.19a}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in J_i} b_i^j \theta_i^j \leq B \tag{6.19b}$$

$$\sum_{j\in J_i} \theta_i^j \leq 1 \qquad \forall i \in \mathcal{N} \tag{6.19c}$$

$$H^\omega + M G_i^\omega \geq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.19d}$$

$$H^\omega - M(1 - G_i^\omega) \leq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.19e}$$

$$t_i^\omega + M' G_i^\omega \geq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.19f}$$

$$t_i^\omega - M' G_i^\omega \leq t_i \qquad \forall i \in \mathcal{N}, \omega \in \Omega \tag{6.19g}$$

$$\theta_i^{j\omega} + \bar{\theta}_i^j G_i^\omega \geq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \tag{6.19h}$$

$$\theta_i^{j\omega} - \bar{\theta}_i^j G_i^\omega \geq \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \tag{6.19i}$$

$$t_k^\omega - t_i^\omega \geq d_k + X_k^\omega G_k^\omega - \sum_{j\in J_k} d_k e_k^j \theta_k^{j\omega} - \sum_{j\in J_k} X_k^\omega e_k^j z_k^{j\omega} \qquad \forall (i,k) \in \mathcal{A}, \omega \in \Omega \tag{6.19j}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in J_i} b_i^j \theta_i^{j\omega} \le B \qquad \forall \omega \in \Omega \qquad (6.19k)$$

$$\sum_{j\in J_i} \theta_i^{j\omega} \le 1 \qquad \forall i \in \mathcal{N}, \omega \in \Omega \qquad (6.19l)$$

$$z_i^{j\omega} \le \bar{\theta}_i^j G_i^\omega \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.19m)$$

$$z_i^{j\omega} \le \theta_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.19n)$$

$$z_i^{j\omega} \le \theta_i^j + \bar{\theta}_i^j (G_i^\omega - 1) \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.19o)$$

$$t_i \ge 0 \qquad \forall i \in \mathcal{N} \qquad (6.19p)$$

$$t_i^\omega \ge H^\omega G_i^\omega \qquad \forall i \in \mathcal{N}, \omega \in \Omega \qquad (6.19q)$$

$$0 \le \theta_i^j \le \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.19r)$$

$$0 \le \theta_i^{j\omega} \le \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i, \omega \in \Omega \qquad (6.19s)$$

$$0 \le z_i^{j\omega} \le 1 \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.19t)$$

$$G_i^\omega \in \{0, 1\} \qquad \forall i \in \mathcal{N}, \omega \in \Omega. \qquad (6.19u)$$

The difference between Problem (6.19) and Problem (6.11) is: (1) In Problem (6.19), t_i denotes the completion (finish) time of activity i instead of the starting time of activity i; (2) In constraints (6.19a) and (6.19i), if node i precedes k, then the difference between the finish time of activity k and i is no less than the duration of activity k after crashing. Formulation (6.19) assumes that the crashing decisions can be adjusted before the activity ends, which seems unrealistic for most real-world applications. For example, if the disruption happens near the very end of completion of activity i, one would expect that adjusting the crashing decision shouldn't decrease the total duration of activity i, only the remaining time. We do not make this assumption, and our

corresponding new amended formulation is given by

$$f(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{X}, H) = \min \quad \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[N_D]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[N'_D]} + d_{N'_D} \leq H\} + \tilde{t}_{N_D} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[N'_D]} + d_{N'_D} > H\} \tag{6.20}$$

$$\begin{aligned}
\text{s.t.} \quad & \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k']} + d_{k'} \leq H\} + \tilde{t}_k I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k']} + d_{k'} > H\} \\
& - (\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} + d_{i'} \leq H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} + d_{i'} > H\}) \\
& \geq (X_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i > H\} + d_i)\Big(1 - \sum_{j \in J_i} e_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} \leq H\} \\
& \quad + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} > H\})\Big) \qquad \forall (i,k) \in \mathcal{A} \qquad \text{(6.20a)} \\
& \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i > H\}) \leq B \qquad \text{(6.20b)} \\
& \sum_{j \in J_i} (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i > H\}) \leq 1 \qquad \forall i \in \mathcal{N} \qquad \text{(6.20c)} \\
& 0 \leq \theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} + d_i > H\} \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad \text{(6.20d)} \\
& \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} + d_{i'} \leq H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[i']} + d_{i'} > H\} \geq 0 \qquad \forall i \in \mathcal{N} \qquad \text{(6.20e)}
\end{aligned}$$

In Problem (6.20), $\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon})_{[k]}$ still denotes the starting time of activity k, but $k' = \underset{i \in \mathbf{pred}(k)}{\arg\max} \quad (t_i + d_i)$, i.e., k' is the index of the predecessor node of node k with the largest completion time. Note that in constraint (6.20a), the crashing decision of activity i can be revised if and only if it starts after the disruption. The activities' completion times are denoted by $\boldsymbol{\nu}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) =$

$[\nu_0, \nu_1, \nu_2, ..., \nu_{N_D}]$, where $\nu_i = t_i + d_i$. Reorder the element of $\boldsymbol{\nu}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ in ascending order so that $\nu^{(k)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ is the k^{th} smallest element of $\boldsymbol{\nu}$. We define the event $E^{(m)} = \{\nu^{(m-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H \leq \nu^{(m)}(\boldsymbol{\theta}, \boldsymbol{\epsilon})\}$ for $1 \leq m \leq N_D - 1$, and $E^{(N_D)} = \{\nu^{(N_D-1)}(\boldsymbol{\theta}, \boldsymbol{\epsilon}) < H\}$. By conditioning on $E^{(m)}$ and unconditioning, we rewrite Equation (6.4b) as:

$$g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) = \sum_{m=0}^{N_D} \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]\mathbb{P}(E^{(m)})$$

The gradient estimator and proof of unbiasedness are similar to Sections 6.3.2 and 6.3.3 and hence omitted here.

6.6.2 Proportional Duration Delay

Under the settings up to now, a disruption occurring near the end of an activity causes the same delay as if it occurred at the start of the activity, whereas an alternative is to have the delay proportional to the remaining duration. This can be handled by replacing each $\{X_i\}$ in Problem (6.20) by

$$\frac{d_i - \max(\min(t_i + d_i - H, d_i), 0)}{d_i} X_i,$$

but now $f(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{X}, H)$ in Problem (6.20) depends on H, given that H is in $E^{(m)}$. As long as all the indicator functions in Problem (6.20) do not depend on H, given that H is in $E^{(m)}$, the gradient estimators in Section 6.3 still work.

6.6.3 Multiple Disruptions

In the problem setting of [37], it is assumed that there can be at most a single disruption during the course of the project, which is reasonable when the disruptions are rare events, such as hurricanes, labor strikes, etc. However, in real applications, disruptions with high frequency can also cause delays for projects, for example, rainy days, floods, high temperature, which can occur multiple times during the course of the project. Our model can be adjusted to handle multiple disruptions. Assume $\{H_i\}$ is the set of occurrence time of different disruptions, whose joint distribution is known. Then Problem (6.5) can be adjusted to:

$$
\begin{aligned}
f(\mathbf{t}, \boldsymbol{\theta}, \mathbf{X}, \{H_i\}) = \min \quad & t_{N_D} \prod_{n=1}^{n_E} I\{q_{N_D} \le H_n\} + \tilde{t}_{N_D} \prod_{n=1}^{n_E} I\{q_{N_D} > H_n\} && (6.21)\\
\text{s.t.} \quad & t_k \prod_{n=1}^{n_E} I\{q_k \le H_n\} + \tilde{t}_k \prod_{n=1}^{n_E} I\{q_k > H_n\} \\
& \quad - (t_i \prod_{n=1}^{n_E} I\{q_i \le H_n\} + \tilde{t}_i \prod_{n=1}^{n_E} I\{q_i > H_n\}) \\
& \ge (d_i + \sum_{n=1}^{n_E} X_i^n I\{t_i > H_n\}) \\
& \quad (1 - \sum_{j \in J_i} e_i^j (\theta_i^j \prod_{n=1}^{n_E} I\{t_i \le H_n\} + \tilde{\theta}_i^j \prod_{n=1}^{n_E} I\{t_i > H_n\})) \qquad \forall (i,k) \in \mathcal{A} \\
& && (6.21a)\\
& \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j (\theta_i^j \prod_{n=1}^{n_E} I\{t_i \le H_n\} + \tilde{\theta}_i^j \prod_{n=1}^{n_E} I\{t_i > H_n\}) \le B && (6.21b)\\
& \sum_{j \in J_i} (\theta_i^j \prod_{n=1}^{n_E} I\{t_i \le H_n\} + \tilde{\theta}_i^j \prod_{n=1}^{n_E} I\{t_i > H_n\}) \le 1 \qquad \forall i \in \mathcal{N} \\
& && (6.21c)
\end{aligned}
$$

$$0 \leq \theta_i^j \prod_{n=1}^{n_E} I\{t_i \leq H_n\} + \tilde{\theta}_i^j \prod_{n=1}^{n_E} I\{t_i > H_n\} \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \tag{6.21d}$$

$$t_i \prod_{n=1}^{n_E} I\{q_i \leq H_n\} + \tilde{t}_i \prod_{n=1}^{n_E} I\{q_i > H_n\} \geq 0 \qquad \forall i \in \mathcal{N} \tag{6.21e}$$

In constraint (6.21a), X_i^n is the random delay of activity i caused by disruption n. At the begining of Section 6.3.1, we partition the positive real line into $N_D + 1$ intervals and define event $E^{(m)}$ as H falls into the mth interval. Similarly, we can partition the $\mathbb{R}^{+n}$ space into $(N_D + 1)^{n_E}$ disjoint sets and define event $E^{(m)}$ as $\{H_n\}$ falls into the mth partition. Then Equation (6.4c) can be reformulated as:

$$g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}) = \sum_{m=0}^{N_D^{n_E}} \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H) | E^{(m)}] \mathbb{P}(E^{(m)}),$$

and the corresponding one-sided gradient estimators are similar to Section 6.3.

6.6.4 Random Durations of Activities

In previous sections, it is assumed that the duration of activities are deterministic if no disruption happened or disruptions happened after the completion of the project. Our formulation in (6.7S) and (6.8) can be adjusted to handle the setting where activity durations are continuous random variables instead of deterministic. To adjust our model, first redefine Equation (6.4b) as

$$g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, H, \mathbf{D}}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})] \tag{6.22}$$

and the second-stage Problem (6.8) becomes:

$$f(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{X}, H, \mathbf{D}) = \min \quad \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[N_D]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[N'_D]} \leq H\} + \tilde{t}_{N_D} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[N'_D]} > H\} \tag{6.23}$$

$$\begin{aligned}
\text{s.t.} \quad & \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[k]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[k']} \leq H\} + \tilde{t}_k I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[k']} > H\} \\
& - (\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']} \leq H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']} > H\}) \\
& \geq (X_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} > H\} + D_i)\Big(1 - \sum_{j \in J_i} e_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} \leq H\} \\
& \qquad + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} > H\})\Big) \qquad \forall (i,k) \in \mathcal{A} \qquad (6.23a) \\
& \sum_{i \in \mathcal{N}} \sum_{j \in J_i} b_i^j (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} > H\}) \leq B \qquad (6.23b) \\
& \sum_{j \in J_i} (\theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} > H\}) \leq 1 \qquad \forall i \in \mathcal{N} \qquad (6.23c) \\
& 0 \leq \theta_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} \leq H\} + \tilde{\theta}_i^j I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} > H\} \leq \bar{\theta}_i^j \qquad \forall i \in \mathcal{N}, j \in J_i \qquad (6.23d) \\
& \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i]} I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']} \leq H\} + \tilde{t}_i I\{\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']} > H\} \geq 0 \qquad \forall i \in \mathcal{N}, \qquad (6.23e)
\end{aligned}$$

where $\{D_i\}$ are independently distributed continuous random variables and $\mathbf{D}$ is the vector representation of $\{D_i\}$. Further rewrite Equation (6.4b) as

$$\begin{aligned} g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}) =& \mathbb{E}_{\mathbf{X},H,\mathbf{D}}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})] \\ =& \sum_{m=0}^{N_D} \mathbb{E}_{\mathbf{D}}[\mathbb{E}_{\mathbf{X}}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})|E^{(m)}]\mathbb{P}(E^{(m)})] \\ =& \sum_{m=0}^{N_D} \mathbb{E}_{\mathbf{D},\mathbf{X}}[f^{(m)}(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})\mathbb{P}(E^{(m)})] \end{aligned}$$

where $f^{(m)}(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})$ is the function in Problem (6.23), given that H is in $E^{(m)}$. Then we have the adjusted gradient estimators given by

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})}{\partial \theta_i^{j\pm}} = \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']}}{\partial \theta_i^{j\pm}} y^*_{n_t(i',k')} - \sum_{\substack{(i',k') \in \mathcal{A}_3^H \\ i' = i}} D_i e_i^j y^*_{n_t(i',k')} + b_i^j y^*_{N_c} \tag{6.24A}$$

$$\frac{\partial f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H, \mathbf{D})}{\partial \epsilon_i^{\pm}} = \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \frac{\partial \mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}, \mathbf{D})_{[i']}}{\partial \epsilon_i^{\pm}} y^*_{n_t(i',k')} \tag{6.24B}$$

The unbiasedness proof for estimators (6.24A) and (6.24B) is similar to Theorem 1 and is omitted here.

Finally, notice that our formulation can handle not only the situations in Sections 6.6.1, 6.6.2, 6.6.3 and 6.6.4 separately, but also any combination of those cases, as well.

6.7 Future Research

Future research will focus on analyzing the extension cases in Section 6.6, both theoretically and empirically via numerical experiments. In Section 6.6.3, the complexity of the algorithm is NP-hard due to the exponentially increased number of partitions of the disruptions occurrence sample space. Future work will focus on reducing the time complexity of the multiple disruptions setting and applying parallel computing to accelerate computing speed. Finally, it would be interesting to apply the algorithms proposed to some real world business cases (e.g., deep water drilling, disaster and pandemic management).

Appendix A: Activity-on-Arc (A-on-A) Formulation

In this section, we will use the A-on-A representation to reformulate the two-stage stochastic programming problem in Section 6.2.1 and derive its stochastic gradient estimator based on the new formulation. The materials in this section are from [38]. The the A-on-A version stochastic gradient descent algorithm is provided. Finally, the difference between the A-on-A formulation and the A-on-N formulation is provided.

A.1 Linear Programming Formulation

The activity on arc LP formulation of Problem 6.1 is given by:

$$z^* = \min_{\boldsymbol{\theta}} \quad t_4 \tag{A.0}$$

$$\text{s.t.} \quad t_4 - t_3 \geq (1 - \theta_{34}) d_{34} \tag{A.0a}$$

$$t_4 - t_2 \geq (1 - \theta_{24}) d_{24} \tag{A.0b}$$

$$t_3 - t_2 \geq (1 - \theta_{23}) d_{23} \tag{A.0c}$$

$$t_3 - t_1 \geq (1 - \theta_{13}) d_{13} \tag{A.0d}$$

$$t_2 - t_1 \geq (1 - \theta_{12}) d_{12} \tag{A.0e}$$

$$\theta_{12} + \theta_{13} + \theta_{23} + \theta_{24} + \theta_{34} \leq B \tag{A.0f}$$

$$t_i \geq 0 \qquad \forall i \in \{1,2,3,4\} \qquad \text{(A.0g)}$$

The difference between the A-on-N and A-on-A formulations of our two-stage stochastic programming problem is that Problem 6.7 and 6.8 are substituted by the A-on-A LP formulation. Also, for the same problem setting, A-on-A and A-on-N formulations are equivalent if and only if the $\boldsymbol{\theta}$ in Equations (6.6a) and (6.6b) is a zero vector.

A.2 A-on-A Two-Stage Formulation

Notations will be used in this section are given by:

X_{ik} = delay of duration of arc (i,k) created by the disruption

d_{ik} = fixed duration of activity (i,k)

t_k = the node release time of node k

J_{ik} = the set of crashing option for activity (i,k)

b_{ik}^j = per unit cost of the jth crashing option of activity (i,k)

e_{ik}^j = ratio of decreasing of activity (i,k)'s duration per unit jth crashing option applied

θ_{ik}^j = amount of jth option of activity (i,k)

$\bar{\theta}_{ik}^j$ = upper bound of jth option of activity (i,k)

$\textbf{pred}(k) = \{i|(i,k) \in \mathcal{A}\}$

$\textbf{succ}(k) = \{i|(k,i) \in \mathcal{A}\}$

$p_k = \max_{i \in \textbf{pred}(k)} \{t_i\}$

$t(\boldsymbol{\theta})_{[k]} =$ node release time of node k

$t(\boldsymbol{\theta})_{[k']} = \max_{i \in \mathbf{pred}(k)} \{t(\boldsymbol{\theta})_{[i]}\}$

The two-stage A-on-A formulation is given by:

$$z^* = \min_{\boldsymbol{\theta}} \quad g(t(\boldsymbol{\theta}), \boldsymbol{\theta}) \tag{A.1}$$

$$\text{s.t.} \quad \sum_{j \in J_{ik}} \sum_{(i,k) \in \mathcal{A}} b_{ik}^j \theta_{ik}^j \leq B \tag{A.1a}$$

$$\sum_{j \in J_{ik}} \theta_{ik}^j \leq 1 \qquad \forall (i,k) \in \mathcal{A} \tag{A.1b}$$

$$0 \leq \theta_{ik}^j \leq \bar{\theta}_{ik}^j \qquad \forall j \in J_{ik}, \forall (i,k) \in \mathcal{A} \tag{A.1c}$$

and its second-stage formulation given by:

$$f(t(\boldsymbol{\theta}), \boldsymbol{\theta}, \mathbf{X}, H) = \min \quad t(\boldsymbol{\theta})_{[N_D]} I\{t(\boldsymbol{\theta})_{[N'_D]} \leq H\} + \tilde{t}_{N_D} I\{t(\boldsymbol{\theta})_{[N'_D]} > H\} \tag{A.2}$$

$$\begin{aligned} \text{s.t.} \quad & t(\boldsymbol{\theta})_{[k]} I\{t(\boldsymbol{\theta})_{[k']} \leq H\} + \tilde{t}_k I\{t(\boldsymbol{\theta})_{[k']} > H\} \\ & - (t(\boldsymbol{\theta})_{[i]} I\{t(\boldsymbol{\theta})_{[i']} \leq H\} + \tilde{t}_i I\{t(\boldsymbol{\theta})_{[i']} > H\}) \\ & \geq (X_{ik} I\{t(\boldsymbol{\theta})_{[i]} > H\} + d_{ik}) \\ & \quad (1 - \sum_{j \in J_{ik}} e_{ik}^j (\theta_{ik}^j I\{t(\boldsymbol{\theta})_{[i]} \leq H\} + \tilde{\theta}_{ik}^j I\{t(\boldsymbol{\theta})_{[i]} > H\})) \quad \forall (i,k) \in \mathcal{A} \end{aligned} \tag{A.2a}$$

$$\sum_{j \in J_{ik}} \sum_{(i,k) \in \mathcal{A}} b_{ik}^j (\theta_{ik}^j I\{t(\boldsymbol{\theta})_{[i]} \leq H\} + \tilde{\theta}_{ik}^j I\{t(\boldsymbol{\theta})_{[i]} > H\}) \leq B$$

$$\sum_{j\in J_{ik}}(\theta^j_{ik}I\{t(\boldsymbol{\theta})_{[i]}\le H\}+\tilde{\theta}^j_{ik}I\{t(\boldsymbol{\theta})_{[i]}>H\})\le 1 \qquad \forall(i,k)\in\mathcal{A}$$

$$0\le \theta^j_{ik}I\{t(\boldsymbol{\theta})_{[i]}\le H\}+\tilde{\theta}^j_{ik}I\{t(\boldsymbol{\theta})_{[i]}>H\}\le \bar{\theta}^j_{ik} \qquad \forall j\in J_{ik},\forall(i,k)\in\mathcal{A}$$

$$t(\boldsymbol{\theta})_{[i]}I\{t(\boldsymbol{\theta})_{[i']}\le H\}+\tilde{t}_iI\{t(\boldsymbol{\theta})_{[i']}>H\}\ge 0 \qquad \forall i\in\{0,1,...,N_D\}$$

where we have

$$g(t(\boldsymbol{\theta}),\boldsymbol{\theta})=\mathbb{E}_{\mathbf{X},H}[f(t(\boldsymbol{\theta}),\boldsymbol{\theta},\mathbf{X},H)]$$

and $t(\boldsymbol{\theta})$ as a vector is defined to be as below, whose elements are given by two equivalent formulations:

$$t(\boldsymbol{\theta})_{[k]}=\max_{i\in\textbf{pred}(k)}\Big\{d_{ik}(1-\sum_{j\in J_{ik}}e^j_{ik}\theta^j_{ik})+t(\boldsymbol{\theta})_{[i]}\Big\},\quad \forall 2\le k\le N_D \tag{A.3a}$$

$$t(\boldsymbol{\theta})_{[k]}=\max_{P\in\mathcal{P}_k}\Big\{\sum_{(i_1,i_2)\in P}d_{i_1i_2}(1-\sum_{j\in J_{i_1i_2}}e^j_{i_1i_2}\theta^j_{i_1i_2})\Big\},\quad \forall 2\le k\le N_D \tag{A.3b}$$

Similarly to Table 6.1 in Section 6.2.1.3, for a given realization of H, the disruption occurrence time, we classify constraint (A.2a) into four mutually exclusive classes in Table A.1.

Table A.1: Four Types of Second Stage Constraints

Types	**Constraint**	**Range of H**
Type I	$\tilde{t}_k-\tilde{t}_i\ge(X_{ik}+d_{ik})(1-\sum_{j\in J_{ik}}e^j_{ik}\tilde{\theta}^j_{ik})$	$H<p_i$
Type II	$\tilde{t}_k-t_i\ge(X_{ik}+d_{ik})(1-\sum_{j\in J_{ik}}e^j_{ik}\tilde{\theta}^j_{ik})$	$p_i\le H<t_i$
Type III	$\tilde{t}_k-t_i\ge d_{ik}(1-\sum_{j\in J_{ik}}e^j_{ik}\theta^j_{ik})$	$t_i\le H<p_k$
Type IV	$t_k-t_i\ge d_{ik}(1-\sum_{j\in J_{ik}}e^j_{ik}\theta^j_{ik})$	$H\ge p_k$

A.3 A-on-A Stochastic Gradient Estimator

Similarly to Section 6.3.1, for a given fixed set of crashing parameter $\boldsymbol{\theta}$, the node release time of all nodes without disruption $\mathbf{T}$ can be calculated as $\mathbf{T} = t(\boldsymbol{\theta})$, where $\mathbf{T} = [t_1, t_2, ..., t_{N_D}]$. Reorder the element of $\mathbf{T}$ in ascending order so that $t^{(k)}$ is the kth order element. By conditioning on $t^{(m-1)} < H \leq t^{(m)}$ and unconditioning, we have the equivalent formulation of Equations (6.4c) and (6.13a) given by:

$$g(\mathbf{t}(\boldsymbol{\theta}), \boldsymbol{\theta}) = \sum_{m=1}^{N_D} \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]\mathbb{P}(E^{(m)}) \tag{A.4}$$

and

$$\begin{aligned}
&\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}, \boldsymbol{\epsilon}), \boldsymbol{\theta})}{\partial \theta_{ik}^j} \\
&= \sum_{m=1}^{N_D} \Bigg\{ \frac{\partial \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}]}{\partial \theta_{ik}^j} \mathbb{P}(E^{(m)}) \\
&\quad + \mathbb{E}[f(\mathbf{t}(\boldsymbol{\theta}), \boldsymbol{\theta}, \mathbf{X}, H)|E^{(m)}] \Big(f_H(t^{(m)}(\boldsymbol{\theta})) \frac{\partial t^{(m)}(\boldsymbol{\theta})}{\partial \theta_{ik}^j} - f_H(t^{(m-1)}(\boldsymbol{\theta})) \frac{\partial t^{(m-1)}(\boldsymbol{\theta})}{\partial \theta_{ik}^j} \Big) \Bigg\}.
\end{aligned} \tag{A.5}$$

And the stochastic gradient estimators are given by:

$$\frac{\partial f(t(\boldsymbol{\theta}), \boldsymbol{\theta}, \mathbf{X}, H)}{\partial \theta_{ik}^{j\pm}} = \sum_{\substack{(i',k') \in \mathcal{A}_2^H \cup \mathcal{A}_3^H \\ i' > i}} \frac{\partial t(\boldsymbol{\theta})_{[i']}}{\partial \theta_{ik}^{j\pm}} y^*_{n_t(i',k')} - I\{(i,k) \in \mathcal{A}_3^H\} d_{ik} e_{ik}^j y^*_{n_t(i,k)} + b_{ik}^j y^*_{N_G}. \tag{A.6}$$

where $n_t(i', k')$ is the line number correspond to activity (i', k') in constraint (A.2a) and N_G is the line number corresponding to the budget constraint. Proof of the unbiasness of the estimator in Equation (A.6) is very similar to Theorem 1 and is omitted here.

A.4 A-on-A Gradient Descent Algorithm

The stochastic gradient descent algorithm based on the estimator in Equation (A.6) is given by Algorithm 7 and 8 below:

Algorithm 7: Phase 1 Optimization

$\boldsymbol{\theta} \leftarrow \mathbf{0}$
while $B > 0$ **do**
  $(i, j) \leftarrow \underset{\substack{(i,k,j)\in\mathcal{O} \\ \theta^j_{ik}<\bar{\theta}^j_{ik}}}{\arg\min} \{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}),\boldsymbol{\theta})}{\partial \theta^{j+}_{ik}} / b^j_{ik}\}$
  $\delta^j_{ik} \leftarrow \alpha_1 \bar{\theta}^j_{ik}$
  $\theta^j_{ik} \leftarrow \theta^j_{ik} + \min(b^j_{ik}\delta^j_{ik}, b^j_{ik}(1 - \sum_{n\in J_i, n\neq j} \theta^n_{ik}), B)/b^j_{ik}$
  $B \leftarrow B - \min(b^j_{ik}\delta^j_{ik}, b^j_{ik}(1 - \sum_{n\in J_i, n\neq j} \theta^n_{ik}), B)$
end

Algorithm 8: Phase 2 Optimization

$\delta_d \leftarrow \frac{B}{N_d}$; $n \leftarrow 0$
while $n < M$ **do**
  $(i_1, k_1, j_1) \leftarrow \underset{\substack{(i,k,j)\in\mathcal{O} \\ 0<\theta^j_{ik}<\bar{\theta}^j_{ik}}}{\arg\min} \{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}),\boldsymbol{\theta})}{\partial \theta^{j-}_{ik}} / b^j_{ik}\}$
  $(i_2, k_2, j_2) \leftarrow \underset{\substack{(i,k,j)\in\mathcal{O} \\ 0<\theta^j_{ik}<\bar{\theta}^j_{ik}}}{\arg\max} \{\frac{\partial g(\mathbf{t}(\boldsymbol{\theta}),\boldsymbol{\theta})}{\partial \theta^{j-}_{ik}} / b^j_{ik}\}$
  $\delta_b \leftarrow \min(\delta_d, b^{j_2}_{i_2k_2}\theta^{j_2}_{i_2k_2}, b^{j_1}_{i_1k_1}(1 - \sum_{n\in J_{i_1k_1}, n\neq j_1} \theta^n_{i_1k_1}))$
  $\theta^{j_1}_{i_1k_1} \leftarrow \theta^{j_1}_{i_1k_1} + \delta_b / b^{j_1}_{i_1k_1}$
  $\theta^{j_2}_{i_2k_2} \leftarrow \theta^{j_2}_{i_2k_2} - \delta_b / b^{j_2}_{i_2k_2}$
  $n \leftarrow n + 1$
end

where we have $\mathcal{O} = \{(i, k, j) | j \in J_{ik}, (i, k) \in \mathcal{A}\}$. Algorithm 8 takes algorithm 7's output

as input. The only difference between the A-on-N Algorithm (Algorithm 5 and 6) and the A-on-A Algorithm (Algorithm 7 and 8) is that no iterations of $\boldsymbol{\epsilon}$ is required, because we assume that $\boldsymbol{\epsilon}$ is a zero vector.

A.5 Difference between the Two Formulations

Notice that the A-on-A formulation and the A-on-N formulation are equivalent if and only if for the optimal solution we have that $\boldsymbol{\epsilon}^* = \mathbf{0}$. The advantage of the A-on-A formulation is that for ANs whose A-on-A representation have less nodes and arcs compared to its A-on-N representation, the A-on-A algorithm compute faster than the A-on-N algorithm, which is the reason why for cases 11, 14 and 19 in Section 6.5, the A-on-A algorithm is used. Since we cannot check the condition that $\boldsymbol{\epsilon}^* = \mathbf{0}$, we can run Algorithm 6 a few times only for iteration of $\boldsymbol{\epsilon}$ with input the optimal solution calculated by the A-on-A algorithm.

Appendix B: Experimental Data for Chapter 6

B.1 Optimal Crashing Activity Networks

The experimental data of Section 6.5 are from [37] and [38] and provided here. For all test cases, we assume there is only one possible crashing option for each activity. The option consumes 1 unit of resource and has effectiveness parameter $e_i^1 = 0.5$ for all $i \in \mathcal{N}$. The probability that the disruption will not occur is $p_0 = 0.1$. The disruption occurrence time H follows a lognormal distribution with parameters μ and σ, where the mode is $e^{\mu-\sigma^2}$. The random delays $\{X_i\}$ are independent and follow an exponential distribution with mean μ_i. The values of duration d_i and mean delay μ_i are shown in Tables B.1 - B.5. Figures B.1 - B.8 give the AN structure. The crashing budget B and lognormal distribution parameters are as follows:

- Case 11: B = 3, μ = ln 6, σ = 0.5
- Case 14: B = 4, μ = ln 35, σ = 0.5
- Case 19: B = 4, μ = ln 8, σ = 0.5
- Case 19a: B = 4, μ = ln 8, σ = 0.5
- Case 35: B = 8, μ = ln 4, σ = 0.3

Figure B.1: Activity Network with 11 Activities (A-on-A)

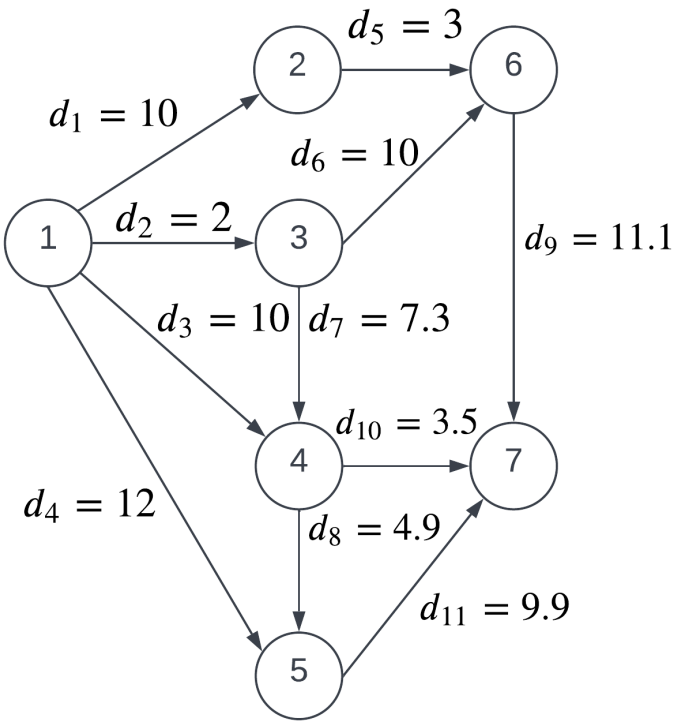


Figure B.2: Activity Network with 11 Activities (A-on-N)

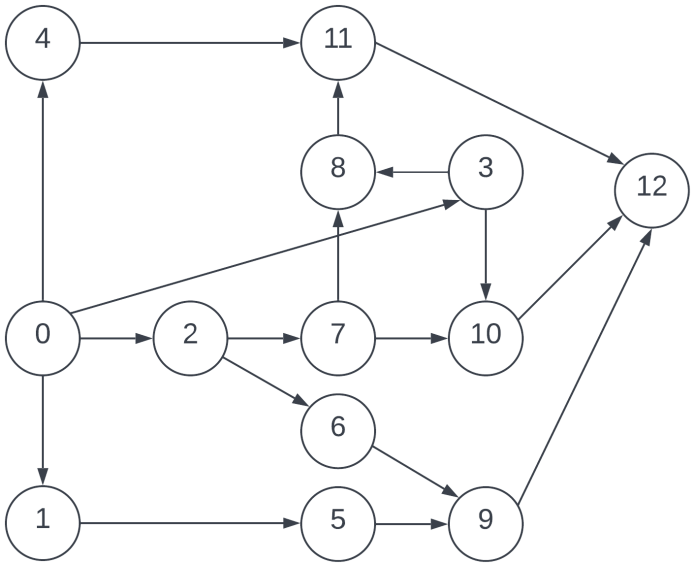


Table B.1: Activity Duration d_i and the Mean of Disruption Magnitude μ_i for Case 11

Activity	d_i	μ_i	**Activity**	d_i	μ_i
1	10	10^{-5}	7	7.3	1
2	2	4	8	4.9	50
3	10	2	9	11.1	40
4	12	30	10	3.5	40
5	3	1500	11	9.9	5
6	10	1			

Figure B.3: Activity Network with 14 Activities (A-on-A)

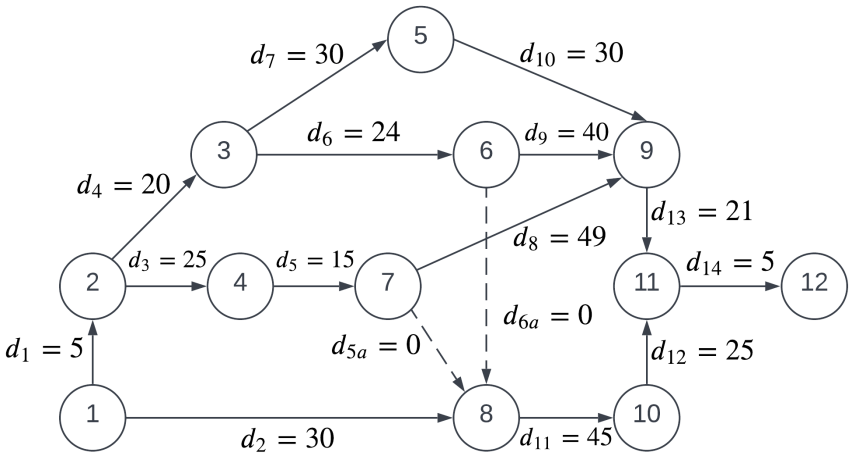


Figure B.4: Activity Network with 14 Activities (A-on-N)

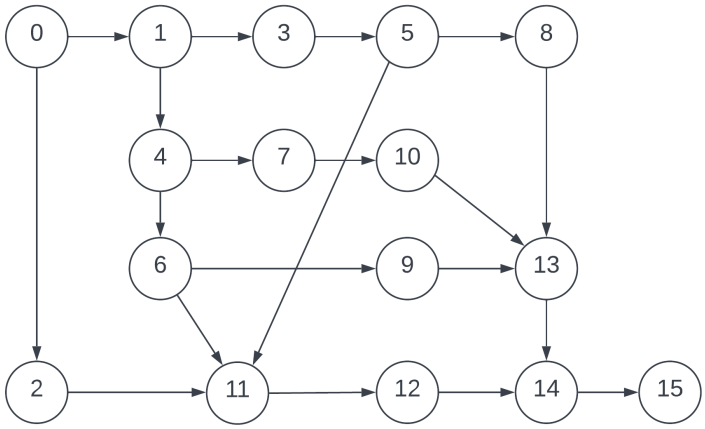


Table B.2: Activity Duration d_i and the Mean of Disruption Magnitude μ_i for Case 14

Activity	d_i	μ_i	**Activity**	d_i	μ_i
1	5	10^{-5}	8	49	4000
2	30	5	9	40	4000
3	25	40000	10	30	3000
4	20	40000	11	45	4000
5	15	1500	12	25	5
6	24	20000	13	21	5
7	30	20000	14	5	5

Figure B.5: Activity Network with 19 Activities (A-on-A)

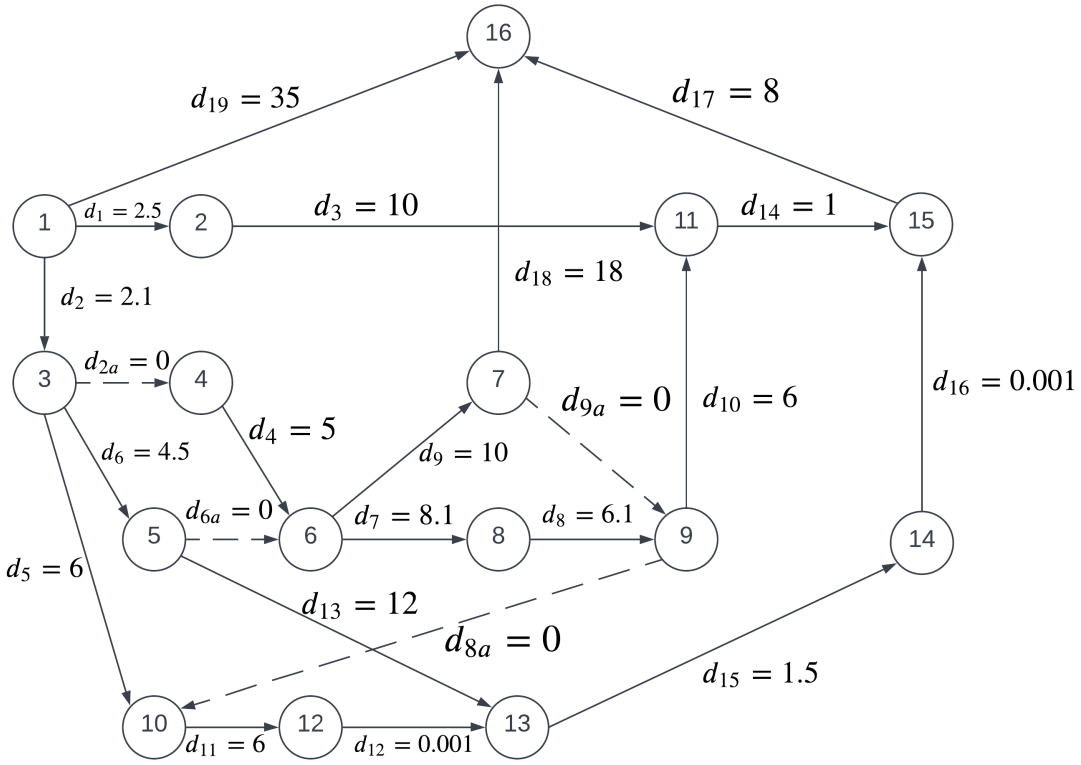


Figure B.6: Activity Network with 19 Activities (A-on-N)

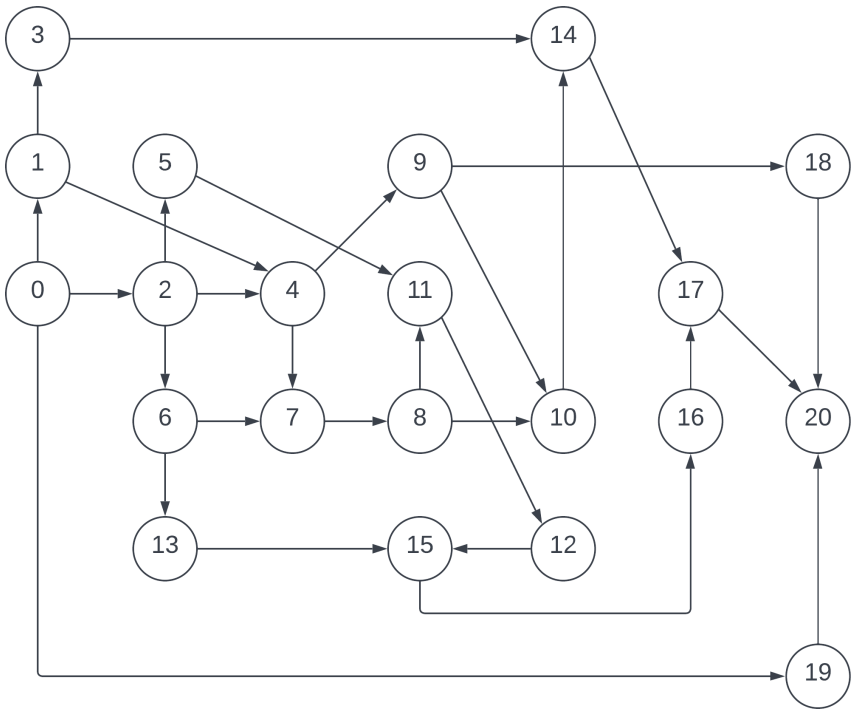


Table B.3: Activity Duration d_i and the Mean of Disruption Magnitude μ_i for Case 19

Activity	d_i	μ_i	**Activity**	d_i	μ_i
1	2.5	400	11	6	50
2	2.1	200	12	0.001	40
3	10	1000	13	12	1000
4	5	10	14	1	50
5	6	40	15	1.5	10
6	4.5	300	16	0.001	300
7	8.1	2	17	8	100
8	6.1	100	18	18	300
9	10	100	19	35	300
10	6	50			

Figure B.7: Activity Network with 19a Activities

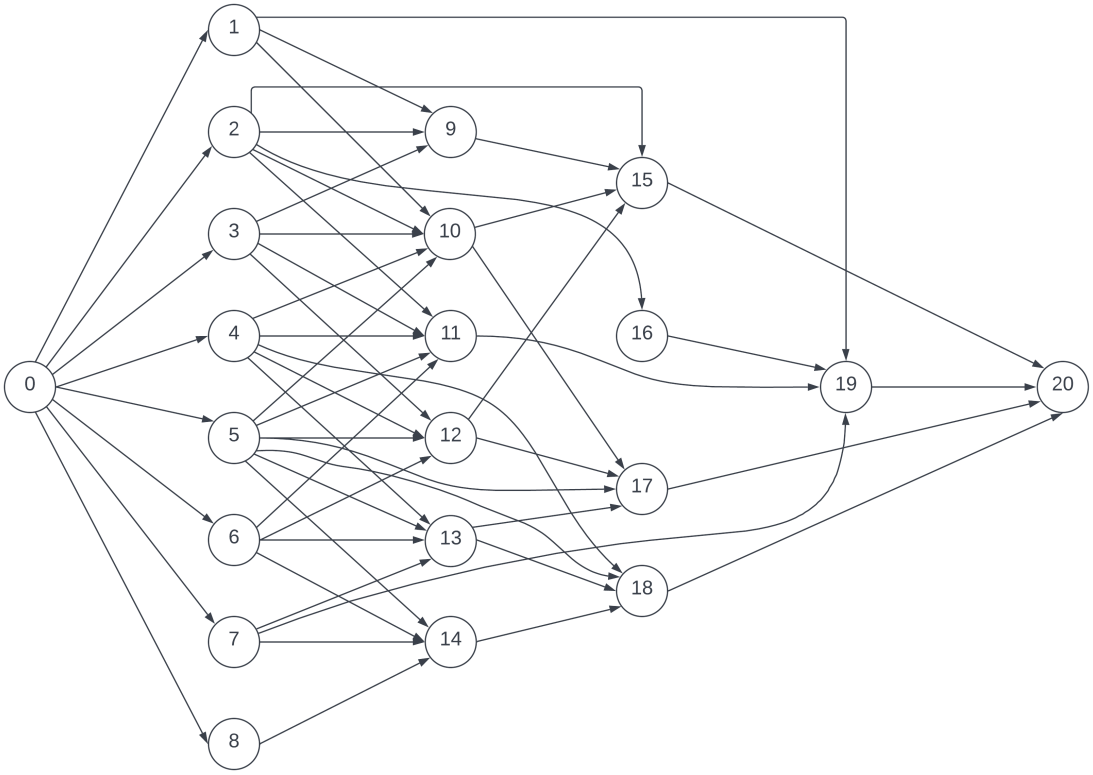


Table B.4: Activity Duration d_i and the Mean of Disruption Magnitude μ_i for Case 19a

Activity	d_i	μ_i	**Activity**	d_i	μ_i
1	5	10	11	6	400
2	10	100	12	4.5	300
3	18	300	13	12	1000
4	35	300	14	1.5	10
5	6	50	15	2.1	200
6	6	50	16	8.1	2
7	2.5	400	17	6.1	100
8	8	20	18	0.001	40
9	10	1000	19	0.001	300
10	1	50			

Figure B.8: Activity Network with 35 Activities

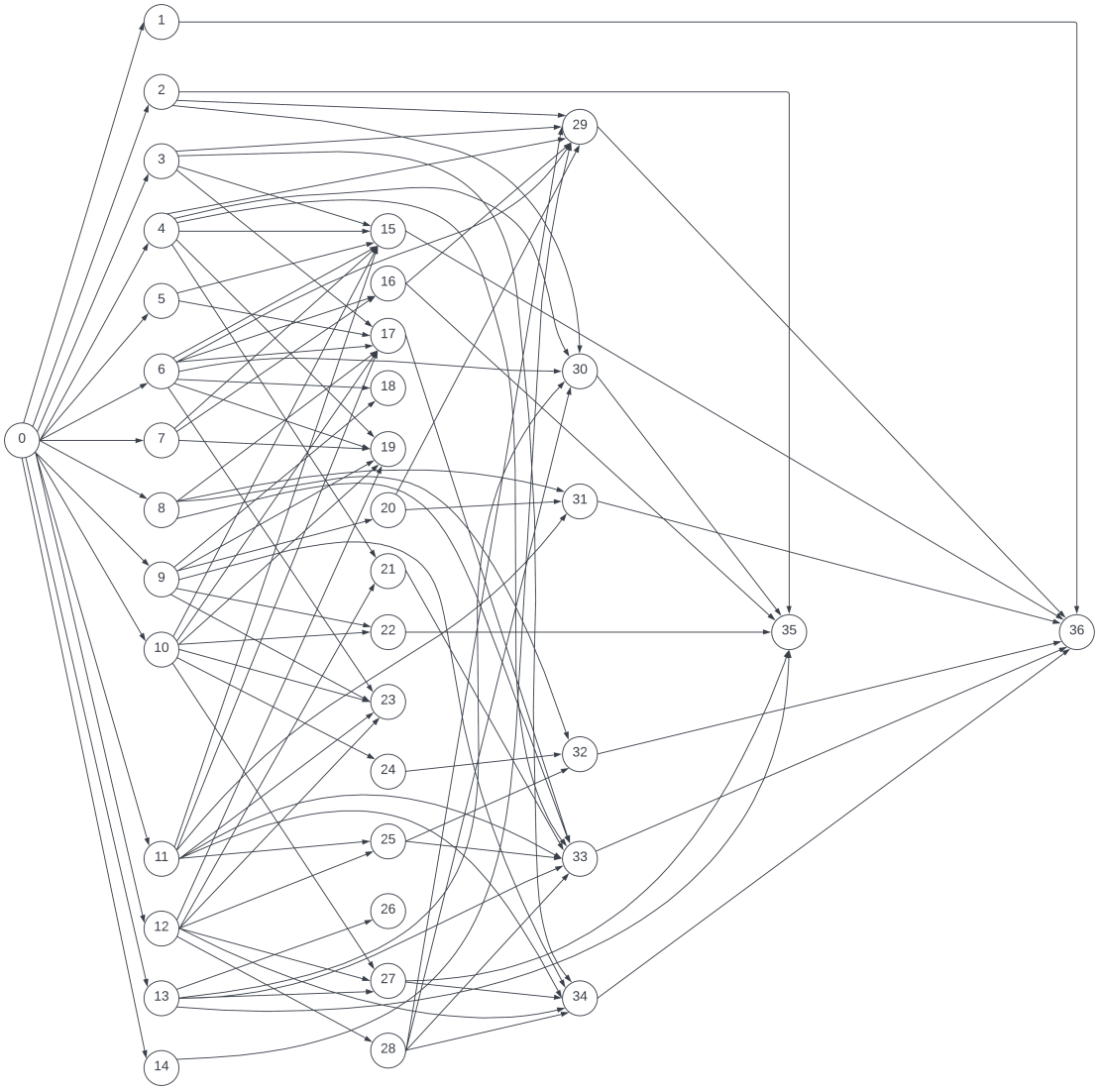


Table B.5: Activity Duration d_i and the Mean of Disruption Magnitude μ_i for Case 35

Activity	d_i	μ_i	**Activity**	d_i	μ_i	**Activity**	d_i	μ_i
1	9	10	13	5	200	25	1	300
2	7	40	14	2	10	26	4	400
3	3	30	15	5	400	27	3	200
4	4	100	16	2	10	28	4	1000
5	6	50	17	10	10	29	10	300
6	3	10	18	4	2000	30	7	500
7	10	10	19	8	10	31	2	200
8	4	20	20	8	500	32	9	100
9	3	10	21	1	500	33	7	100
10	6	1000	22	5	500	34	1	100
11	9	10	23	2	10	35	7	200
12	8	500	24	7	10			

Bibliography

[1] C.E. Sigal, A.A.B. Pritsker, and J.J. Solberg. The use of cutsets in Monte Carlo analysis of stochastic networks. *Mathematics and Computers in Simulation*, 21(4):376–384, 1979.

[2] R.A. Bowman. Efficient estimation of arc criticalities in stochastic activity networks. *Management Science*, 41(1):58–67, 1995.

[3] Salah E Elmaghraby. *Project Planning and Control by Network Models*. John Wiley and Sons New York, 1977.

[4] Seung-Jean Kim, Stephen P Boyd, Sunghee Yun, Dinesh D Patil, and Mark A Horowitz. A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing. *Optimization and Engineering*, 8(4):397–430, 2007.

[5] Erik Demeulemeester, Bajis Dodin, and Willy Herroelen. A random activity network generator. *Operations Research*, 41(5):972–980, 1993.

[6] Yuanjiang Chang, Yajie Jiang, Changshuai Zhang, Anti Xue, Bin Chen, Weiguo Zhang, Liangbin Xu, Xiuquan Liu, and Yongguo Dai. PERT-based emergency disposal technique for fracture failure of deepwater drilling riser. *Journal of Petroleum Science and Engineering*, 201:108407, 2021.

[7] John T. Robinson. Some analysis techniques for asynchronous multiprocessor algorithms. *IEEE Transactions on Software Engineering*, (1):24–31, 1979.

[8] Michael C Fu. Sensitivity analysis in Monte Carlo simulation of stochastic activity networks. In F. B. Alt, M. C. Fu, and B. L. Golden, editors, *Perspectives in Operations Research*, pages 351–366. Springer, 2006.

[9] C.E. Sigal, A.A.B. Pritsker, and J.J. Solberg. The stochastic shortest route problem. *Operations Research*, 28(5):1122–1129, 1980.

[10] Peng Wan and Michael C Fu. Sensitivity analysis of arc criticalities in stochastic activity networks. In *Proceedings of the 2020 Winter Simulation Conference (WSC)*, pages 2911–2922. IEEE, 2020.

[11] R.A. Bowman. Stochastic gradient-based time-cost tradeoffs in pert networks using simulation. *Annals of Operations Research*, 53(1):533–551, 1994.

[12] Bajis M Dodin and Salah E Elmaghraby. Approximating the criticality indices of the activities in PERT networks. *Management Science*, 31(2):207–223, 1985.

[13] Michael C Fu. *Handbook of Simulation Optimization*. Springer, 2015.

[14] Yu-Chi Ho and Xi-Ren Cao. *Discrete Event Dynamic Systems and Perturbation Analysis*. Kluwer Academic Publishers, Boston, 1991.

[15] Paul Glasserman. *Gradient Estimation via Perturbation Analysis*. Kluwer Academic Publishers, 1991.

[16] Reuven Y Rubinstein and Alexander Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, volume 13. Wiley, 1993.

[17] Shane G Henderson and Barry L Nelson. *Handbooks in Operations Research and Management Science: Simulation*. Elsevier, 2006.

[18] Michael C Fu and Jian-Qiang Hu. *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Kluwer Academic Publishers, 1997.

[19] Georg Ch Pflug. *Optimization of Stochastic Models: The Interface between Simulation and Optimization*. Kluwer Academic Publishers, 1996.

[20] Martin I Reiman and Alan Weiss. Sensitivity analysis via likelihood ratios. In *Proceedings of the 1986 Winter Simulation Conference*, pages 285–289, 1986.

[21] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[22] Reuven Y Rubinstein and Peter W Glynn. How to deal with the curse of dimensionality of likelihood ratios in Monte Carlo simulation. *Stochastic Models*, 25(4):547–568, 2009.

[23] Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, II: shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.

[24] J.H. Venter. An extension of the robbins-monro procedure. *The Annals of Mathematical Statistics*, 38(1):181–190, 1967.

[25] Sheldon M Ross. *Simulation*. Academic Press, 2013.

[26] S.E. Elmaghraby, Y. Fathi, and M.R. Taner. On the sensitivity of project variability to activity mean duration. *International Journal of Production Economics*, 62(3):219–232, 1999.

[27] Ron Aharoni, Abraham Berman, and Yair Censor. An interior points algorithm for the convex feasibility problem. *Advances in Applied Mathematics*, 4(4):479–489, 1983.

[28] J. Herskovits. An interior points method for nonlinear constrained optimization. In *Optimization of Large Structural Systems*, pages 589–608. Springer, 1993.

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

[30] Renato Mauricio Manterola. Characterization of gradient estimators for stochastic activity networks. Master's thesis, The University of Maryland at College Park, 2011.

[31] Bernd Heidergott, Felisa J Vázquez-Abad, and Warren Volk-Makarewicz. Sensitivity estimation for Gaussian systems. *European Journal of Operational Research*, 187(1):193–207, 2008.

[32] Peng Wan, Michael C Fu, and Steven I Marcus. Sensitivity analysis and time-cost tradeoffs in stochastic activity networks. In *Proceedings of the 2021 Winter Simulation Conference (WSC)*. IEEE, 2021.

[33] Joel Goh and Nicholas G Hall. Total cost control in project management via satisficing. *Management Science*, 59(6):1354–1372, 2013.

[34] J.G. Cho and Bong-Jin Yum. Functional estimation of activity criticality indices and sensitivity analysis of expected project completion time. *Journal of the Operational Research Society*, 55(8):850–859, 2004.

[35] John E Angus. The probability integral transform and related results. *SIAM review*, 36(4):652–654, 1994.

[36] Charles E Clark. The PERT model for the distribution of an activity time. *Operations Research*, 10(3):405–406, 1962.

[37] Haoxiang Yang and David P Morton. Optimal crashing of an activity network with disruptions. *Mathematical Programming*, 194(1):1113–1162, 2022.

[38] Peng Wan, Michael C Fu, and Steven I Marcus. Crashing of activity networks with disruptions using gradient-based simulation optimization. Working paper.

[39] John R Birge and Francois Louveaux. *Introduction to Stochastic Programming*. Springer Science & Business Media, 2011.

[40] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2021.

[41] Alexander Shapiro and Tito Homem-de Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81(3):301–325, 1998.

[42] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to Linear Optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[43] Halsey L Royden and Patrick Fitzpatrick. *Real Analysis*, volume 32. Macmillan New York, 1988.

[44] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[45] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.