1-1-2012

Sampling based progressive hedging algorithms for stochastic programming problems

Nezir Aydin
Wayne State University,

SAMPLING BASED PROGRESSIVE HEDGING ALGORITHMS FOR STOCHASTIC PROGRAMMING PROBLEMS

by

NEZIR AYDIN

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for degree of

DOCTOR OF PHILOSOPHY

2012

MAJOR: INDUSTRIAL AND SYSTEMS ENGINEERING

Approved By:

____
Advisor Date

____

____

____

DEDICATION

To my Family...

ACKNOWLEDGEMENTS

I would like to thank many people who directly or indirectly helped me with the research presented in this dissertation. First of all, I cannot express enough my gratitude to my advisor Dr. Alper E. Murat. His original ideas, perfectionism, and enthusiasm, have not only directly contributed to the work presented in this dissertation, but have also meant a lot to me personally.

I express my sincere appreciation to the committee members, Dr. Ratna Babu Chinnam, Dr. Leslie Monplaisir and Dr. Boris S. Mordukhovich. I sincerely believe that their valuable inputs to this research make it diverse and successful. Special thanks go to Dr. Leslie Monplaisir, Chair of the Industrial & Systems Engineering Department at Wayne State University, and Dr. Ratna Babu Chinnam, Graduate Chair of the Industrial & Systems Engineering Department at Wayne State University, for their support throughout my doctoral studies and also being excellent examples as supervisors during my teaching assistantship to them.

I am more than grateful to my family and I would especially like to thank my parents for always giving me their unconditional love, support and encouragement throughout the years. I can only hope that I made them proud to complete this far away from home yet rewarding journey. I would also like to thank Lisa M. Corlew and Erkan Isikli for sharing the difficult and happy moments of my life and making me feel at home during Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

CMND:	Capacitated Multi-commodity Network Design
CPLEX:	An optimization software package
CRFLP:	Capacitated Reliable Facility Location
CRFLPAP:	Capacitated Reliable Facility Location and Product Allocation Problem
d-SBPHA:	Discarding-Sampling Based Progressive Hedging Algorithm
LLN:	Law of Large Numbers
LP:	Linear Programming
MINOS:	Modular In-Core Non-Linear Optimization System
MIP:	Mixed Integer Programming
PHA:	Progressive Hedging Algorithm
PMP:	P-Median Problem
SAA:	Sample Average Approximation
SBPHA:	Sampling Based Progressive Hedging Algorithm
SP:	Stochastic Programming
UFLP:	Uncapacitated Facility Location Problem
URFLP:	Uncapacitated Reliable Facility Location Problem

CHAPTER I: INTRODUCTION

Deterministic mathematical programming is an important research field in modeling and analyzing the systems that involve complex decision making which cannot be handled through non-mathematical and non-computational approaches (e.g., intuition, experience based). This research field can be restrictive in its practicality because of the assumption that the model parameters are always known with certainty. For example, in supply chain networks, the supply process is not always certain because of fluctuations or disruptions that might happen during the period of supply. These fluctuations can occur because of seasonality, quality problems, transportation problems or disruptions in supply resources. In production planning, the production capacity is not certain because of the unexpected events that can happen during production, or unexpected changes in production requirements such as specific tools, machines, etc. The deterministic mathematical programming cannot account for these uncertainties and their effect on the solution except in few instances to a certain degree such as in the case of Linear Programming where sensitivity analysis can be employed. Furthermore, deterministic mathematical models do not consider possible future scenarios that are subject to changes in parameter values when optimizing problems (Chiralaksanakul 2003). Therefore, Stochastic Programming (SP) models are introduced as an extension of deterministic mathematical programs in order to deal with uncertain parameters in the system.

SP problems with recourse was first introduced by Dantzig (1955) for mathematical programs with uncertainties. Since then, SP has become one of the most important methods to optimize systems that include uncertain parameters or variables in some or all aspects of the model. The most important assumption in SP is that the probability distributions of the random variables are known. A commonly used objective of the SP is to identify a feasible solution which is optimal for the expected value function over all possible realizations (Solak 2007). There are other objectives such as identification of robust solutions and solutions that optimal with respect to a pre-specified trade-off between the expected value function and its variability.

The most extensively studied SP models are the two-stage models. In the two-stage SP approach, the decision variables are partitioned into two sets. The first stage variables are decided before the realization of the uncertain parameters becomes known. Once the random events have presented themselves, the second stage decisions are made to given the fixed first stage decisions. The objective is to find first stage decisions in a way that the sum of first stage costs and the expected objective function value of the random second stage or recourse costs are minimized or maximized (Ahmed and Shapiro 2002).

To illustrate, consider the two-stage stochastic programming problem for fixed-charge facility location where the decisions are locations of facilities and allocation of demand to the located facilities. The first stage decisions are the locations of facilities subject to the probabilistic knowledge of the demand. After first stage decisions are made and demand realizations are observed, the demand is assigned optimally through the second stage decision.

Two-stage SPs are generalized to multi-stage SP in order to model complex systems that include decisions subject to uncertainty in multiple time stages. The decision and uncertainty realization sequence of two-stage SPs is generalized in multi-stage SPs. In particular, decisions are first made at the beginning of each stage before any of the future realizations are observed. These decisions depend only on the previously observed realizations and decisions made. Following the observation of uncertain realizations, the decisions for the next stage are made. The objective is to make decisions for each stage, sequentially, that optimizes the expected objective function value over all possible realizations. One main difference from two-stage SPs is that the expected recourse functions are recursive. For instance, in lot sizing problems, at the beginning of each stage, the production decisions are made that depend on the previous stages' production decisions and the observed demand. The optimal production amount is decided with only probabilistic knowledge of the future demand scenarios.

In order to determine the size of a stochastic programming problem, the mathematical model's dimensions and the number of realizations of random vectors of the problem need to be considered. If the model's random vectors have continuous distribution or have infinitely many dimensions, then the optimization of such SPs are typically impossible. One alternative is to approximate the uncertainty through scenario aggregation or discretizing the continuous probability distributions. In the multi-stage model, the complexity continues to escalate because in this model the problem size grows exponentially also with the number of decision stages subject to uncertainty.

SPs with small number of scenarios can be optimized with exact solution methods, e.g., by solving the deterministic equivalent of the problem or through primal or dual decomposition methods. When the numbers of scenarios in stochastic programming are too large to optimize with exact solution methods then sampling based methods, such as Sample Average Approximation method (SAA), are needed to approximate the objective function value. SAA solves stochastic programming problems through the Monte Carlo simulation (Kleywegt, Shapiro et al. 2001; Ahmed and Shapiro 2002). In the Monte Carlo simulation a random sample is utilized to find a sample average estimate to approximate the expected objective function. To solve the sample average approximation problem, deterministic optimization techniques are then used.

A commonly used exact solution method to solve large scale stochastic programming problem is the Progressive Hedging Algorithm (PHA). The PHA proceeds by converting the original stochastic problem into its deterministic equivalent. This deterministic equivalent formulation includes non-anticipativity constraints which ensure that decisions in a set of scenarios are identical if those scenarios are undistinguishable up to the time of those decisions. The non-anticipativity constraints are then relaxed using Augmented Lagrangean relaxation and the problem becomes separable by each scenario. The scenario sub-problems have augmented objective functions which include linear and quadratic Lagrange penalty functions corresponding to the relaxed non-anticipativity constraints. These scenario sub-problems are then solved as deterministic problems. At each iteration of the PHA algorithm, solutions from all scenario sub-problems are collected and averaged according to their non-anticipativity constraints and scenario probabilities. The deviation of each scenario sub-problem solution from these averages is used to update the Lagrangian multipliers. Next, the scenario sub-problems are re-solved with the updated augmented Lagrangean objective function. This iterative process continues until the Lagrangean dual problem converges to a solution satisfying the non-anticipativity constraints. It is guaranteed that with the right multiplier values, the solution of scenario subproblems yield optimal solutions. However, in practice this guarantee is not available since the PHA's convergence to the optimal multipliers is not guaranteed.

Optimizing under uncertainty has been extensively studied in many areas other than stochastic programs, such as the Optimal Control Theory, Markov Decision Processes, Statistical Decision Theory and Stochastic Dynamic Programming. Although aforementioned techniques are related to the proposed work in coping with uncertainty, we do not discuss these methods because the theory of these techniques are developed and improved independently from SP literature (Chiralaksanakul 2003).

1.1. Motivation

Stochastic programming is an increasingly popularized methodology in the Operation Research field. It extends the deterministic mathematical programming approach to the problem instances where there are uncertainties in the problem parameters. With the latest optimization algorithms realistic sized deterministic optimization problems can be successfully solved. However, it is still a challenge to solve large scale stochastic programming problems when the number of scenarios is large and the underlying deterministic problem is already large scale and NP-complete.

In today's business environment, companies around the world are struggling with decision making under uncertainty that relate to their supply chain and production planning and often resort to myopic planning or consider few "most likely" scenarios for their long term planning. In the academic literature, large-scale multi-stage stochastic problems arising in practice are rarely tackled successfully to an acceptable level of optimality. In practice, near future decisions are often more important than far future decisions because the uncertainty in far future resolutions are more indeterminate than near future resolutions. Better information is usually available or can be acquired (e.g., at a cost) for planning out near future rather than far future decisions. While there is less uncertainty affecting near-term decisions, the need to consider uncertainty affecting long term decisions are ever increasing. The globalization and increased competition requires companies to be agile and prepared for a wide range of uncertainties in the short as well as medium-to-long term planning. Therefore, in practice, it is often necessary to consider all of the scenarios for near future and the most likely or random sample scenarios for the far future.

Current state of the art in two-stage and multi-stage SP literature is partially able to answer the need for computationally efficient and approximately optimal solution methods. For instance, the SAA procedure is widely applied to two-stage large scale SP programs. The success of the SAA resides on its demonstrated property of increasing the sample sizes, exponentially increases the likelihood of finding the optimal solution. However, the recommended sample sizes are often very conservative and not applicable to practical sized problems. Accordingly, many users of SAA select the sample sizes and number of samples, not based on the recommended threshold values, but rather based on what can be solved with the current deterministic mathematical programming solvers. Hence, there is significant risk of suboptimality due to insufficient sampling. Further, the SAA procedure cannot be applied to the multi-stage problems due to the multiplicatively increasing number of scenarios required with the number of decision stages. Lastly, the SAA procedure can be calibrated through two parameters, i.e., number of samples and sample size and the flexibility in terms of adjusting the computational effort necessary for a given optimality requirement is limited. In contrast, the PHA method can solve SPs (both two-stage and multi-stage) optimally and provides tractability by decomposing large scale problems into smaller, hence solvable, deterministic instances. However, solving a large scale SP with PHA requires solving many through a multitude of iterations. Moreover, at an intermediate step, PHA does not provide a feasible solution that is satisfying all the non-anticipativity constraints. Hence, there is a need to combine the exact methods such as PHA with sampling based methods such as SAA to develop flexible solution methods to tackle large-scale SPs.

1.2. Research Objectives

The goal of this study is to develop efficient and effective hybrid methods for large scale stochastic programming problem based on Progressive Hedging

Algorithm with Sample Average Approximation. The specific objectives are as follows:

1. Developing hybrid method(s) integrating the exact solution method, Progressive Hedging Algorithm (PHA), and the sample based method, Sample Average Approximation (SAA).

2. Demonstrating the effectiveness of proposed methods over the commonly used solution methods, such as, SAA and deterministic equivalent.

The contribution of this research is to provide a configurable solution method which improves the sampling based methods' efficiency and PHA's accuracy for stochastic programs. The attainment of this contribution is demonstrated through the consistency of the solutions found by the proposed hybrid methods.

1.3. Dissertation Organization

The dissertation is organized as follows. Preceding literature on Stochastic Programming (SP) problems, two solution methodologies (Progressive Hedging Algorithm (PHA) and Sample Average Approximation (SAA)) and the mathematical and algorithmic formulations of the existing methodologies are provided in the earlier part of the Chapter 2. In the later part of the Chapter 2, we describe in detail the two proposed algorithms, Sampling Based Progressive Hedging Algorithm (SBPHA) and *d*-SBPHA.

In Chapter 3, we first introduce the Capacitated Reliable Facility Location Problem (CRFLP) which is a two-stage SP and provide a brief literature review on the CRFLP. Next we apply the SBPHA and $d-$SBPHA methods to CRFLP and demonstrate the effectiveness in solving two-stage SPs. Lastly, we apply the *d*-

SBPHA to multi-stage stochastic lot-sizing problem. Experimental results, computational times and the comparisons, in terms of computational effort and the solution quality, of the SBPHA and d-SBPHA with SAA are studied in detail.

In the Chapter 4, analytical and computational analysis (using SBPHA) on the effect of the flexibility to mitigate supply chain disruptions is studied. Conclusions and future studies are given in Chapter 5 of the dissertation.

CHAPTER II: NEW HYBRID ALGORITHMS FOR STOCHASTIC PROGRAMMING PROBLEMS

2.1. Stochastic Programming

Stochastic Programming (SP) is an increasingly growing field for solving mathematical programming problems subject to parameter uncertainty (Kall and Wallace 1994; Ruszczynski and Shapiro 2003). The main assumption in SP is that the probability distributions of the random events are known or can be approximated. The objective of SP is to identify a feasible solution that optimizes the expected value of a function over all possible realizations of the random events.

An extensive number of solution methods are proposed for solving SP problems. These solution methods can be classified into two classes; exact or approximation. Analytical solution methods and computational methods that solve SP algorithmically without approximating the objective function value to the optimal are considered as exact solution techniques. Solution techniques that approximate the optimal objective function value are considered as approximation methods. In this section we briefly review these two types (exact and approximation) of methods.

Exact solution methods provide optimal solution and objective function value. If random variable set is finite with a moderately small number of realizations, so called "scenarios" (Rockafellar and Wets 1991), SP can be modeled as a deterministic equivalent program and solved to optimally by an optimization algorithm. Small sized linear SPs can be solved by simplex algorithm through the deterministic equivalent formulation of the SP. If the problem is mixed integer program then it can easily be solved by a branch and bound (or another divide-and-conquer based) algorithm applied to the deterministic equivalent formulation of the SP. However, if the number of scenarios is very large, generic linear programming techniques are not able to solve because of the large size of deterministic equivalent problems. Some decomposition algorithms are proposed to solve stochastic linear programming problems that have a modest number of scenarios.

Decomposition methods that are used for solving SP are categorized either by stage-based decomposition or scenario based decomposition methods. The most widely known stage-based decomposition method is L-shaped method proposed by Slyke and Wets (1969); (Birge and Louveaux 1997). In the stage-based decomposition method, each stage t has a number of sub-problems that is related to each node at time stage t. In this thesis, our focus is on the scenario based decomposition (Lagrangean-based decomposition) methods and thus we will not discuss the stage-based decomposition methods further

The goal of Lagrangean decomposition is the same as other decomposition methods, which is to decompose the complex problem into sub-problems to be able to solve them more efficiently. In the scenario based decomposition method, the non-anticipativity constraints are relaxed by Lagrangean penalty terms. Once the problem is decomposed, then each scenario becomes a deterministic problem to be solved. The disadvantage of this method is the computational time since the method iteratively solves the scenario subproblems and the convergence could be slow. For detailed explanation reader is referred (Rockafellar and Wets 1991; Mulvey and Ruszczynski 1995; Rosa and Ruszczynski 1996).

When the number of scenarios is large and exact solution methods fail to solve the problem, approximation methods are then used to estimate the objective function value. As an approximation method, Monte Carlo Sampling-based method is widely used. A wide survey on Monte Carlo sampling in SP can be found in Morton and Popova (2001). Monte Carlo sampling can be applied to SP either as interior or exterior of the algorithm (Chiralaksanakul 2003). In the internal sampling method, Monte Carlo estimates can be replaced with the difficult exact computations during the solution process (Higle and Sen 1991; Infanger 1992). If a scenario tree is constructed through Monte Carlo sampling and the objective function value of the original problem is approximated at the onset, then this approach is referred to as external sampling method. External sampling is also called Sample Average Approximation (SAA) (Shapiro 2002). We cover the SAA which is used in the proposed hybrid algorithms.

One of the motivation of this dissertation is that the decomposition and the approximation algorithms can be used together to solve stochastic programming problems efficiently and effectively. For instance, when the number of the scenarios in a SP is very large, then the problem can be approximated by an approximation method and the approximated problem can then be solved by the decomposition methods.

In the next two subsections, we present the mathematical formulation of the two-stage SP and multi-stage SP problems and review the related literature.

2.1.1. Two-Stage Stochastic Programming

The two-stage stochastic programming models are the most common models in SP. In the two-stage model, the first stage decisions are made prior to the realization of the uncertainty. In the second stage, the uncertainty is realized and second stage decisions are made given the first stage decision and realizations. The goal of the two-stage stochastic programming is to optimize the expected objective value of the second stage decisions' objective function and the objective of the first stage decisions.

A typical formulation of class of the two-stage SP is as follows(Kall and Wallace 1994; Birge and Louveaux 1997; Ahmed and Shapiro 2002):

$$\mathrm{Min}_{x\in X}\{g(x) := c^T x + \mathbb{E}[\mathcal{Q}(x,\xi)]\}, \tag{2.1}$$

where

$$\mathcal{Q}(x,\xi) := \inf_{y\in Y}\{q^T y : Wy \geq h - Tx\} \tag{2.2}$$

is the optimal value and $\xi := (q, T, W, h)$ denotes vector of parameters of the second stage problem. The vector of second stage ξ represents the randomness in some or all of the second stage parameters. The expectation of (2.1) is taken with respect to the known probability distribution of ξ. The problem (2.1) decides on the first stage variables, $x \in \mathbb{R}^{n_1}$, prior to a realization of ξ, and problem (2.2) decides on the second stage variables, $y \in \mathbb{R}^{n_2}$, given the first stage decision and a realization of ξ.

Many solution approaches are proposed in literature for the two-stage SP problems: Benders decomposition, L-shaped method, Progressively Hedging

Algorithm. Benders decomposition technique generates a split between first stage and second stage decision variables (Benders 1962; Slyke and Wets 1969). This method solves a first stage master problem to find a good solution for the first stage, while optimistically estimating the objective function value of the second stage sub-problems. If the first stage decision variables are infeasible then a “feasibility” cut is added to the master problem. Furthermore, if the objective function value of the sub problems is not optimal, then an “optimality” cut is added to the master problem. Benders decomposition method repeats this iterative process until no cut is necessary.

Another solution approach to the two-stage SP problems is the L-shaped method. Louveaux and Vlerk (1993) studied a two-stage SP with integer second stage recourse problem. Second stage problem minimizes the penalty cost that results in the shortage and excess first stage decisions. Laporte and Louveaux (1993) studied a two-stage SP problem with binary first stage decision variables.

A generalization of the integer L-shaped method is done by Caroe and Tind (1997) via duality theory. Penuel, Smith et al. (2010) introduced an integer decomposition algorithm for solving two-stage SP problem with second–stage activation costs. They applied the proposed algorithm to a scenario based facility location problem and compared to the Laporte and Louveaux (1993) in terms of performance.

The other two popular methods for the two-stage SPs are Progressive Hedging Algorithm (PHA) and SAA which are discussed in detail in the next section.

2.1.2. Multi-Stage Stochastic Programming

Multi-stage SP problems generalizes the two-stage SPs. In the multi-stage SP models, the decision at any time stage is made after the realization of random vectors and decision variables of the previous time stages. The goal of the multi-stage model is to make decisions for different time periods in a sequence, while optimizing the expected objective function value of the current and future stages. In the multi-stage model, the size of the SP grows exponentially with the number of time horizons. If the number of scenarios is small, then multi-stage SPs can be optimized with exact solution methods.

In multi-stage SP problems, a T-stage problem is considered and a series of decisions, $\{x_t\}_{t=1}^{T}$, is made with respect to random events $\{\xi_t\}_{t=1}^{T}$. The decision $x_t \in \mathbb{R}^{n_t}$ at time stage t, is made based on the information of the previous stages' decisions, $x_1, \dots, x_{t-1}$, and the observed random events, $\xi_1, \dots, \xi_t$, while optimizing the objective function, $f_t(x_1, \dots, x_{t-1}, \xi_1, \dots, \xi_{t+1})$. Decision variables x_t are subject to constraints and may depend on $x_1, \dots, x_{t-1}$ and $\xi_1, \dots, \xi_t$.

Using the notation and the mathematical representation, used by (Solak 2007), of the multi-stage SP problem with recourse can be formulated as follows:

$$\min\ f_1(x_1) + E_{\xi_1}[\min f_2(x_1, x_2, \xi_1) + E_{\xi_2|\xi_1}[\min f_3(x_1, x_2, x_3, \xi_1, \xi_2)$$

$$+ \cdots + E_{\xi_{T-1}|\xi_1,\dots,\xi_{T-2}}[\min f_T(x_1, \dots, x_T, \xi_1, \dots, \xi_{T-1})] \dots]] \quad (2.3)$$

s.t.

$$g_1(x_1) \leq 0$$

$$g_2(x_1, x_2, \xi_1) \leq 0$$

$$\vdots \qquad (2.4)$$

$$g_T(x_1, \dots, x_T, \xi_1, \dots, \xi_{T-1}) \leq 0$$

$$x_t \in x_T, t = 1,2, \dots, T \qquad (2.5)$$

where x_t and ξ_t are decision variable and random event vectors, respectively, so that $x_t \in \mathbb{R}^{n_t}$, $\xi_t \in \mathbb{R}^{m_t}$ and $t = 1,2, \dots, T$. Then $f_t: \mathcal{R}^{n_1+\cdots+n_t} \times \mathcal{R}^{m_1+\cdots+m_{t-1}} \rightarrow \mathcal{R}$ and $g_t: \mathcal{R}^{n_1+\cdots+n_t} \times \mathcal{R}^{m_1+\cdots+m_{t-1}} \rightarrow \mathcal{R}^{d_t}$.

If number of random events are finite and S denote the set of all possible realizations of the random events (i.e., scenarios), then the deterministic equivalent of the multi-stage SP (2.3)-(2.5) can be easily formulated as follows:

$$\min \textstyle\sum_{s \in S} p_s [c_1^s x_1^s + c_2^s x_2^s + \cdots + c_T^s x_T^s] \qquad (2.6)$$

s.t.

$$A_{11}^s x_1^s \leq b_1^s \;\; \forall s \in S$$

$$A_{21}^s x_1^s + A_{22}^s x_2^s \leq b_2^s \;\; \forall s \in S$$

$$\vdots \qquad (2.7)$$

$$A_{TT-1}^s x_{T-1}^s + A_{TT}^s x_T^s \leq b_T^s \;\; \forall s \in S$$

$$x_t^s - x_t^{s'} = 0 \; \forall s, s' \in S: (\xi_1^s, \dots, \xi_t^s) = \left(\xi_1^{s'}, \dots, \xi_t^{s'}\right), t = 1,2, \dots, T \qquad (2.8)$$

$$x_t^s \geq 0 \; \forall s \in S, t = 1,2, \dots, T \qquad (2.9)$$

In the formulation given above p_s represent the occurrence probability of scenario s and objective function and constraints are assumed to be linear.

Constraints (2.8), are the non-anticipativity constraints, which ensure the equivalence of the decision variables that are made at time stage t for all scenarios that have the same history. Non-anticipativity constraints ensure that the decisions are implementable (Higle 2005; Solak 2007).

The multi-stage SP formulation in (2.6)-(2.9) is scenario based formulation. Tree representation of the random events ξ_t is shown in Figure 1. Each path from root node to the last nodes represents a scenario in the scenario tree.

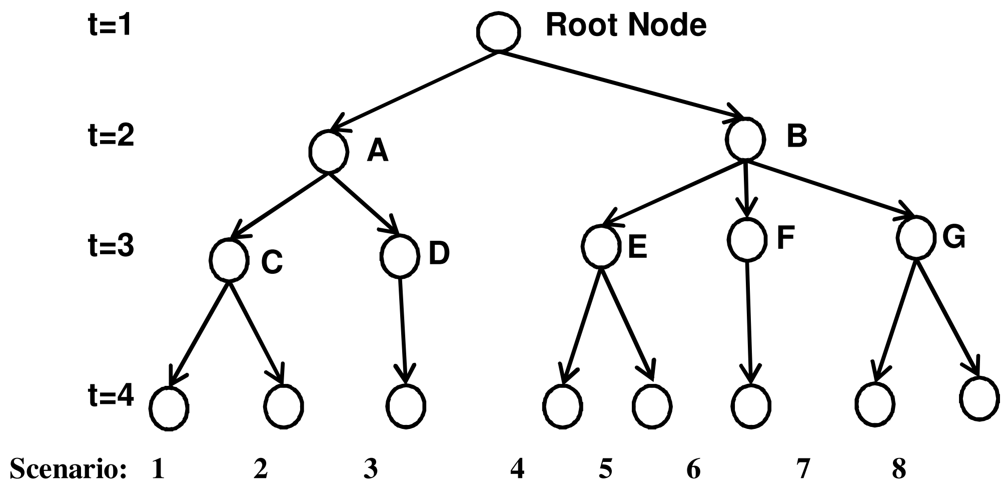


Figure 1: An example of four-stage scenario tree

Table 1: Random events and decision variables for scenario three in Figure 1

Node:	Root	A	B	C	D	E	F	G
Random Event	ξ_1^1	ξ_2^1	ξ_2^2	ξ_3^1	ξ_3^2	ξ_3^3	ξ_3^4	ξ_3^5
Decision variables	x_1^1	x_2^1	x_2^2	x_3^1	x_3^2	x_3^3	x_3^4	x_3^5

The non-anticipativity implies that the decision variables at the root node have to be same for all scenarios $s = 1,2,\dots,8$, decision variables taken at node A have to be same for scenarios 1,2, and 3 and at node C have to be same for 1 and 2. The similar logic applies to other scenarios.

While some efficient solution methods for multi-stage linear programming SPs have been developed in the stochastic programming literature, most of these solution methods are not efficient when applied to the multistage stochastic integer programming problem. This is because of the non-convexities caused by the integer decision variables. (Solak 2007).

Rosa and Ruszczynski (1996) propose two augmented Lagrangean techniques for the multistage SP problems: decomposing the problem into nodes depending on stages and decomposing the problem by the scenarios. Another solution method is to implement Benders decomposition algorithm recursively to the nested decomposition introduced by Birge (1985). Rockafellar and Wets (1991) introduced a new solution method named Progressive Hedging Algorithm. The main idea of this algorithm is to decompose the problem into sub-problems based on scenarios and solving these sub-problems iteratively while applying non-anticipativity constraints in a novel manner. More detailed information on PHA is provided in the Progressive Hedging Algorithm section. Sampling based approaches for multi-stage SPs are also used to calculate lower bounds on the optimal value of objective function (Kleywegt, Shapiro et al. 2001; Wang 2007).The detailed description is provided in the SAA section.

We now describe in detail the PHA and SAA integrated to obtain the proposed Sampling Based Progressive Hedging Algorithm (SBPHA).

2.2. Exact and Sampling Methods: PHA and SAA

In this section, we review and present the algorithmic formulation of the two algorithms that are used to develop the hybrid SBPHA algorithm. First, method is the exact solution method, Progressive Hedging Algorithm (PHA), and the second one is the Monte-Carlo sampling based approximation method, Sample Average Approximation (SAA).

2.2.1. Progressive Hedging Algorithm (PHA)

When the SPs are very large or the underlying mixed-integer problem is difficult, then solving the SP as a single mathematical programming problem is impractical due to memory and computational time restrictions. Often, decomposition based methods are used to divide the problem into smaller and more manageable subproblems. These subproblems are then solved iteratively while enforcing those aspects of the problem relaxed for decomposition. One such decomposition method is the PHA proposed by (Rockafellar and Wets 1991). PHA decomposes SPs by scenarios rather than by time stages. PHA converges to the optimal solution when SPs convex programs. For the case when the decisions are integer, the PHA is used as a heuristic method (Lokketangen and Woodruff 1996; Fan and Liu 2010; Watson and Woodruff 2011).

PHA was first introduced by Rockafellar and Wets (1991) to solve SPs. It was the first rigorous algorithmic procedure that worked to put a policy in place for a variety of scenarios. Rockafellar and Wets (1991) introduced theoretical and mathematical discussion on PHA which was constructed based on the principle of scenario aggregation (Wallace and Helgason 1991). Wallace and Helgason (1991) studied computational performance on implementing the solution methodology of solving scenario problems approximately and individually using Lagrangean approach in scenario aggregation procedure. They showed that the proposed procedure does not need to increase the number of iterations to solve the sub-problems more accurately. However, their method needed improvements in terms of the speed of convergence.

In 1991, Mulvey and Vladimirou (1991) applied PHA to solve multi-stage stochastic networks as an application of scenario aggregation. They also applied the technique to stochastic generalized networks. In their problem the uncertainty is in the constraint coefficients. They tested the performance of PHA by applying to the stochastic generalized networks. In most cases they found a better solution than other researchers by using MINOS with medium sized problems. Also, in their literature review of financial risk management, Mulvey, Rosenbaum et al. (1997) proposed PHA as an efficient methodology to deal with multi-stage stochastic programming in the risk management field. Mulvey and Vladimirou (1992) proposed using PHA as an proficient method in financial planning problems. Takriti, Birge et al. (1996) report on favorable results from applying the PHA to stochastic multi-stage commitment problems. They studied more general cases in which any of the data could be stochastic and where there could be varying number of integer variables. Robinson (1991) applied a new methodology to portfolio optimization problems to extend the scenario analysis technique. His proposed research is different than the algorithm proposed by Rockafellar and Wets (1991) in terms of having the chance to put non-separable convex constraints instead of non-anticipativity constraints into problem.

Takriti, Birge et al. (1996) applied Lagrangean relaxation to mixed-integer SP production problem. In their study, they proposed a mathematical model and an efficient solution methodology. In their comparison, they concluded that pure Lagrangean relaxation needs many iterations to converge, on the other hand, PHA converges much faster than pure Lagrangean relaxation methodology.

Table 2: Summary of literature review on Progressive Hedging Algorithm (PHA)

Reference	Summary
Rockafellar and Wets (1991)	Introduced PHA through this research and showed that convergence to optimization is reached to optimal (depending on the convergence criteria) both for convex and non-convex types of problems.
Wallace and Helgason (1991)	Introduced generic procedures and definitions on PHA. Showed that using a scenario tree in computation is superior.
Helgason and Wallace (1991)	Proposed methodology solves scenario problems approximately and individually using Lagrangean approach in scenario aggregation procedure. For accurate solutions, incremental in number of scenario is not needed. However, the method needed improvements in terms of the speed of convergence.
Mulvey and Vladimirou (1991)	Tested the performance of PHA by applying to stochastic generalized networks. In most cases, finds a better solution than other published results. Used MINOS for their computations of medium sized problems.
Lokketangen and Woodruff (1996)	Introduced a general purpose to improve solutions to multi-stage stochastic mixed integer (0,1) by using Tabu search and PHA .
Haugen, Lokketangen et al. (2001)	Applied PHA to minimize multi-stage stochastic mixed integer problems. Optimized each sub problem (scenario) by dynamic programming.
Woodruff and Vob (2006)	Proposed a heuristic method to reduce problem size. The method is to solve scenario sub-problems and then combine these solutions to determine the values to binary variables.
Robinson (1991)	Proposed research is different than the algorithm proposed by Rockafeller and Wets (1991) in terms of having chance to put non-separable convex constraints other than non-anticipativity constraints into the problem.

Haugen, Lokketangen et al. (2001) applied PHA as a meta-heuristic by solving each sub-problem that developed from each scenario heuristically while working with multi-stage lot-sizing problems. In their study, they applied an exact solution method instead of using an approximation algorithm. (Woodruff and Vob 2006) applied the PHA to production planning in supply chain disruptions. In this mixed integer stochastic problem binary variables were considered as indicators for production. The goal of the problem was to minimize total cost. They developed a heuristic method to increase the performance of algorithm. Crainic, Fu et al. (2009) proposed Tabu search and PHA based heuristic method to the two-stage capacitated multi-commodity network design (CMND) problem with stochastic demand. The objective of the problem is to optimize the cost of designing the first stage and expected distribution cost acquired in second stage. Detailed information on behavior of the PHA methodology can be found in (Wallace and Helgason 1991), (Mulvey and Vladimirou 1991), (Lokketangen and Woodruff 1996), (Crainic, Fu et al. 2009), and (Watson and Woodruff 2011).

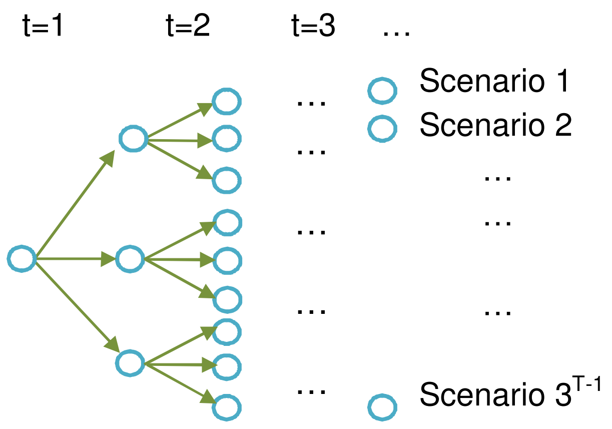


Figure 2: A Multistage Stochastic Tree Having T Stages and 3 Branches Per Stage

In implementing the PHA, the first step is to create a scenario set, such as the example shown in Figure 1 and Figure 2. In Figure 1, each path (from root node to leaf node) is a scenario where the number of scenario sets is equal to S (s denotes an individual scenario) and each scenario includes a full collection of random variable realizations with a probability. Each scenario s is considered as a deterministic problem which can be solved using exact solution methods.

To decompose the problem by scenarios, the PHA relaxes the non-anticipativity constraints at each node. These non-anticipativity constraints are then enforced during the solution procedure through the Augmented Lagrangian terms. The augmented Lagrangian terms are calculated using the dual variables and the mean of the solutions, e.g., probability averaged decision variables, at each node. Dual variables for each relaxed non-anticipativity constraints are updated using the mean solutions. The PHA convergence is based on the rate of change in the subgradient, i.e, the deviation of scenario solutions for each stage from the mean of the solutions. As noted in (Haugen, Lokketangen et al. 2001), solving the scenario subproblems to optimality is not necessary and that approximating methods can be applied.

We now illustrate the PHA for the two-stage SP in (1)-(2). Given a finite number of realizations ξ_s, $s = 1, \dots, S$, and a positive occurrence probability $\mathrm{p_s}$, such that $\sum_{s=1}^{S} \mathrm{p_s} = 1$, the set $\{\xi_s, \dots, \xi_S\}$,of scenarios, with the corresponding probabilities $\mathrm{p_s}, \dots, \mathrm{p_S}$, can be considered as a representation of the probability distribution. The expected value function $\mathbb{E}[\mathcal{Q}(\mathrm{x}, \xi)]$ can be expressed as the finite summation

$\mathbb{E}[\mathcal{Q}(x,\xi)] = \sum_{s=1}^{S} p_s \mathcal{Q}(x,\xi_s)$. By creating a copy of the second stage decision vector, y_s, for every scenario ξ_s, i.e. by considering $y_s = y(\xi_s), s = 1, \dots, S$, as a function of scenarios, we can write the two-stage SP problem (1)-(2) as (Shapiro 2008):

$$\min_{x, y_s, \dots, y_S} c^T x + \sum_{s=1}^{S} p_s g(x, y_s, \xi_s) \qquad (2.10)$$

$$x \in X, y_s \in \mathcal{Q}(x, \xi_s), s = 1, \dots, S$$

where $\xi_s := (q_s, T_s, W_s, h_s), s = 1, \dots, S$, are the corresponding scenarios (Shapiro, 2008). Each scenario sub-problem can be expressed as follows:

$$\text{Min } (c^T x) + \sum_{s \in S} p_s (f_s y_s) \qquad (2.11)$$

$$\text{s.t} \qquad (x, y_s) \in \mathcal{Q}(x, \xi_s)\ \forall s \in S$$

where x is not scenario specific and $x_s = x, \forall s \in S$. The y_s represent second stage decision variables which are determined with a given first stage decision (x) and a particular random point, ξ_s, and f_s. The f_s represents second stage scenario specific coefficient vectors. Problem (2.11) is the well-known extensive form of a two-stage stochastic program (Watson and Woodruff 2011).

The following is the pseudo-code of the PHA algorithm (Watson and Woodruff 2011). Let ρ be a penalty factor($\rho > 0$), and ϵ be a convergence threshold.

PHA Algorithm for Two-Stage SP

```
1:  $k := 0$
2:  For all $s \in S, x_s^k := \text{argmin}_{x, y_s}(cx + f_s y_s): (x, y_s) \in \mathcal{Q}(x, \xi_s)$
3:  $\bar{x}^k := \sum_{s \in S} p_s x_s^k$
4:  For all $s \in S, w_s^k := \rho(x_s^k - \bar{x}^k)$
```

```
5:  $k \coloneqq k + 1$
6:  For all $s \in S$, $x_s^k \coloneqq \mathrm{argmin}_{x,y_s}\left(cx + \omega_s^{k-1}x + \frac{\rho}{2}\left\|x - \bar{x}^{k-1}\right\|^2 + f_s y_s\right) : (x, y_s) \in \mathcal{Q}(x, \xi_s)$
7:  $\bar{x}^k \coloneqq \sum_{s \in S} p_s x_s^k$
8:  For all $s \in S, \omega_s^k \coloneqq \omega_s^{k-1} + \rho\left(x_s^k - \bar{x}^k\right)$
9:  $\pi^k \coloneqq \sum_{s \in S} p_s \left\|x_s^k - \bar{x}^k\right\|$
10: If $\pi^k < \epsilon$, then go to step 5. Otherwise, terminate.
```

The index k shows the Iteration number; and Euclidian distance in iteration k is expressed by π^k. The vectors x_s^k, $\bar{x}^k$, and ω_s^k show decisions for scenario s in iteration k, weighted (average) decisions of the problem in iteration k, and dual multiplier for scenario s at iteration k, respectively. PHA converges to a common $\bar{x}$ in linear time if decision vector x is continuous (Watson and Woodruff, 2011). However problem becomes much more complex to solve when x vector is integer i.e., binary.

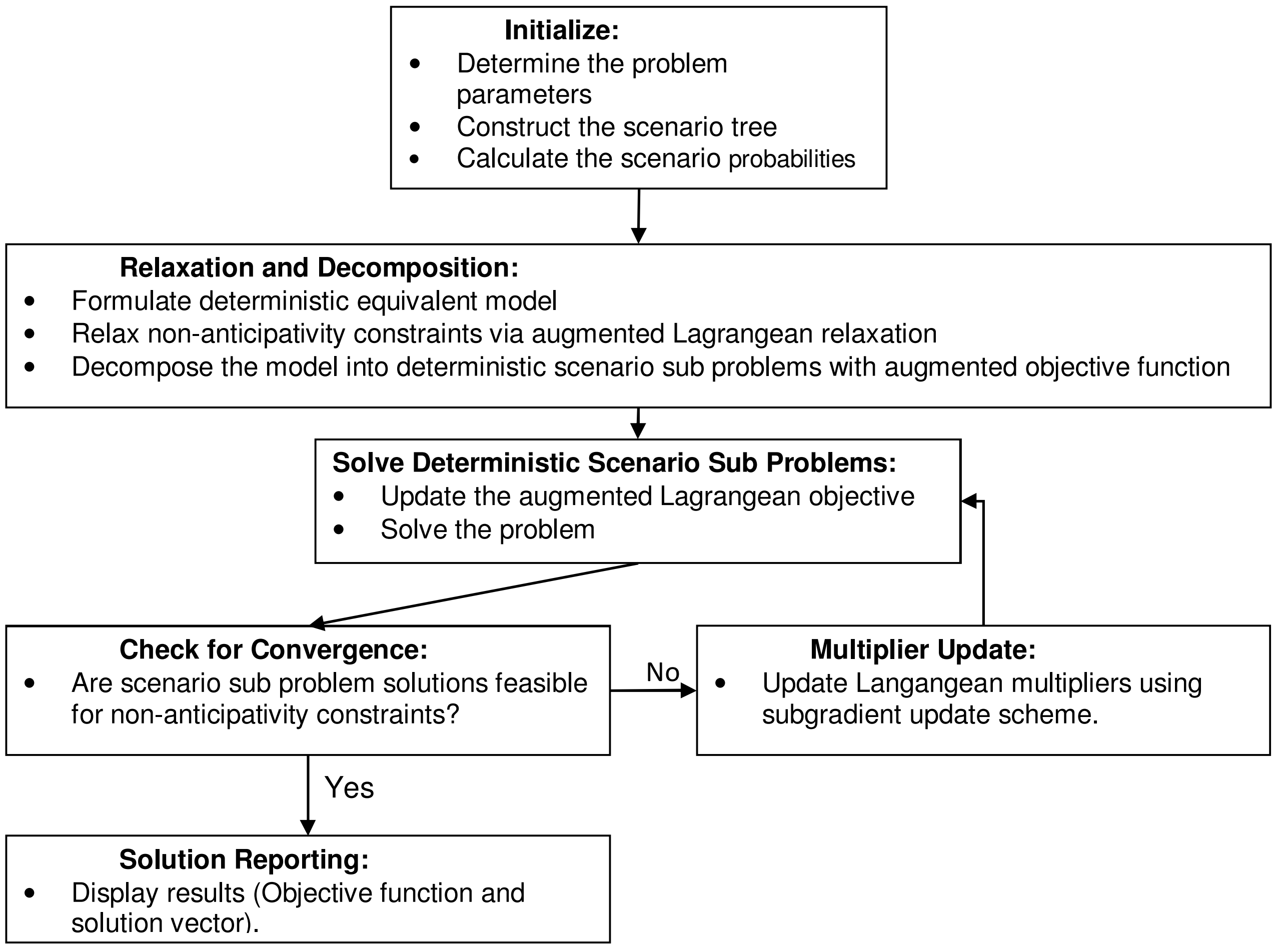

Initialize:
Determine the problem parameters
Construct the scenario tree
Calculate the scenario probabilities
Relaxation and Decomposition:
Formulate deterministic equivalent model
Relax non-anticipativity constraints via augmented Lagrangean relaxation
Decompose the model into deterministic scenario sub problems with augmented objective function
Solve Deterministic Scenario Sub Problems:
Update the augmented Lagrangean objective
Solve the problem
Check for Convergence:
Are scenario sub problem solutions feasible for non-anticipativity constraints?
No
Multiplier Update:
Update Langangean multipliers using subgradient update scheme.
Yes
Solution Reporting:
Display results (Objective function and solution vector).

Figure 3 Flow Chart of the Progressive Hedging Algorithm (PHA)

2.2.2. Sample Average Approximation (SAA)

Sampling based methods are usually used when the stochastic problem is too large to solve by exact solution techniques. With the sampling based approaches, the objective function is approximated through a random sample of scenarios.

Typically sampling based approaches are classified into two: Interior sampling and exterior sampling methods (Verweij, Ahmed et al. 2003). In interior sampling methods, sampling is performed inside a chosen algorithm with new (independent) samples generated during the iterative solution process. These samples can be generated through a resampling from the entire scenario set or by selecting subsets of the previously selected samples. There are several studies using interior sampling based approaches. Higle and Sen (1991) developed a stochastic decomposition algorithm for two-stage stochastic programming problems. Infanger (1992) studied statistical L-Shaped method. An interior sampling based branch and bound method is introduced by Norkin, Pflug et al. (1998) for discrete stochastic problems.

In the exterior sampling approach, a sample of scenarios is generated from possible realizations, and then deterministic optimization problem is developed from the generated samples and solved. This procedure (generating samples and solving deterministic problems) repeated several times. SAA is one of the exterior type sampling based method.

SAA can be explained through a number of steps; random samples are generated, a sample average function is applied to the selected random samples to approximate the expected value function. The optimization problem that is developed is solved iteratively until the stopping criterion is satisfied. Since SAA, using exterior sampling, splits sampling part from optimization part (Rubinstein and Shapiro 1990), applying SAA method is easier. Some advantages of SAA can be listed as (Shapiro (2005): Ease of numerical implementation, often one can use existing software, good convergence properties, well developed statistical inference (validation and error analysis, stopping rules), easily amendable to variance reduction techniques and ideal for parallel computations.

The main idea of Sample Average Approximation (SAA) approach can be explained as follows. A sample $\xi_1, \dots, \xi_N$ of N realizations of the random vector ξ is generated, and consequently the expected value function $\mathbb{E}[\mathcal{Q}(\mathrm{x}, \xi)]$ is approximated by the sample average function $\mathrm{N}^{-1}\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathcal{Q}(\mathrm{x}, \xi)$. The obtained sample average approximation $\mathrm{Min}_{\mathrm{x}\in\mathrm{X}}\{\hat{\mathrm{g}}_{\mathrm{N}}(\mathrm{x}) \coloneqq \mathrm{c}^{\mathrm{T}}\mathrm{x} + \mathrm{N}^{-1}\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathcal{Q}(\mathrm{x}, \xi_{\mathrm{n}})\}$, of the stochastic program (2.1) is then solved by a deterministic optimization algorithm.

<u>SAA Procedure:</u>

Initialize: Generate M independent random samples $m = 1,2, \dots, M$ with scenario sets N_m where $|N_m| = N$. Each sample m consists of N realizations of independently and identically distributed (i.i.d.) random scenarios. We also select a reference sample which is sufficiently large, e.g., $|N'| \gg N$.

Step 1: For each sample m, solve the following two-stage SP problem and record the sample optimal objective function value v^m and the sample optimal solution x^m.

$$Min_{x\in X}\left\{c^T x + \frac{1}{|N_m|}\sum_{n=1}^{N_m} \mathcal{Q}(x, \xi_n)\right\} \tag{2.12}$$

Step 2: Calculate the average $\bar{v}^M$ of the sample optimal objective function values obtained in Step 1 as follows.

$$\bar{v}^M = \frac{1}{M}\sum_{m=1}^{M} v^m \tag{2.13}$$

Step 3: Estimate the true objective function value $\hat{v}^m$ of the original problem for each sample's optimal solution. Solve the following problem for each sample using the optimal first stage decisions x^m from step 1.

$$\hat{v}^m = \mathrm{Minimize}\sum c^T x^m + \frac{1}{|N'|}\sum_{s=1}^{N'} \mathcal{Q}(x^m, \xi_s) \tag{2.14}$$

Step 4: Select the solution x^m with the best $\hat{v}^m$, i.e. $x^{SAA} = argmin_{m=1,\dots,M}$ as the solution and $v^{SAA} = \min_{m=1,\dots,M} \hat{v}^m$, as the solution value of SAA.

Let v^*denote the optimal objective function value of the original problem (2.1)-(2.2). The $\bar{v}^M$ is an unbiased estimator of $\mathbb{E}[v]$ which is the expected optimal objective function value of sample problems. Since $\mathbb{E}[v] \leq v^*$, the $\bar{v}^M$ provides a statistical lower bound on the v^* (Ahmed and Shapiro 2002). Each $\hat{v}^m_{m=1,\dots,M}$ provides statistical upper bound for v^*. Kleywegt, Shapiro et al. (2001) and Shapiro (2003) discussed that the optimal value of SAA problem converges to optimal value of "true" problem with probability of one under condition of $N \to \infty$. Clearly, selecting larger sample size will provide better approximation (statistically speaking). Since larger sample size causes computational complexity, solving SAA several times with smaller independent and identically distributed samples is easier.

Another important estimator that needs to be considered for the quality of the solution is the estimator of the variance of the optimality gap. The estimates of the variances for $v^*$and $\bar{v}^M$can be calculated by (2.15) and (2.16), respectively.

$$\hat{\sigma}^2_{v_N^M} = \frac{1}{M(M-1)} \sum_{m=1}^{M} \left(v^m - \bar{v}^M\right)^2 \tag{2.15}$$

$$\hat{\sigma}^2_{\hat{v}^m} = \frac{1}{N'(N'-1)} \sum_{n=1}^{N'} \left(Q(x^m, \xi_n) - \hat{v}^m\right)^2 \tag{2.16}$$

The estimate of the optimality gap of a candidate solution can be calculated via $\hat{v}^m - \bar{v}^M$, and the variance of the optimality gap can be calculated by $\hat{\sigma}^2_{v_N^M} + \hat{\sigma}^2_{\hat{v}^m}$.

Quality of a solution to stochastic programming based sampling methods depends on several criteria such as, sample size, convergence rate, and stopping rules (criterion). Detailed discussion on the quality of a solution, which is found via sampling methods, for stochastic programs can be found in (Bayraksan and Morton 2009). Bayraksan and Morton (2009) in their tutorial introduced a procedure that shapes an interval estimator on the optimality gap of a certain solution. They provide methods reducing the variance and computational effort of the estimator they introduced. Also, they discussed ways to increase sample size without hurting computational effort in a smart way what they call “sequential sampling procedure”. Researches similar to the sequential sampling procedure are done both in simulation and statistics (Chow and Robbins 1965; Law, Kelton et al. 1981; Law and Kelton 1982).

If the decision variables in (2.1) and (2.2) are continuous, it has been proven that an optimal solution of the SAA problem provides exact solution of the true problem with probability approaching one exponentially fast as N increases (Shapiro and Homem-de-Mello 2001; Ahmed and Shapiro 2002). Many studies are conducted to determine the required sample size.

Ahmed and Shapiro (2002) proposed using a decomposed branch and bound algorithm. The algorithm proposed studies to partition the search space by creating approximate problems with sampling method and solving by a novel optimization algorithm. The quality of the solution converged is described with statistical and deterministic bounds with fixed sample sizes.

Kleywegt, Shapiro et al. (2001) applied the SAA method to stochastic discrete optimization problems, i.e., knapsack problem. They noted that the complexity of the SAA methods usually increases exponentially, at least linearly, in terms of sample size selected. Selecting the sample size is needed to consider the tradeoff between the bounds on the optimality gap and the quality of an optimal solution of a SAA problem and the computational performance. They expressed that selecting sample size should dynamically change depending on the previous results that were computed and the more proficient gap estimator of the approximated value function improves the performance of SAA method applied to the algorithm.

Recent studies on SAA are usually focused on improving the quality of a solution obtained from SAA and improving computational effort of SAA by combining meta-heuristic methods especially in multi-stage stochastic mixed integer programming problems (Solak 2007). For more detailed information on the SAA, we refer the reader to several key articles (Kleywegt, Shapiro et al. 2001; Shapiro and Homem-de-Mello 2001; Ahmed and Shapiro 2002; Shapiro 2002; Ruszczynski and Shapiro 2003; Verweij, Ahmed et al. 2003).

2.3. Proposed Algorithms for Two-Stage Stochastic Programming Problems

In proposed algorithm, SBPHA, we hybridize an exact solution method, Progressive Hedging Algorithm (PHA), and an external sampling-based approximation algorithm, Sample Average Approximation (SAA), to efficiently solve two-stage SP problems. While the standard SAA procedure is effective with sufficiently large samples, the required sample size can be quite large for the desired confidence level. Further, the SAA procedure selects the best performing sample solution and discards the remaining sample solutions which contain valuable information about the problem's uncertainty.

The main idea of the proposed hybrid method is to re-use all the information embedded in sample solutions by iteratively solving the samples with an Augmented Lagrangian penalty term to find a common solution that all samples agree on.

2.3.1. Sampling Based Progressive Hedging Algorithm (SBPHA)

The proposed algorithm is a hybridization of the SAA and PHA. The motivation for this hybridization originates from the final step of the SAA method (Step 4, in SAA) where the best performing solution is selected and the rest of the solutions are discarded. However, this discarding of $(M-1)$ sample solution is a loss of valuable sample information as well as loss of effort spent in solving each sample's problem. Let's consider the implementation of the classical SAA procedure in the context of PHA and treat each sample solution as a scenario. Then implementation of SAA would correspond to iterating the sample solutions in PHA only once and then selecting the best performing sample. In the PHA, however, the scenarios are sustained with the scenarios recent solution as well as the weighted solution of all scenarios. Hence, in the proposed SBPHA approach, we modify the SAA method by continuing the solution of the sample problems while enforcing probability weighted solution of the samples and the best performing solution in original problem. The underlying premise of this hybridization is that, by starting with sufficient number of samples (representative of the entire scenario set), the continued iteration of the SAA method, with implementing probability weighted and the best performing solutions via augmented Lagrangian penalty concept would converge the sample solutions to the optimal solution of the original problem.

An important distinction of the SBPHA from classical PHA is the sampling concept and the size of the samples. Classical PHA considers the entire scenario set, solves a problem for each scenario one by one at every iteration, and evaluates the probability weighted solution for the original problem. In comparison, the SBPHA solves only a few numbers of samples which entail multiple scenarios, and then determines the probability weighted in a different way than PHA (will be explained in algorithm section in detailed) and the best performing solution (so far found at any iteration), at every iteration. Solving sample problems is more difficult than solving one scenario problem but considerably easier than to solve original problem.

We first present the proposed SBPHA algorithm and then describe its steps in detail. For clarity, we give notation before describing the algorithm's steps.

<u>Notation for SBPHA:</u>

k, k_{max} : iteration index and maximum number of iterations

$P_m, \widehat{P}_m$: probability and normalized probability of realization of sample m

$x^{m,k}$: solution vector for sample m at iteration k

$\bar{x}^k$: samples' probability weighted solution vector at iteration k

$\bar{\bar{x}}^k$: samples' balanced solution vector at iteration k

x_{best} : best incumbent solution

$\hat{v}_{best}$: objective function value of the best incumbent solution with respect to N'

$\hat{v}_{best}^k$: objective function value of the best solution at iteration k with respect to N'

ω_m^k : dual variable vector for sample m at iteration k

ρ^k : penalty factor at iteration k

β : update parameter for the penalty factor

α^k : weight for the best incumbent solution at iteration k

Δ_α : update factor for the weight of the best incumbent solution

ϵ_α : Euclidean norm distance of the sample solutions from $\bar{\bar{x}}^k$ at iteration $k-1$

ϵ : convergence threshold for solution spread

x^{SBPHA} : best solution found by SBPHA

v^{SBPHA} : objective function value of the best solution found by SBPHA

The pseudo-code for the sampling based progressive hedging algorithm is as follows:

Sampling Based Progressive Hedging Algorithm (SBPHA) for Two-Stage SP Problems:

SBPHA Algorithm

```
1:   Initialize: Generate $M$ samples, $m = 1,2,\dots,M$ each with $N_m$ scenarios, where $|\mathrm{N_m}| = \mathrm{N}$.
2:   Generate a reference sample with $\mathrm{N}'$ scenarios, where $|\mathrm{N}'| \gg \mathrm{N}$.
3:   $\mathrm{k} \leftarrow 0$, $\omega_{\mathrm{m}}^{\mathrm{k}=0} \leftarrow 0$ for $\forall \mathrm{m} = 1,\dots,\mathrm{M}$, $\alpha^{\mathrm{k}=0} \leftarrow 1$, and require $\rho^{\mathrm{k}=0} \geq 0$,
4:   $\mathrm{P_m} \leftarrow \prod_{\mathrm{s}\in\mathrm{N_m}} \mathrm{p_s}$, $\widehat{\mathrm{P}}_{\mathrm{m}} \leftarrow \frac{\mathrm{P_m}}{\sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{P_m}}$, $\widehat{\mathrm{P}} = \{\widehat{\mathrm{P}}_{\mathrm{m}}\}_{\forall\mathrm{m}}$.
5:   Execute steps 2-4 of SAA Algorithm.
6:   $\mathrm{x_{best}} \leftarrow \mathrm{x}^{\mathrm{SAA}}$,
7:   for $\mathrm{m} = 1 \rightarrow \mathrm{M}$, do
8:       $\mathrm{x}^{m,k=0} \leftarrow \mathrm{x}^{\mathrm{m}}$
9:   end for
10:  While $(\epsilon_k \geq \varepsilon \text{ or } \bar{\bar{x}}^k \neq x_{best}), \text{and} (k < k_{max})$ do
11:          $\mathrm{k} \leftarrow \mathrm{k} + 1$,
12:          $\bar{\mathrm{x}}^{\mathrm{k}} \leftarrow \widehat{\mathrm{P}}\mathrm{x}^{\mathrm{m,k-1}}$,
13:          $\bar{\bar{\mathrm{x}}}^{\mathrm{k}} \leftarrow \alpha^{\mathrm{k}}\bar{\mathrm{x}}^{\mathrm{k}} + (1 - \alpha^{\mathrm{k}})\mathrm{x_{best}}$,
14:          if $\alpha^{\mathrm{k}-1} = 0, \alpha^{\mathrm{k}} \leftarrow \alpha^{\mathrm{k}-1}$ else $\alpha^{\mathrm{k}} \leftarrow \alpha^{\mathrm{k}-1} - \Delta_\alpha$.
15:          if $\mathrm{k} \geq 2, \rho^{\mathrm{k}} \leftarrow \begin{cases} \beta\rho^{\mathrm{k}-1} & \text{if } \epsilon_{\mathrm{k}} > \epsilon_{\mathrm{k}-1}/2 \\ \rho^{\mathrm{k}-1} & \text{otherwise} \end{cases}$, else $\rho^{\mathrm{k}} \leftarrow \rho^{\mathrm{k}}$.
```

```
16:         $\omega_m^k \leftarrow \omega_m^{k-1} + \rho^k\left(x^{m,k-1} - \bar{\bar{x}}^k\right)$
17:         for $m = 1 \rightarrow M$, do
18:             $\left[v^{m,k}, x^{m,k}\right] \leftarrow \text{Min}\left\{c^T x^{m,k} + \frac{1}{|N_m|}\sum_{n=1}^{N_m} Q(x,\xi_n) + \omega_m^k x^{m,k} + \frac{\rho^k}{2}\left\|x^{m,k} - \bar{\bar{x}}^k\right\|^2\right\}$   (2.17)
19:         end for
20:         $\epsilon_k := \left(\sum_{m=1}^{M}\|x^{m,k} - \bar{\bar{x}}^k\|\right)^{\frac{1}{2}}$
21:         for $m = 1 \rightarrow M$, do
22:             $[\hat{v}^{m,k}] \leftarrow \text{Min}\left\{c^T x^{m,k} + \frac{1}{|N'|}\sum_{s=1}^{N'} Q(x,\xi_s)\right\}$   (2.18)
23:         end for
24:         $\hat{v}_{best} \leftarrow \begin{cases} \hat{v}_{best}^k & if\ \hat{v}_{best}^k < \hat{v}_{best} \\ \hat{v}_{best} & otherwise \end{cases}$
25:         $x_{best} \leftarrow \begin{cases} x^{m',k} | m' = argmin_{m=1,\ldots,M}\ \hat{v}^{m,k} & if\ \hat{v}_{best}^k < \hat{v}_{best} \\ x_{best} & otherwise \end{cases}$
26:     end while
27:     $x^{SBPHA} \leftarrow x_{best}, v^{SBPHA} \leftarrow \hat{v}_{best}$.
```

The initialization step of the SBPHA is similar to the SAA and the only additional calculation is the sample m's probability and normalized probabilities, e.g., P_m and $\widehat{P}_m$. The probability $\widehat{P}_m$ is used to calculate the samples probability weighted solution $\bar{x}^k$ at iteration k (Step 4). The first step in SBPHA is to execute the standard SAA procedure (Step 5). Next, we calculate the samples' weighted average solution and the balanced solution. The samples' balanced solution ($\bar{\bar{x}}^k$) is a weighted average of the average solution ($\bar{x}^k$) and the incumbent best solution (x_{best}) as calculated in Step 6. The weight factor $\alpha^k \in [0,1]$ determines the bias of the best incumbent solution; whereas high values tend the sample solutions to the sample average solution, low values tend to the incumbent best solution. There are two alternative implementations; α^k can be static by setting $\Delta_\alpha = 0$ or dynamically changing over the iterations by setting $\Delta_\alpha > 0$ (see Step 14). The advantage of dynamic α^k is that, beginning with a large α^k, we first prioritize the sample average solution until the incumbent best solution quality improves. This approach allows guiding the sample solutions to a consensus sample average initially and then directing the consensus sample average in the direction of evolving best solution.

In Step 15, we update the penalty factor ρ^k depending whether the distance (ϵ_k) of sample solutions from the most recent balanced solution has sufficiently improved. We choose the improvement threshold as half of the distance in the previous iteration (e.g., ϵ_{k-1}) . Similarly, in Step 16, we update the dual variable (ω_m^k) for the linear deviation of every sample's solution from the balanced solution at iteration k. Note that the ω_m^k are the Lagrange multipliers corresponding to the equivalence of each sample's solution to the balanced solution.

In Step 18, we solve each sample problem with additional objective function terms representing the dual variables and calculate the deviation of the sample solutions from the balanced solution (i.e., ϵ_k). Step 22 estimates the objective function value of each sample solution in the original problem using the reference set N'. Steps 24 and 25 identify the sample solution $x^{m,k}$ with the best $\hat{v}^{m,k}$ in iteration k and updates the incumbent best $\hat{v}_{best}$ if there is improvement. The Steps 22 and 24-25 correspond to the integration of SAA method's selection of the best performing sample solution. Rather than terminating, the proposed SBPHA retains this information in the next iteration through the balanced solution. Step 10 checks whether the stopping conditions are met. If the iteration limit is reached $\mathrm{k} \geq \mathrm{k}_{\max}$ or when the all sample solutions converged to the balanced solution within a tolerance then the SBPHA terminates with the best found solution. The worst-case solution of the SBPHA is equivalent to the SAA solution with the same set of samples. This can be observed by noting that the best incumbent solution is initialized with the SAA's solution. Hence, the SBPHA ensures that there is always a feasible solution which has same performance or better than that of SAA's.

2.3.2. Discarding-SBPHA (d-SBPHA) Algorithm for Binary First Stage SP Problems

The Discarding-SBPHA (*d*-SBPHA) is an enhanced version of the SBPHA and aims at improving the solution. The main idea of the *d*-SBPHA is to re-solve SBPHA, by adding a constraint(s) to optimization problem (17), at the beginning of the iteration where the SBPHA finds the solution that it converges at the end. For instance, if SBPHA finds a solution at iteration 5 and converges to that solution at iteration 10, then *d*-SBPHA starts re-solving the sample problems by starting from iteration 5 with the values $(\omega, \alpha, \rho \ldots)$ at iteration 5 , in order to follow a different path from that SBPHA followed.

This modification of SBPHA can be considered as *globalization of the SBPHA* in that by repeating the discarding steps, the d-SBPHA can find the optimum solution, albeit the number of discarding steps could be infinite.

<u>Additional Notation For d-SBPHA:</u>

o_{new}: Iteration number where the SBPHA or *d*-SBPHA finds the solution, where it is found the very first, which is converged at the end.

d, d_{max}: discarding index and maximum number of discards

D: set of discarded feasible solutions, ($D_{t=1,\ldots,D}$)

nD_t: number of binary decision variables that are equal to 1 in discarded solution D_t

D_t^1: set of decision variables, that are =1 in discarded solution t,

D_t^0: set of decision variables, that are =0 in discarded solution t,

Discarding-SBPHA Algorithm for Binary First Stage SP Problems:

d-SBPHA Algorithm

```
1:  Initialize: execute steps 1-27 of SBPHA
2:  $x_{best} \leftarrow x^{SBPHA}$, $D \leftarrow \phi$, $o := 0$, $d \leftarrow 0$.
3:  While $d \le d_{max}$ do
4:      $d \leftarrow d + 1$,
5:      $o \leftarrow o_{new}$,
6:      for $m = 1 \rightarrow M$, do
7:          $x^{m,k} \leftarrow x^{m,o}$
8:      end for
9:      $\rho^k \leftarrow \rho^o$.
10:     $\alpha^k \leftarrow \alpha^o$.
11:     for $m = 1 \rightarrow M$, do
12:         $\omega_m^k \leftarrow \omega_m^o$
```

```
13:         end for
14:         $D \leftarrow D \cup \mathrm{x}_{\mathrm{best}}$
15:         execute steps 10-16 of SBPHA.
16:         for $\mathrm{m} = 1 \rightarrow \mathrm{M}$, do
17:             $[v^{m,k}, x^{m,k}] \leftarrow \mathrm{Min}\left\{c^T x^{m,k} + \frac{1}{|N_m|}\sum_{n=1}^{N_m} \mathcal{Q}(x, \xi_n) + \omega_{\mathrm{m}}^{\mathrm{k}} x^{m,k} + \frac{\rho^{\mathrm{k}}}{2}\left\|x^{m,k} - \bar{\bar{\mathrm{x}}}^{\mathrm{k}}\right\|^2\right\}$    (2.19)
                $s.t. \sum_{x_i \in D_t^1} x_i - \sum_{x_j \in D_t^0} x_j \leq nD_t - 1, \forall t = 1, \dots, D$
18:         end for
19:         execute steps 20-26 of SBPHA.
20:         $x_d^{d-SBPHA} \leftarrow x_{best}, v_d^{d-SBPHA} \leftarrow \hat{v}_{best}$ .
21:     end while
22:     $v_{best}^{d-SBPHA} \leftarrow min_{d=1,\dots,D}\ v_d^{d-SBPHA}$
23:     $x_{best}^{d-SBPHA} \leftarrow x_d^{d-SBPHA} = argmin_{d=1,\dots,D}\ v_d^{d-SBPHA}$
```

Initialization step of the *d*-SBPHA is the implementation of the original SBPHA and the only difference is to set up of the starting values of parameters for *d*-SBPHA. In step1, parameters of the algorithm are set up as the values of the algorithm when current best solution is found. Also in step 1, the set of the solutions that will be discarded is updated to prevent the algorithm to re-converge to the same solution. In step 2 to 13, algorithm updates the parameters according to starting point of the *d*-SBPHA. Step 14 updates the set of discarded solution and step 15 executes SBPHA steps to solve the sample problems. Please note that step 17 has the same objective function as in step 18 of SBPHA with extra constraint(s). These constraints prevent the algorithm to find the solutions that are already found (first stage solutions that are discarded). Further, the discarding constraints prevent *d*-SBPHA to find the best solution(s) that already found. Step 19 executes steps from 20 to 26 of the SBPHA, which test solutions' quality and performs the updating of the best solution. The only difference, in step 21, of d-SBPHA then SBPHA is to check whether the maximum number of discards is reached or not. If it is reached, then the algorithm reports the solution with the best performance (in step 22 and 23) else continues discarding. Discarding strategy provides better or (at least) the same solution as SBPHA provides.

Lower Bounds for SBPHA and d-SBPHA:

The computational effort of SBPHA is spent in solving many sample problems as well as in evaluating the first stage decisions in the larger reference set. As a computational time improvement technique for SBPHA and d-SBPHA, a lower bound can be provided for each sample that is solved by the optimization problem at step 18 in SBPHA and 17 in *d*-SBPHA. Providing a consistent/tight lower bound improves solution time. The theoretical justification of the use of a lower bound for sample problems is that if the balanced solution does not change, then the solution of the sample problems is non-decreasing. Hence, one can use the previously found solution as the lower bound (due to the Lagrangean duality property). However, if the balanced solution changes, then the lower bound property of the previous solution is not guaranteed hence a conservative estimate for lower bound is needed.

Let $lb^{m,k}$ be the lower bound for sample m at iteration k in SBPHA or *d*-SBPHA. At step 18 in SBPHA (step 17 in *d*-SBPHA) another constraint should be added: $v^{m,k} \geq \ lb^{m,k}$, $\forall m, m = 1, \dots M$, where $lb^{m,k} = c_{lb} x lb^{m,k-1}$, and $0 \leq c_{lb} \leq 1$. However c_{lb}, should not be close 1 because it might cause infeasible solutions. There is a trade of on value of c_{lb}. Higher values might cause either infeasible or sub-optimal solutions, lower values does not provide consistent/tight constraints that should help improving solution time. In this study, we tested multiple values for c_{lb}, we suggest applicants to choose $0.4 \leq c_{lb} \leq 0.6$. Providing lower bound to optimization problem saved 10%-15% of the solution time.

2.3.3. Properties of SBPHA and d-SBPHA

Proposition 1 (Equivalence): SBPHA is equivalent to SAA if algorithm is executed in one iteration. Further, SBPHA is equivalent to PHA if the samples are mutually exclusive and their union is the entire scenario set.

Proof: We prove this in two parts.

SAA: *If SBPHA terminated at step 1, then* $x^{SBPHA} = x^{SAA}$, *and* $v^{SBPHA} = v^{SAA}$. *It can be conluded that SBPHA is equivalent to SAA.*

PHA: *Under specified assumptions and for* $M = S$ *and* $N_{m=1,\dots,M} = 1$, *SBPHA=PHA.*

Let's consider a two-stage SP problem with finite number of scenarios ξ_s, $s = 1, \dots, S$, and each scenario occurs with a probability p_s, where $\sum_{s=1}^{S} p_s = 1$. Consider the SBPHA with samples as the individual scenarios, e.g., $M = S$ and $N_{m=1,\dots,M,} = 1$, where $m \neq m'$. It can be concluded that $\mathrm{P_m} = \mathrm{p_s}$. If weight for the best incumbent solution and update factor for the weight of the best incumbent solution are equal to 1 and 0, consecutively, at every iteration ($\alpha^k \coloneqq 1, \Delta_\alpha = 0$), then $\bar{\bar{\mathrm{x}}}^{\mathrm{k}} \colon = \bar{\mathrm{x}}^{\mathrm{k}} \coloneqq \sum_{s \in S} p_s x_s^k$, and $x^{SBPHA} = x^{PHA}$ and $v^{SBPHA} = v^{PHA}$. □

Proposition 2(Convergence) ***SBPHA converges to the best solution found at any iteration.***

Proof: Let assume that SBPHA finds, at iteration k, the best solution as $\mathrm{x_{best}} = \mathrm{x}^*$. Let us assume that SBPHA algorithm converges to a solution $\mathrm{x}' \neq \mathrm{x_{best}}$ and has worse objective value than x^* (with respect to the reference scenario set). Note that convergence implies $\bar{\bar{\mathrm{x}}}^{\mathrm{k}} = \bar{\bar{\mathrm{x}}}^{\mathrm{k}-1} = \mathrm{x}'$ (assuming $k_{max} = \infty$). Further in the last update, we must have $\bar{\bar{\mathrm{x}}}^{\mathrm{k}} = \mathrm{x}' = \alpha^{\mathrm{k}}\mathrm{x}' + \left(1 - \alpha^{\mathrm{k}}\right)\mathrm{x_{best}}$. Since $\alpha^{\mathrm{k}} < 1$, this equality is satisfied if and only if $\mathrm{x}' = \mathrm{x_{best}}$ which is contradiction. □

Proposition 3: SBPHA and d-SBPHA algorithms have the same convergence properties as SAA with respect to the sample size.

Proof: It is showed in (Ahmed and Shapiro 2002) and (Ruszczynski and Shapiro 2003) that SAA converges with probability one (w.p.1) to optimal solution of the original problem as sample size, $(N \to \infty)$, increases to infinity. Since step 1 in SBPHA is the implementation of SAA and that SBPHA does converge to the best solution found (Proposition 2), we can simply argue that SBPHA and *d*-SBPHA converges to optimal solution of the original problem as SAA does with increasing sample size. □

Since SBPHA and *d*-SBPHA guarantee a better or same solution quality as SAA provides, we can conjecture that SBPHA and *d*-SBPHA have more chance to reach the optimality than SAA with a given of samples and sample size.

Proposition 4: The *d*-SBPHA algorithm converges to the optimum solution as $d \rightarrow \infty$.

Proof:

Given that the *d*-SBPHA does not allow finding the same solution, in the worst case, the *d*-SBPHA iterates as many times as the number of feasible solutions (infinite in the continuous and finite in the discrete case) for the first stage decisions before it finds the optimum solution. $\square$

Clearly, as the number of discarding constraints added increases linearly with the number discarding iterations, the resulting problems become more difficult to solve. However, according to our tests for the *d*-SBPHA algorithm we experienced that *d*-SBPHA finds the optimal solution in less than 10 discarding iterations.

2.4. Proposed Algorithm for Multi-Stage Stochastic Programming Problem

In this subsection, we extend the SBPHA to multi-stage SPs. Discretization in the form of scenario tree is the standard method to solve multistage SP problems (Shapiro 2008). Let us consider a multi-stage problem with T, $(t = 1,2,\dots,T)$ time periods. At time period $t = 1$, we have only one root node which is associated with random event ξ_1 which is assumed to be known prior to the first stage decisions. At time period $t = 2$, the number of nodes is equal to the number of different random events of ξ_2 that are considered. Each node i at $t = 2$ is associated with a random event $\xi_2^i$of ξ_2 and is connected to the root node. We generate as many nodes as the number of random event ξ_3 that follow ξ_2^i for each node i. All nodes at $t = 3$ that follow ξ_2^i are connected to the node that is associated with ξ_2^i, etc. Each node at time period t is connected to an irreplaceable node at the previous time period $t-1$, called "*ancestor*" node and is connected to an irreplaceable node at time period $t+1$, called "*children*" node (Shapiro 2008).

Each child node ξ_{t+1}^{ij} that is connected to the node associated with ξ_t^i is associated with a probability $p_{ij} > 0$, so that $\sum_j p_{ij} = 1$. The probability of each scenario (a path starts from the root node and ends at the last period, T) is calculated by the product of the probabilities p_{ij} associated to the nodes on this path. After the scenario tree for multi-stage SP problem is constructed the deterministic formulation of the problem can be written as a one large optimization problem as in (2.6)-(2.9).

We note that even the number of scenarios is comparable in a two-stage and a multi-stage SP, solving the latter is more difficult than solving the former (Shapiro 2008). In both cases, it is impossible to solve if the number of the scenarios is very big. One of the powerful methods to apply in such cases is the SAA. The sampled scenario tree can be created as follows: a randomly selected sample $\xi_2^i, i = 1,2,\dots,N_1$, of N_1 realizations (nodes) are generated. The probability of each node is= $1/N_1$. With the same logic, a randomly selected sample $\xi_3^{ij}, j = 1,2,\dots,N_2$, is generated for each node ξ_2^i, etc. At the end, a scenario tree with $N = \prod_{t=1}^{T-1} N_t$ scenarios will be generated. The occurance probability for all scenarios in this scenario tree, will be equal $(= \frac{1}{N})$. This process is called "*conditional sampling*" in Shapiro (2002); Shapiro (2003); Shapiro (2008).

Once the scenario tree generated, one can create a deterministic equivalent of the problem and solve with any appropriate algorithm. Under mild regularity conditions, it is shown that the optimal first stage solution and the optimal objective function value of the SAA problem converges to the original of the multi-stage problem's first stage solution and objective function value w.p.1 as the sample sizes $N_t, t = 1,2, \dots, T-1$, increases to infinity (Olsen 1976; Shapiro 2003; Pennanen 2005; Shapiro 2008). However the number of scenarios in multi-stage SP problems increases exponentially with increasing number of time stages. Therefore the deterministic equivalent of the created scenario tree becomes too large to solve as the number of time stages increases. Hence that multi-stage SP problems are way more difficult than two-stage SP problems.

The main difficulty in extending the SBPHA to multi-stage SP problems is to find an upper bound with fixing first stage solution and relaxing non-anticipativity constraints in $t > 1$ time periods.

To illustrate, let us consider a three stage SP problem with 9 scenarios as in

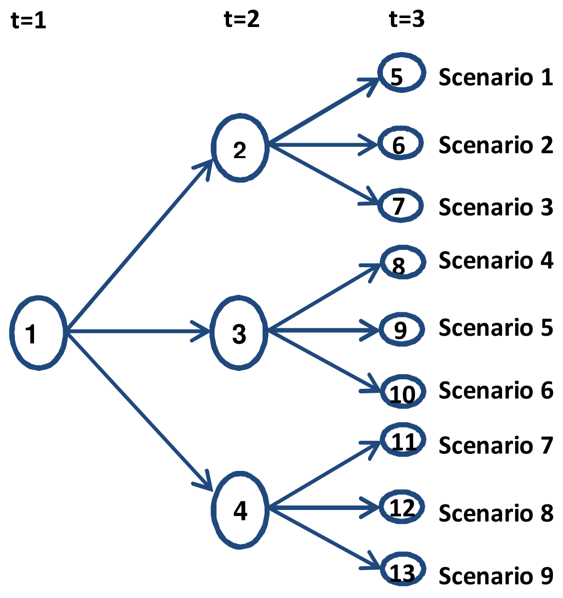


Figure 4.

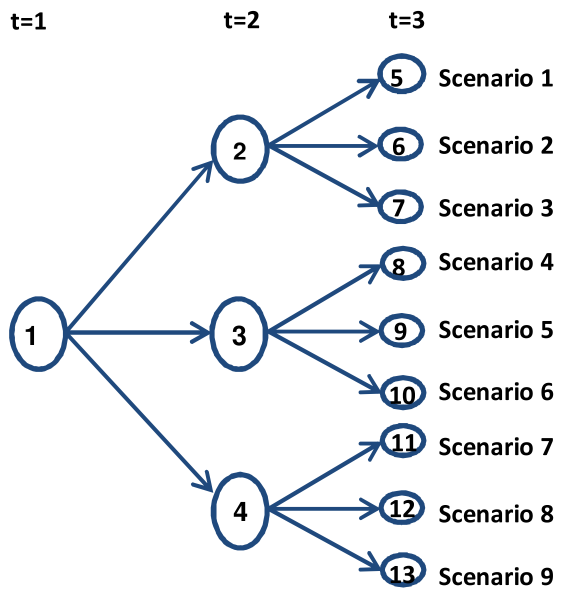


Figure 4: A scenario tree of three-stage SP problem with nine scenarios

Mathematical formulation of the three-stage SP problem in

Figure 4 can be written as two-stage SP problem in the following form. Note that the notation and formulations are from Shapiro (2002):

$$\min_{x_1} c_1 x_1 + \mathbb{E}[Q_2(x_1, \xi_2)] \text{ subject to } A_{11}x_1 = b_1, x_1 \geq 0. \tag{2.20}$$

Note that, $Q_2(x_1, \xi_2)$ itself is a SP problem, and can be estimated by sampling. For a given x_1 and ξ_2, the corresponding expected objective function value can be estimated by SAA. Let $\hat{Q}_2(x_1, \xi_2)$ be the estimator of $Q_2(x_1, \xi_2)$, then

$$Q_2(x_1, \xi_2) \geq \mathbb{E}[\hat{Q}_2(x_1, \boldsymbol{\xi_2}) | \boldsymbol{\xi_2} = \xi_2] \tag{2.21}$$

For every feasible x_1 and ξ_2. $Q_2(x_1, \xi_2)$ can be written as;

$$\min_{x_2} c_2 x_2 + \mathbb{E}[Q_3(x_2, \boldsymbol{\xi_3}) | \boldsymbol{\xi_2}] \text{ subject to } A_{21}x_1 + A_{22}x_2 = b_2, x_2 \geq 0. \tag{2.22}$$

where $Q_3(x_2, \xi_3)$ is optimal of the problem;

$$\min_{x_3} c_3 x_3 \text{ subject to } A_{32}x_2 + A_{33}x_3 = b_3, x_3 \geq 0. \tag{2.23}$$

If we relax the non-anticipativity constraints at the second stage of the problem in (2.20), then two-stage SP problem can be written,

$$\min_{x_1 \in X_1} c_1 x_1 + \mathbb{E}[Q(x_1, \xi_2, \xi_3)], \tag{2.24}$$

where $X_1 := \{x_1 \in \mathbb{R}^{n_1} : A_{11} x_1 = b_1, x_1 \geq 0\}$,

and $Q(x_1, \xi_2, \xi_3)$ represent the optimal value of the problem;

$$\begin{aligned} &\min_{x_1, x_3} c_2 x_2 + c_3 x_3 \\ &\quad \text{subject to} \\ &\qquad A_{21} x_1 + A_{22} x_2 = b_2, \\ &\qquad A_{32} x_2 + A_{33} x_3 = b_3, \\ &\qquad x_2 \geq 0, x_3 \geq 0. \end{aligned} \tag{2.25}$$

The relaxed (non-anticipativity constraints) problem in (2.24) and (2.25) above gives a lower objective function value than the problem in (2.20) – (2.23) gives.

Assume that $\mathbb{E}[Q(x_1, \xi_2, \xi_3)]$ is finite. Then by the Law of Large Numbers (LLN), the expected value of the right hand side is equal to the left hand side of the (2.26).

$$c_1 x_1 + \frac{1}{N} \sum_{i=1}^{N} Q\left(x_1, \xi_2^i, \xi_3^i\right) \to c_1 x_1 + \mathbb{E}[Q(x_1, \xi_2, \xi_3)] \ \text{ w.p. 1 as } N \to \infty, \tag{2.26}$$

Subsequently, we can argue that any (feasible) first stage solution of the problem in the left hand side of (2.26) provides a valid upper statistical bound for the problem given in (2.24) and (2.25). Since the problem given in (2.24) and (2.25) gives smaller objective function values than the original problem-three stage SP problem in (2.20-2.23), we cannot guarantee that the left hand side of the (2.26) gives valid statistical upper bound for the original three-stage SP problem in (2.20-2.23)(Shapiro 2002;

Shapiro 2003). As we mentioned above, finding an upper bound for multi-stage SP problem. In contrast, the upper bound can be calculated in two-stage problems through a simple evaluation of a first stage solution in the reference set.

Further, evaluating the performance of sample solution is not as easy in a multi-stage SP as in a two-stage SP problem. The reason is that a sample's solution might not have a decision at a particular node of the scenario tree. Only the first stage decisions are guaranteed to be available from any sample solution. To overcome this evaluation difficulty, we evaluate only the first stage decisions by solving another SP for the time periods $t > 1$.

In order to adapt the SBPHA to the multi-stage SP, one modification is the non-anticipativity constraints in $t > 1$ time stages. Both two-stage and multi-stage SP problems have non-anticipativity constraints at the first stage. In addition to the first stage, multi-stage SPs have scenario based non-anticipativity in the latter stages. We show this through anillustrative example.

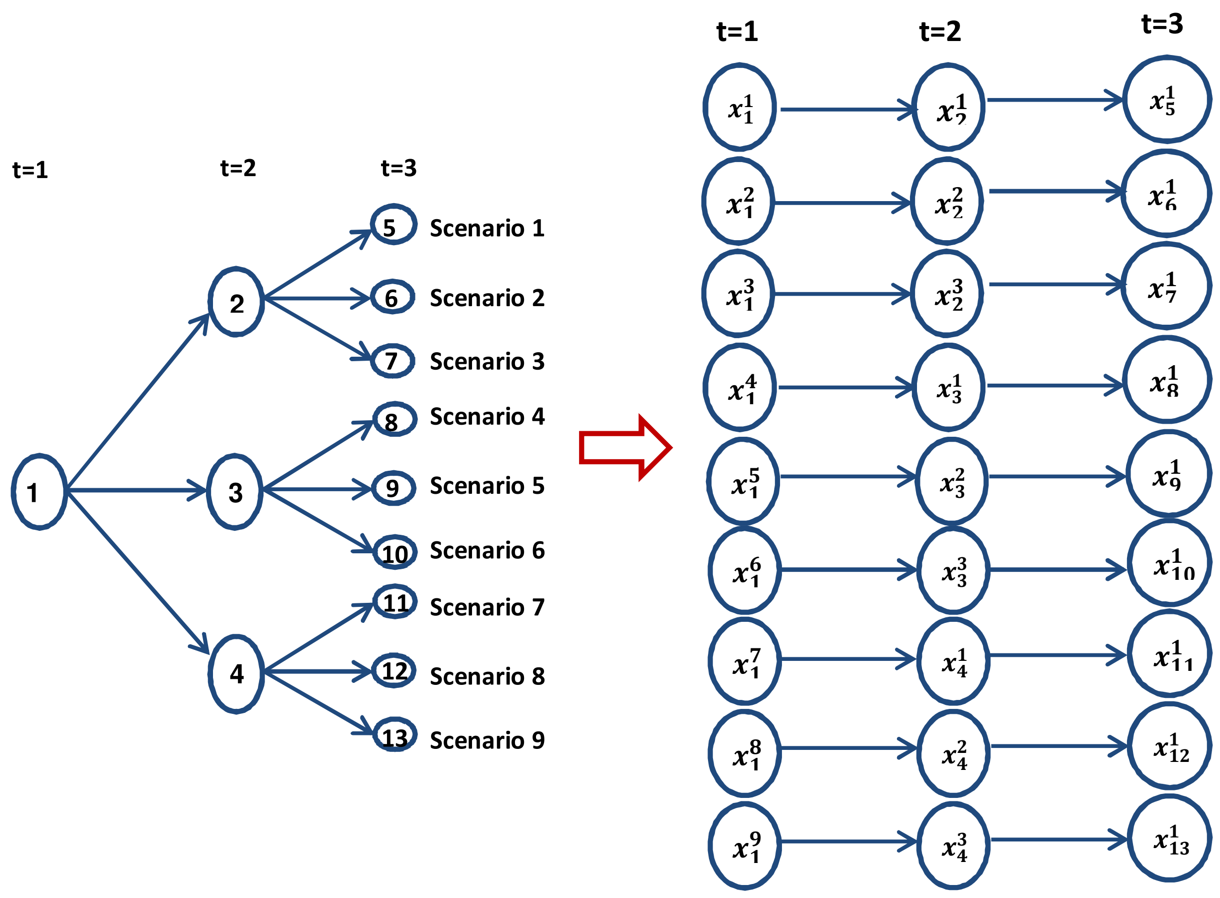


Figure 5: Scenario based decomposition of the problem in

Figure 4

Let us consider the problem in

Figure 4. After decomposing the problem into scenarios, the decomposed tree is shown in Figure 5. Let us sample three samples $(M = 3, m = 1,2,3)$ each with two scenarios $(N = 2, n = 1,2)$ from the scenario tree in Figure 5.

Table 3: Sample s and scenarios for Illustrative Example

Sample m	**Scenarios**		**First stage Decision Variables**	**Second Stage Decision Variables**	
	n=1	***n=2***	***n=1,2***	***n=1***	***n=2***

m=1	1	4	$x_1^{1,4}(m=1)$	$x_2^1(m=1)$	$x_2^4(m=1)$
m=2	5	9	$x_1^{5,9}(m=2)$	$x_2^5(m=2)$	$x_2^9(m=2)$
m=3	1	8	$x_1^{1,8}(m=3)$	$x_2^1(m=3)$	$x_2^8(m=3)$

Clearly, the first stage decisions must satisfy non-anticipativity constraints.

$$x_1^{1,4}(1) = x_1^{5,9}(2) = x_1^{1,8}(3) \tag{2.27}$$

Additional non-anticipativity constraints are as follows:

$$x_2^1(1) = x_2^1(3) \tag{2.28}$$

$$x_2^4(1) = x_2^5(2) \tag{2.29}$$

$$x_2^9(2) = x_2^8(3) \tag{2.30}$$

Equation (2.28) shows that the second stage decisions have to be the same for scenario 1, which is selected by sample 1 and 3. Equation (2.29) shows that the second stage decision variables of scenario 4 in sample 1 and scenario 5 in sample 2 have to be the same since both of them stem from node 3, which means they have the same history until time period 2. Equation (2.30) shows that the second stage decision variables of scenario 9 in sample 2 and scenario 8 in sample 3 have to be the same since both of them stem from node 4. Note that non-anticipativity constraints are not applied to the last time period (T) decision variables.

The SBPHA algorithm is given in section 2.5, it checks the convergence of the equations given above in step 20, updates penalty factor in step 15 and penalizes the gap between the variables and the average values in step 18. The average values are calculated in step 13. The rest of the SBPHA algorithm works the same for multi-stage SP problems as it works for two-stage SP problems.

CHAPTER III: APPLICATIONS of SBPHA and *d*-SBPHA

This section presents the results of an experimental study performed to investigate the computational and solution quality performance of the proposed SBPHA and *d*-SBPHA for solving two-stage and multi-stage SP problems. For two-stage SP problem, we selected the Capacitated Reliable Facility Location Problem (CRFLP) and benchmark the results of SBPHA and *d*-SBPHA with those of SAA. Secondly, we applied the SBPHA algorithm to the multi-stage stochastic lot–sizing problem.

In what follows, we first present the results of the two-stage stochastic programs (Section 3.1). After introducing the CRFLP and relevant past work, we describe the experimental setting in Section 3.1.1., present results for algorithm tuning in Section 3.1.2, and discuss the comparative results of SBPHA and *d*-SBPHA vis-à-vis those of SAA. We then present the results of implementing SBPHA algorithm to the multi-stage stochastic lot–sizing problem on an illustrative example. All the code development and programming is performed in Matlab R2010b and the integer programs are solved with CPLEX 12.1. The experiments are conducted on a PC with Intel(R) Core 2 CPU, 2.13 GHz processor and 2.0 GB RAM running on Windows 7 OS.

3.1. Capacitated Reliable Facility Location Problem (CRFLP)

Facility locations are primary strategic supply chain decisions and require significant investments spanning over long planning horizons, e.g., ranging from 3 to 10 years depending on the industry. Given the duration of the planning horizon and the level and scope of uncertainty in today's business environment, the supply chain designers are compelled to anticipate and plan for uncertain future events in their network design decisions. A notable category of these supply chain uncertainties is the disruption of facilities which affect the supply chain's ability to efficiently satisfy the customer demand (Schütz, Tomasgard et al. 2009). These disruptions can be either natural disasters or man-made (such as terrorist attacks, labor strikes, etc.). In certain cases, the disruption at a region may extend or migrate through the network and affect other parts of the supply chain network (Masihtehrani 2011). Recent examples of such disruptions are the 2011 earthquake in Japan affecting Toyota's ability to ship parts and finished vehicles (Brennan 2011; TheGuardian 2011), hurricanes Katrina and Rita in 2005 disrupting the nation's oil refineries, and the 2000 fire at the Royal Philips Electronics radio frequency chip manufacturing plant in Albuquerque halting the production of Ericsson and Nokia (Snyder, Scaparra et al. 2006).

Following a disruption event, there is hardly any recourse action to change the supply chain substructure rapidly (Snyder, Scaparra et al. 2006). Instead, a common recourse is to reassign customers to other facilities or arrange alternative sources of supply. In either case, the cost of serving the customer demand increases e.g., due to higher transportation cost. Over the past decade, the consideration of such disruptions affecting the supply chain network design has received significant attention from both the researchers and practitioners.

An exemplary earlier study can be found in (Snyder and Daskin 2005). Authors developed a reliability based formulation for Uncapacitated Facility Location

Problem (UFLP) and the p-median problem (PMP). More recently, Shen, Zhan et al. (2011) studied a variant of reliable UFLP, and proposed efficient approximation algorithms for URFLP by using the special structure of the problem. However these approximations cannot be applied to the general class of facility location problems such as Capacitated Reliable Facility Location Problems (CRFLP).

In practice, capacity decisions are considered jointly with the location decisions. Further, the capacity of facilities often cannot be changed in the event of a disruption. Following a facility failure, customers can be assigned to other facilities only if these facilities have sufficient available capacity. Thus capacitated reliable facility location problems are more complex than their uncapacitated counterparts (Shen, Zhan et al. 2011) and the studies considering capacitated reliable facility location problem are limited. Snyder and Ülker (2005) study the CRFLP and propose an algorithm based on Sample Average Approximation (SAA) embedded with Lagrangean relaxation. Gade (2007) apply the SAA method in combination with a dual decomposition method to solve CRFLP.

Peng, Snyder et al. (2011) propose a hybrid meta-heuristic based on genetic algorithm to solve a related problem where the objective is to minimize the total fixed and transportation cost while limiting the disruption risk based on the p-robustness criterion. In summary, the earlier work on CRFLP uses SAA based approximation or meta-heuristic methods to overcome the computational complexity associated with large number of scenario realizations.

We now introduce the notation used for the formulation of CRFLP. Let F_R and F_U denote the set of possible reliable and unreliable facility sites, respectively, and

$F = F_R \cup F_U \cup \{f_e\}$ denote the set of all possible facility sites, including the emergency facility (f_e). Let $\mathcal{D}$ denote the set of customers (i.e., demand points). Let f_i be the fixed cost for facility $i \in F$, which is incurred if the facility is opened, and d_j be the demand for customer $j \in \mathcal{D}$. The c_{ij} denote the cost of satisfying each unit demand of customer j from facility i and include such variable cost drivers as transportation, production, and inventory. Each unit of demand that is satisfied by the emergency facility cause a large penalty (h_j) cost. This penalty can be incurred due to finding an alternative source or due to the lost sale. Lastly, the facility i has limited capacity and can serve at most b_i units of demand.

We formulate the CRFLP as a two-stage SP problem. In the first stage, the location decisions are made prior to the realization of random failures of the located facilities. In the second stage, following the facility failures, the customer-facility assignment decisions are made for every customer given the surviving facilities. The goal is to identify the set of facilities to be opened while minimizing the total cost of open facilities and the expected cost of meeting demand of customers from the surviving facilities and the emergency facility.

In the scenario based formulation of CRFLP, let s denote a failure scenario and the set of all failure scenarios is S, where $s \in S$. Let p_s be the probability that scenario s occurs and $\sum_{s \in S} p_s = 1$. Further let k_i^s be the indicator parameter denoting whether the facility i survives, i.e., $k_i^s = 1$, and $k_i^s = 0$ otherwise. For instance, in case of independent facility failures, we have $|S| = 2^{|F_U|}$ possible failure scenarios. Note that our proposed method does not require any assumption on independence and distribution of each facility's failure.

The binary decision variable x_i specifies whether facility i is opened or not, and the variable y_{ij}^s specifies the fraction of demand of customer j satisfied by facility i in scenario s. The scenario based formulation of the CRFLP as a two-stage stochastic program is as follows.

CRFLP:

$$\text{Minimize} \qquad \sum_{i\in F} f_i\, x_i + \sum_{s\in S} p_s \sum_{j\in D}\sum_{i\in F} d_j c_{ij} y_{ij}^s \tag{3.1}$$

Subject to

$$\sum_{i\in F} y_{ij}^s = 1 \qquad \forall j \in D, s \in S \tag{3.2}$$

$$y_{ij}^s \le x_i \qquad \forall j \in D, i \in F, s \in S \tag{3.3}$$

$$\sum_{j\in D} d_j\, y_{ij}^s \le k_i^s b_i \qquad \forall i \in F, s \in S \tag{3.4}$$

$$x_i \in \{0,1\} \qquad \forall i \in F \tag{3.5}$$

$$y_{ij}^s \in [0,1] \qquad \forall j \in D, i \in F, s \in S \tag{3.6}$$

The objective function in formulation (3.1) minimizes the total fixed cost of opening facilities and the expected second stage cost of satisfying customer demand through surviving facilities and the emergency facility. Constraints (3.2) ensure that demand of each customer is fully satisfied by either open facilities or the emergency facility in every failure scenario. Constraints (3.3) ensure that, in any failure scenario, a customer's demand cannot be served from a facility that is not opened. Constraints (3.4) prevent the assignment of any customer to a facility that is failed and also ensure the total demand allocated to the facility does not exceed its capacity in any failure scenario. Constraints (3.5) are integrality conditions and constraints (3.6) are simple upper and lower bounds on the demand allocation variables.

3.1.1. Experimental Setting

We used the test data sets available from the literature Zhan (2007) which are also used in Shen, Zhan et al. (2011) for the URFLP. In these data sets, the coordinates of site locations (facility, customer) are i.i.d. and sampled from $\mathrm{U}[0,1] \times \mathrm{U}[0,1]$. The sets of customer and facility sites are identical. The customer demand is i.i.d., sampled from $\mathrm{U}[0,1000]$, and rounded to the nearest integer. The fixed cost of opening an unreliable facility is i.i.d. and sampled from $\mathrm{U}[500,1500]$, and rounded to the nearest integer. For the reliable facilities, we set the fixed cost to $2{,}000$ for all facilities. The variable costs c_{ij} for $i = 1, \dots, |F| - 1$ and $\forall j$ are chosen as the Euclidean distance between sites. We assign the large penalty cost $c_{|F|j}$ for serving customer j from the emergency facility as 20. Zhan (2007) and Shen, Zhan et al. (2011) consider URFLP and thus their data sets do not have facility capacities. In all our experiments, we selected identical capacity levels for all facilities, i.e., $b_{i=1,\dots,|F|} = 2{,}000$.

In generating the failure scenarios, we assume that the facility failures are independently and identically distributed according to the Bernoulli distribution with probability q_i, i.e., the failure probability of facility i. We experimented with two sets of failure probabilities; first set of experiments consider uniform failure rates, i.e., $q_{i \in F_U} = q$ where $q = \{0.1, 0.2, 0.3\}$, and the second set of experiments consider bounded non-uniform failure rates i.e. q_i, where $q_i \leq 0.3$. We restricted the failure probabilities with 0.3 since larger failure rates are not practicable. The reliable facilities and emergency facility are perfectly reliable, i.e., $q_{i\in(F_R \cup f_e)} = 1$. Note that the case $q_{i\in F_U} = 0$ corresponds to the deterministic fixed-charge facility location problem, and $q_{i\in F_U} = 1$ corresponds to the case where all unreliable facilities fail. The failure scenarios $s \in S$ are generated as follows. Let $F_f^s \subset F_U$ denote the facilities that are failed, and $F_{r\in F_U}^s \equiv F_U \backslash F_f^s$ be the set of surviving facilities in scenario s. The facility indicator parameter in scenario s become k_i^s=0 if $\mathrm{i} \in F_f^s$, and k_i^s=1 otherwise, e.g., if $\mathrm{i} \in F_r^s \cup F_R \cup \{f_e\}$. The probability of scenario s is then calculated as $p_s = q^{|F_f^s|}(1-q)^{|F_r^s|}$. Detailed date set can be found in the Appendix in Table 16.

In all experiments, we used $|\mathcal{D}| = F_R \cup F_U = 12 + 8 = 20$ sites which is a large-sized CRFLP problem and is more difficult to solve than the uncapacitated version (URFLP). The size of the failure scenario set is $|S| = 4{,}096$. The deterministic equivalent formulation has 20 binary x_i and $1{,}720{,}320 = |F| \times |D| \times |S| = 21 \times 20 \times 4{,}096$ continuous y_{ij}^s variables. Further, there are $1{,}888{,}256 = 81{,}920 + 1{,}720{,}320 + 86{,}016 = |D| \times |S| + |F| \times |D| \times |S| + |F| \times |S|$ constraints corresponding to (3.2-4). Hence, the size of the constraint matrix of the deterministic equivalent MIP formulation is $1{,}720{,}320 \times 1{,}888{,}256$ which cannot be tackled with exact solution procedures (e.g., branch-and-cut or column generation methods). Note that while solving LPs with this size is computationally feasible, the presence of the binary variables makes the solution a daunting task. For instance, there are $1{,}048{,}576 = 2^{20}$ possible combinations of binary variables x_i and a large scale LP must be solved for each combination. We generated sample sets for SAA and the SBPHA (and *d*-

SBPHA) by randomly sampling from $\mathrm{U}[0,1]$ as follows. Given the scenario probabilities, p_s, we calculate the scenario cumulative probability vector $\{p_1, (p_1 + p_2), \ldots, (p_1 + p_2 + \cdots + p_{|S|-1}), 1\}$ which has $|S|$ intervals. We first generate the random number and then select the scenario corresponding to the interval containing the random number. We tested the SAA, SBPHA, and *d*-SBPHA algorithms with varying number of samples (M), and sample sizes (N). Whenever possible, we use the same sample sets for all three methods. We select the reference set (N') as the entire scenario set, i.e., $N' = S$ which is used to evaluate the second stage performance of a solution. We note that this is computationally tractable due to relatively small number of scenarios and that the second stage problem is an LP. In case of large scenario set or integer second stage problem, one should select $N' \ll S$.

3.1.2. Parameter Sensitivity

In this section, we evaluate the sensitivity of the SBPHA with respect to the weight for the best incumbent solution parameter(α), penalty factor(ρ), and update parameter for the penalty factor (β). Recall that α determines the bias of the best incumbent solution in obtaining the samples' balanced solution, which is obtained as a weighted average of the best incumbent solution and the samples' probability weighted solution. The parameter ρ penalizes the Euclidean distance of a solution from the samples' balanced solution and β is the multiplicative update parameter for ρ between two iterations. In all these tests, we set $(M, N) = (5, 10)$, and $q = 0.3$ unless otherwise stated. We experimented with two α strategies, static and dynamic α. We solved in total 480$(= 10 \text{ replications } \times 48 \text{ parameter settings})$ problems.

The summary results of solving CRFLP using 10 independent sample sets (replications) with static strategy α=0.6 and dynamic strategy $\Delta\alpha = 0.03$, $\beta = \{1.1, 1.2, 1.3, 1.4, 1.5, 1.8\}$, and $\rho = \{1, 20, 40, 80, 200\}$ are shown in Table 4. The detailed results of the 10 replications of Table 4 together with the detailed replication results with static strategy for α={0.7,0.8} and dynamic strategy $\Delta\alpha = \{0.02, 0.05\}$ are presented in Table 12 in the Appendix.

The first column in Table 4 shows the α strategy and its parameter value, i.e., Δ_α for dynamic and α for static. Note that in the dynamic strategy, we select the initial value as $\alpha^{k=0} = 1$. The second and third columns show penalty factor (ρ) and update parameter for the penalty factor (β), consecutively. The objective function values for the 10 replications (each replication consists of $M = 5$ samples) are reported in columns $4, 5, \dots, 13$ (shown only for replications $1, 2$ and 10 in Table 4 and detailed results are shown in Table 12). Column 14 presents the average objective function value across 10 replications, and Column 15 presents the optimality gap (i.e., gap_1) between the average replication solution and the best objective function value found which is 8995.08[1]. Columns 16 and 17 present the minimum and maximum objective values across 10 replications. Average objective function value and gap_1 are calculated as follows:

$$v_{Rep}^{\text{Average}} = \frac{1}{Rep} \sum_{r=1}^{Rep} v_r^{SBPHA} \tag{3.7}$$

$$Gap_1 = \frac{v_{Rep}^{\text{Average}} - v^*}{v^*} \times 100\% \tag{3.8}$$

[1] The best solution is obtained by selecting the best amongst all SBPHA solutions (e.g., out of 480 solutions) and the time-restricted solution of the CPLEX. The latter solution is obtained by solving the deterministic equivalent using CPLEX method with %0.05 optimality gap tolerance and 10 hours (36,000 seconds) of time limit until either the CPU time-limit is exceeded or the CPLEX terminates due to insufficient memory.

where Rep is the number of replications, e.g., $Rep = 10$ in this section's experiments.

Table 4: Summary Objective Function Results for Solving 10 Replications of CRFLP with different parameter values.

Alpha(α)	Rho (ρ)		Replication (1,…,Rep)				Objective			
Strategy/ Parameter	Start	Update Parameter(β)	1	2	…	10	Average	Gap1 (%)	Min	Max
Static/ α=0.6	1	1.8	9,825	9,032	…	9,515	**9,358**	**4.0**	**8,995**	**10,006**
	20	1.2	9,751	9,104	…	9,483	**9,316**	**3.6**	**8,995**	**10,006**
	20	1.5	9,547	9,271	…	9,332	**9,361**	**4.1**	**9,024**	**10,006**
	40	1.3	9,547	8,995	…	9,404	**9,335**	**3.8**	**8,995**	**10,006**
	40	1.4	9,528	9,271	…	9,586	**9,337**	**3.8**	**9,024**	**10,006**
	80	1.1	9,362	8,995	…	9,112	**9,177**	**2.0**	**8,995**	**9,528**
	80	1.2	9,547	9,032	…	9,167	**9,134**	**1.5**	**8,995**	**9,547**
	200	1.1	9,362	9,287	…	9,096	**9,251**	**2.8**	**8,995**	**10,006**
Dynamic/ Δα=0.03	1	1.8	9,528	8,995	…	9,637	**9,346**	**3.9**	**8,995**	**9,713**
	20	1.2	9,362	9,024	…	9,528	**9,250**	**2.8**	**8,995**	**9,528**
	20	1.5	9,825	9,292	…	9,467	**9,360**	**4.1**	**8,995**	**9,825**
	40	1.3	9,528	9,167	…	9,112	**9,241**	**2.7**	**8,995**	**9,528**
	40	1.4	9,825	9,292	…	9,467	**9,360**	**4.1**	**8,995**	**9,825**
	80	1.1	9,825	9,024	…	9,362	**9,235**	**2.7**	**8,995**	**9,825**
	80	1.2	9,528	9,104	…	9,944	**9,283**	**3.2**	**8,995**	**9,944**
	200	1.1	9,825	9,167	…	9,112	**9,195**	**2.2**	**8,995**	**9,825**

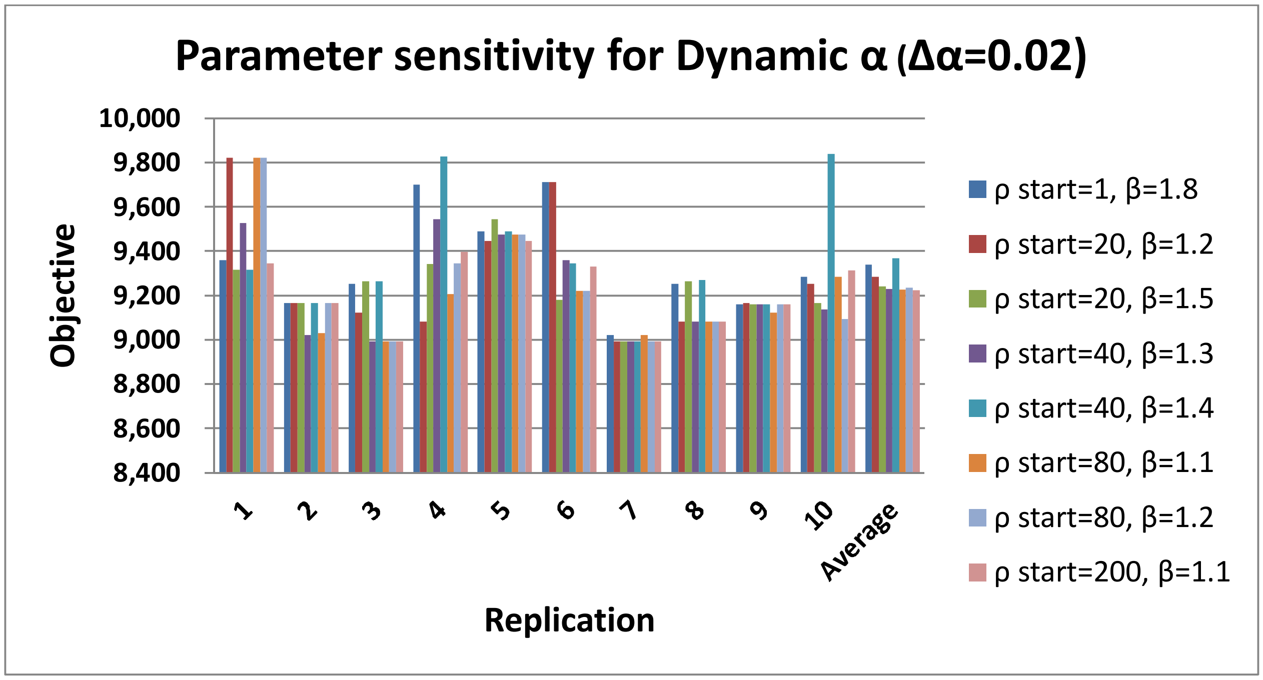


Figure 6: Effect of Dynamic Δα=0.02 and Different ρ and β Strategies on the Solution Quality for CRFLP with Facility Failure Probability $q = 0.3$.

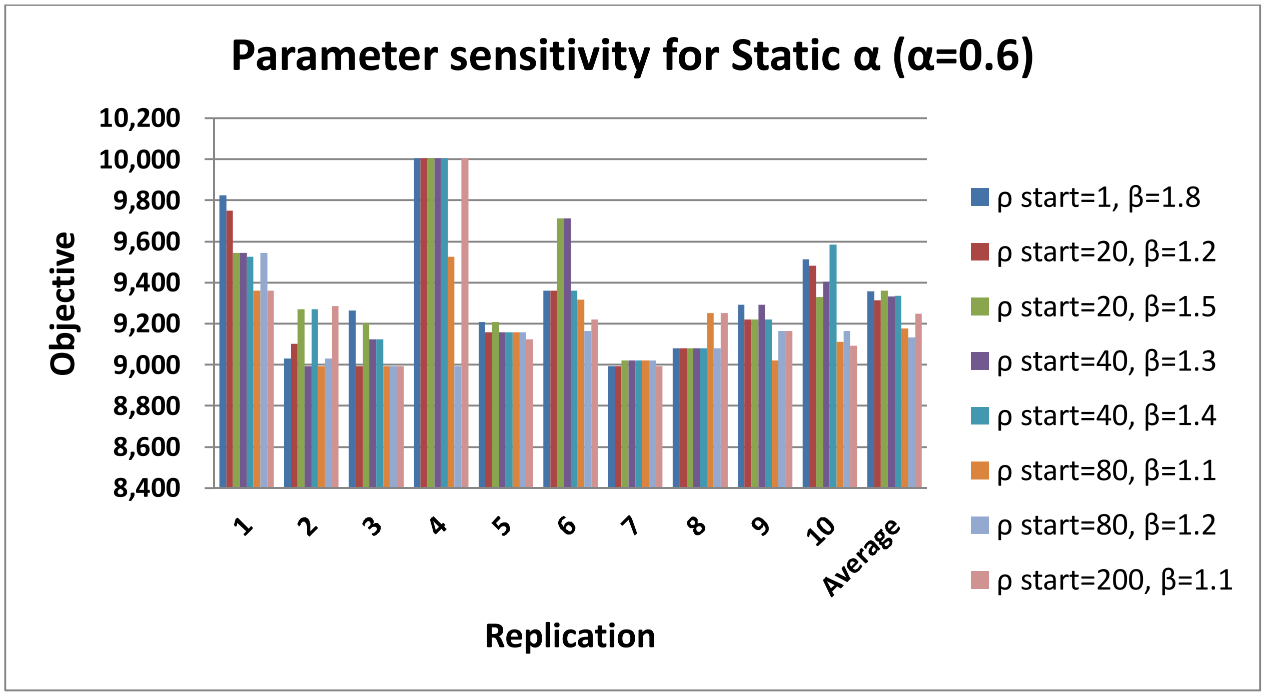


Figure 7: Effect of Static α=0.6 and Different ρ and β Strategies on the Solution Quality for CRFLP with Facility Failure Probability $q = 0.3$.

In Figure 6 and Figure 7 the effect of dynamic Δα=0.02 and static α=0.6 and different ρ and β parameter settings on the solution quality for CRFLP with failure probability $q = 0.3$ over 10 different replications are shown. The average objective function value of these 10 replications shows that the proposed algorithm (SBPHA) is relatively insensitive to the parameter settings to converge to good (optimal) solution(s) with both dynamic and static α strategies.

In Table 5, we report on the computational (CPU) time in seconds for tests presented in Table 4. The complete results are provided in Table 13 in the Appendix. First observation from Table 4 and Table 5 is that the SBPHA is relatively insensitive to the α strategy employed and the parameter settings selected. Secondly, we observe that the performance of the SBPHA with different parameter settings depends highly on the sample. As seen in Table 12 replication 7, most of the configurations show good performance as they all obtain the optimal solution. Further, as the Δ_α increases, the best incumbent solution becomes increasingly more important leading to decreased computational time. While some parameter settings exhibit good performance in solution quality, their computational times are higher, and vice versa.

Table 5: Summary CPU Times for Solving 10 Replications of CRFLP with different parameter values.

Alpha(α)	**Rho (ρ)**		**Replication (1,…,Rep)**				**Time (s)**
Strategy/ Parameter	**Start**	**Update Parameter(β)**	**1**	**2**	**…**	**10**	**Average**
Static/ α=0.6	**1**	**1.8**	399	533	…	419	**470.4**
	20	**1.2**	649	740	…	837	**786.6**
	20	**1.5**	391	462	…	410	**483.8**
	40	**1.3**	439	475	…	491	**498.5**
	40	**1.4**	379	424	…	945	**474.6**
	80	**1.1**	670	665	…	1,402	**801.1**
	80	**1.2**	450	430	…	554	**565.0**
	200	**1.1**	421	390	…	582	**442.9**
Dynamic/ Δα=0.03	**1**	**1.8**	379	468	…	421	**475.1**
	20	**1.2**	575	782	…	742	**794.0**
	20	**1.5**	319	447	…	431	**452.7**
	40	**1.3**	391	484	…	484	**541.1**
	40	**1.4**	337	447	…	466	**455.3**
	80	**1.1**	515	729	…	710	**810.9**
	80	**1.2**	388	546	…	568	**560.9**
	200	**1.1**	346	401	…	595	**545.0**

In Figure 8 the effect of static α=0.6 and different ρ and β parameter settings on the CPU time for CRFLP with failure probability $q = 0.3$ over 10 different replications are shown. The average CPU time value of these 10 replications shows that the proposed algorithm (SBPHA) is relatively sensitive to the parameter settings to converge to good (optimal) solution(s) with static α strategy. The same tests are done for other parameter settings. We observe that a parameter selection index is needed in order to test the SBPHA and *d*-SBPHA algorithms performances.

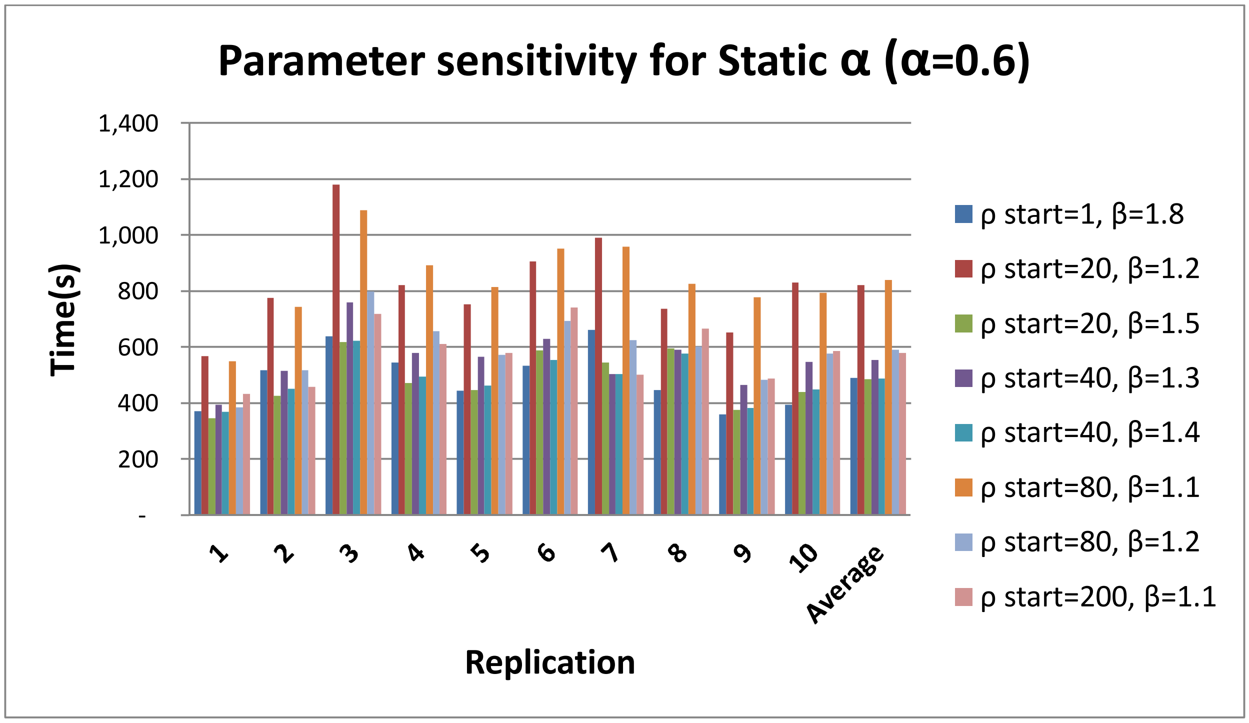
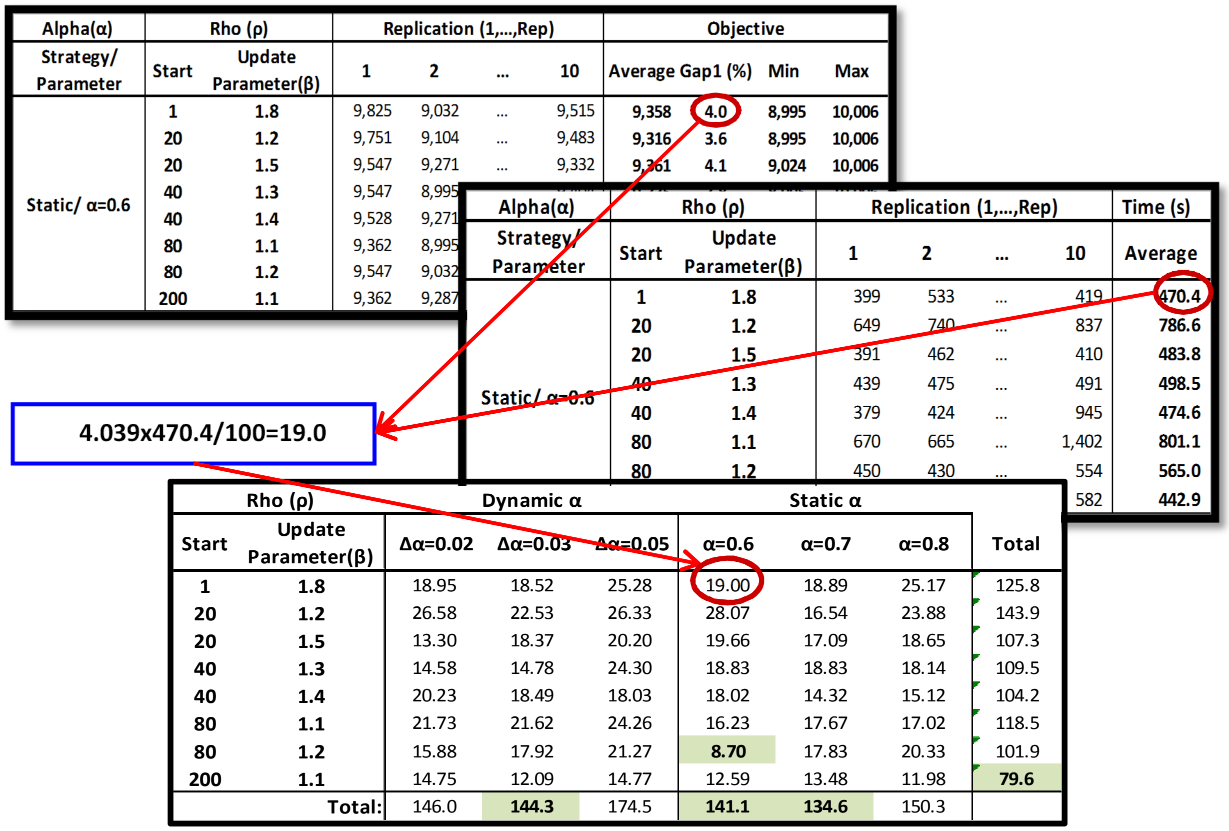


Figure 8: Effect of Static α=0.6 and Different ρ and β Strategies on the CPU Time for CRFLP with Facility Failure Probability$q = 0.3$.

In selecting the parameter settings for SBPHA, we are interested in a parameter setting that offers a balanced trade-off between the solution quality and the solution time. In order to determine such parameter settings, we developed a simple, yet effective, parameter configuration selection index. The selection criterion is the product of average gap_1 and CPU time. Clearly, smaller the index value, the better is the performance of the corresponding parameter configuration. To illustrate, using the results of the first row in Table 4 and Table 5, the index of static $\alpha = 0.6$, with starting penalty factor $(\rho) = 1$ and penalty factor update $\beta = 1.8$, is calculated as 19.00 ($= 4.039\% \times 470.4$). Parameter selection indexes corresponding to all 480 experiments are shown in Table 6 (and in Figure 9). According to the aggregate results in Table 6 (the ‘Total’ row at the bottom), the static $\alpha = 0.7$ setting is the best, the static $\alpha = 0.6$ is the second best, and dynamic α with $\Delta_{\alpha} = 0.03$ provides the third best performance. Hence, we use only these α parameter configurations in our experiments. In terms of penalty parameter configuration, we select the best setting, i.e., starting penalty factor $(\rho) = 200$ and penalty factor update $\beta = 1.1$ for all α parameter configurations. Note that by selecting a larger starting penalty factor, the SBPHA would converge faster but the quality of the solution converged would be lower. Therefore, we restricted our experiments to penalty factor starting with $(\rho) = 200$ which provides good solution quality and time trade-off.



Alpha(α)	Rho (ρ)		Replication (1,...,Rep)				Objective			
Strategy/ Parameter	Start	Update Parameter(β)	1	2	...	10	Average	Gap1 (%)	Min	Max
Static/ α=0.6	1	1.8	9,825	9,032	...	9,515	9,358	4.0	8,995	10,006
	20	1.2	9,751	9,104	...	9,483	9,316	3.6	8,995	10,006
	20	1.5	9,547	9,271	...	9,332	9,361	4.1	9,024	10,006
	40	1.3	9,547	8,995						
	40	1.4	9,528	9,271						
	80	1.1	9,362	8,995						
	80	1.2	9,547	9,032						
	200	1.1	9,362	9,287						

Alpha(α)	Rho (ρ)		Replication (1,...,Rep)				Time (s)
Strategy/ Parameter	Start	Update Parameter(β)	1	2	...	10	Average
Static/ α=0.6	1	1.8	399	533	...	419	470.4
	20	1.2	649	740	...	837	786.6
	20	1.5	391	462	...	410	483.8
	40	1.3	439	475	...	491	498.5
	40	1.4	379	424	...	945	474.6
	80	1.1	670	665	...	1,402	801.1
	80	1.2	450	430	...	554	565.0
						582	442.9

4.039x470.4/100=19.0

Rho (ρ)		Dynamic α			Static α			
Start	Update Parameter(β)	Δα=0.02	Δα=0.03	Δα=0.05	α=0.6	α=0.7	α=0.8	Total
1	1.8	18.95	18.52	25.28	19.00	18.89	25.17	125.8
20	1.2	26.58	22.53	26.33	28.07	16.54	23.88	143.9
20	1.5	13.30	18.37	20.20	19.66	17.09	18.65	107.3
40	1.3	14.58	14.78	24.30	18.83	18.83	18.14	109.5
40	1.4	20.23	18.49	18.03	18.02	14.32	15.12	104.2
80	1.1	21.73	21.62	24.26	16.23	17.67	17.02	118.5
80	1.2	15.88	17.92	21.27	**8.70**	17.83	20.33	101.9
200	1.1	14.75	12.09	14.77	12.59	13.48	11.98	**79.6**
	Total:	146.0	**144.3**	174.5	**141.1**	**134.6**	150.3	

Figure 9: Creating Parameter Selection Index

Further, among all the experiments, the best parameter configuration in terms of index is with static $\alpha = 0.6$, when starting penalty factor $\rho = 80$ and update parameter $(\beta) = 1.2$. In Table 4 and Table 12, this configuration of parameter selection provides the best average gap performance and a good CPU time performance. Hence we also included this parameter configuration in our experiments.

Table 6: Index for Parameter Selection

Rho (ρ)		**Dynamic α**			**Static α**			
Start	**Update Parameter(β)**	**Δα=0.02**	**Δα=0.03**	**Δα=0.05**	**α=0.6**	**α=0.7**	**α=0.8**	**Total**
1	**1.8**	18.95	18.52	25.28	19.00	18.89	25.17	125.8
20	**1.2**	26.58	22.53	26.33	28.07	16.54	23.88	143.9
20	**1.5**	13.30	18.37	20.20	19.66	17.09	18.65	107.3
40	**1.3**	14.58	14.78	24.30	18.83	18.83	18.14	109.5
40	**1.4**	20.23	18.49	18.03	18.02	14.32	15.12	104.2
80	**1.1**	21.73	21.62	24.26	16.23	17.67	17.02	118.5
80	**1.2**	15.88	17.92	21.27	**8.70**	17.83	20.33	101.9
200	**1.1**	14.75	12.09	14.77	12.59	13.48	11.98	**79.6**
	Total:	146.0	**144.3**	174.5	**141.1**	**134.6**	150.3	

In the remainder of the computational experiments, we used sample size and number as $(M, N) = (5, 10)$, which enables the SBPHA to search the solution space while maintaining computational time efficiency.

3.1.3. Computational Performance of SBPHA and d-SBPHA

In this section, we first show the performance of the *d*-SBPHA in improving the solution quality of SBPHA and then compare the performances of the SAA and the proposed d-SBPHA algorithm.

In the remainder of the experiments, with an abuse of the optimality definition, we refer to the best solution as the “exact solution”. This solutions is is obtained by selecting the best amongst all SBPHA, d-SBPHA, and SAA solutions and the time-restricted solution of the CPLEX. The latter solution is obtained by solving the deterministic equivalent using CPLEX method with %0.05 optimality gap tolerance and 10 hours (36,000 seconds) of time limit until either the CPU time-limit is exceeded or the CPLEX terminates due to insufficient memory.

3.1.3.1 Analyze on d-SBPHA and SBPHA

Figure 10 shows effect of discarding strategy on the solution quality for different facility failure probabilities. In all figures, results are based on the average of 10 replications. Optimality gap (shown as ‘Gap’) is calculated as in (3.7) but substituted v_r^{SBPHA} with $v_{best,r}^{d-SBPHA}$ in (3.8) to calculate v_{Rep}^{Average}. First observation is that the *d*-SBPHA not only improves solution quality but also finds the optimal solution in most facility failure probabilities cases. When failure probability is $q = 0.1$, *d*-SBPHA converges to optimal solution in less than 5 discarding iterations with all parameter configurations (Figure 10a). When $q = 0.2$, *d*-SBPHA converges to optimal solution in all static α strategies in less than 5 discarding strategies, and less than 0.2% optimality gap with dynamic α strategy (Figure 10b).

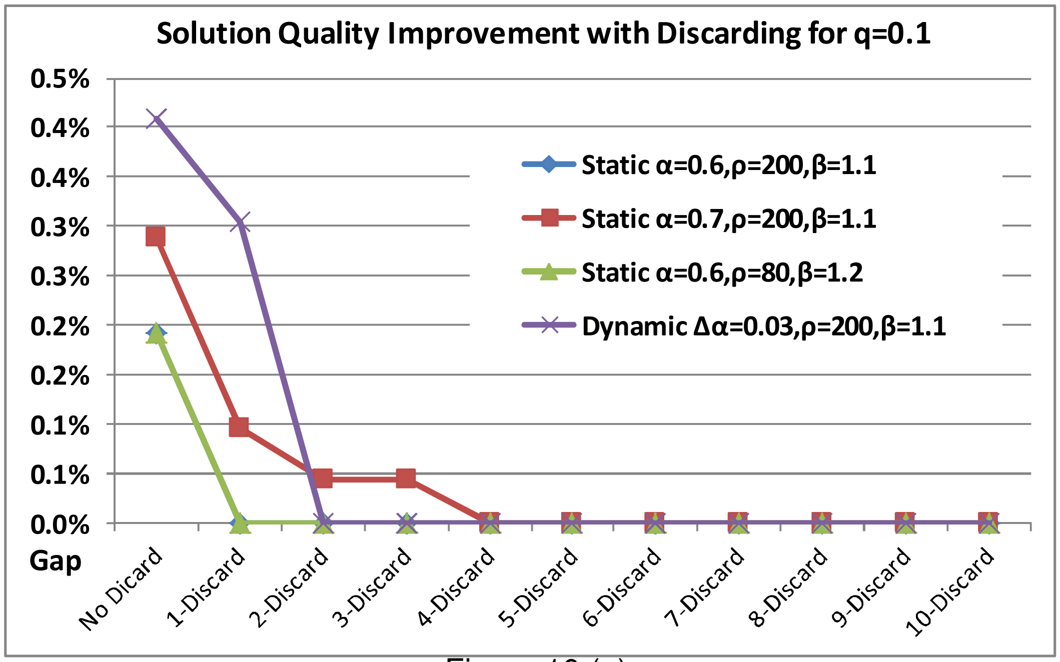


Figure 10 (a)

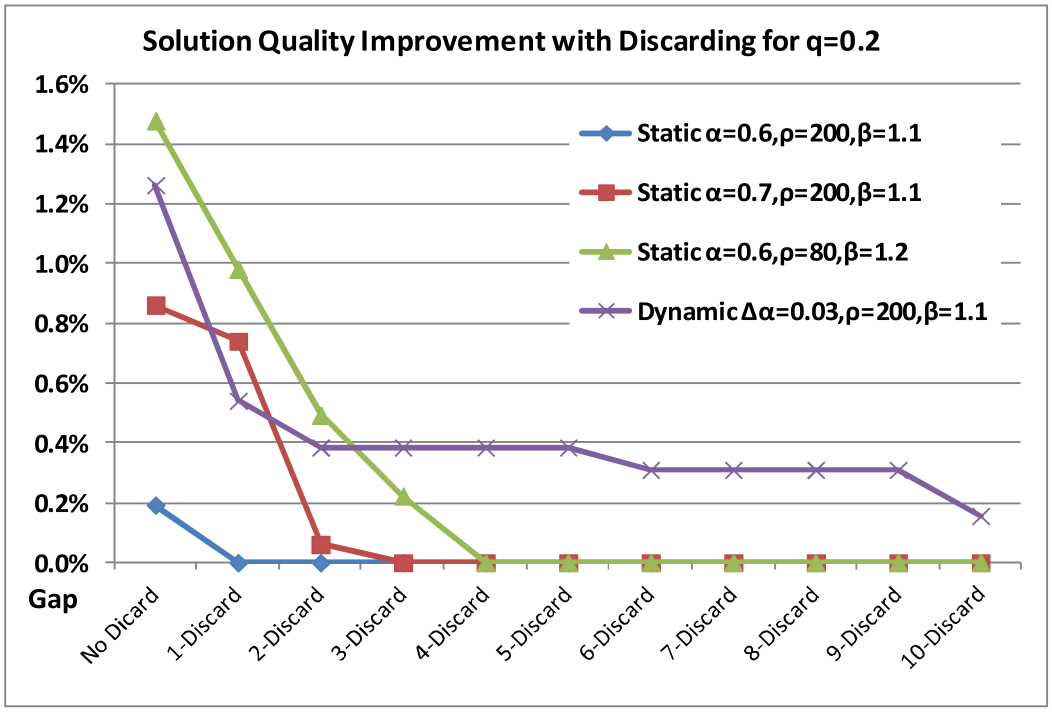


Figure 10 (b)

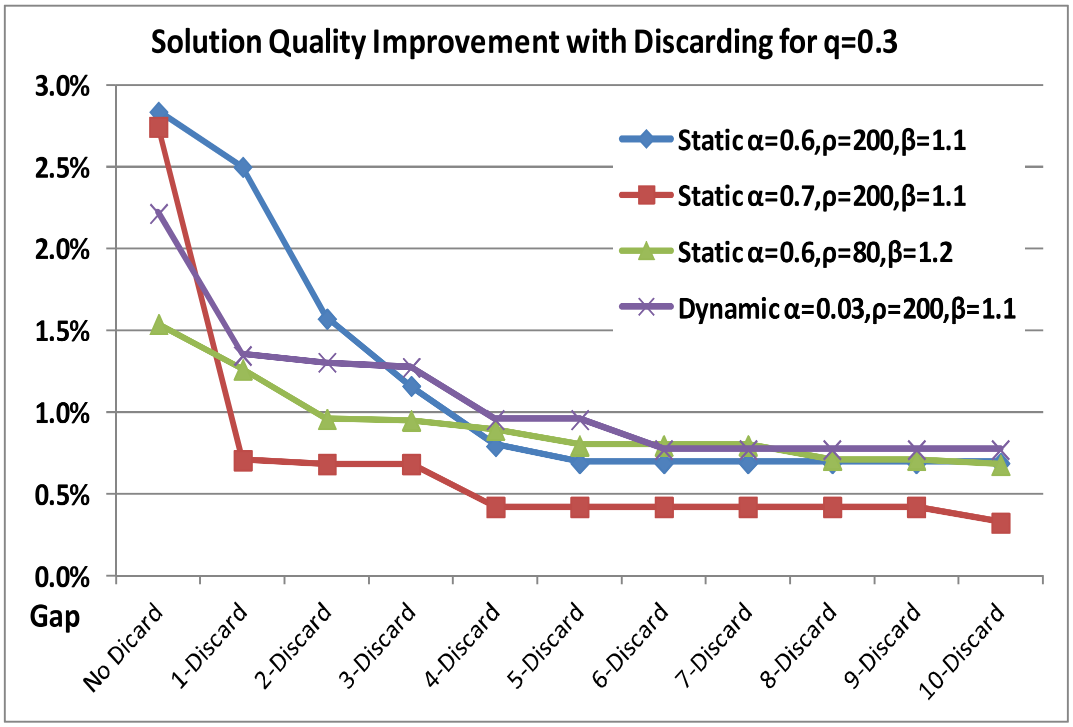


Figure 10

Further, when failure probability is 0.3, *d*-SBPHA is not able converge to optimal solution; however it converges to solutions that are less than 1% away from the optimal on average (Figure 10c). Note that these results are based on the average of 10 replications, and at least 5 out of 10 replications are converged to optimal solution in all parameter configurations. Detailed results are provided in the next section.

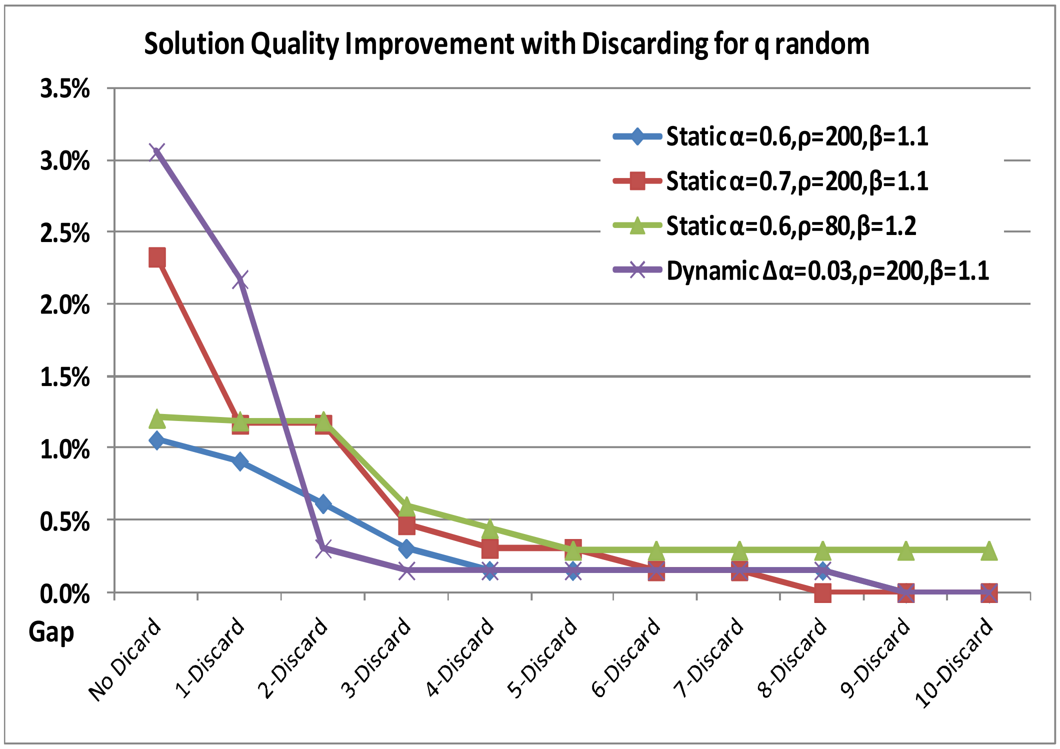


Figure 10 (d)

Figure 10: Effect of Discarding Strategy on the Solution Quality for CRFLP with Facility Failure Probabilities (a) $q = 0.1$, (b) $q = 0.2$, (c) $q = 0.3$, and (d) when q is random.

Lastly, when failure probability is randomized, *d*-SBPHA converges to optimal solution in *10* discarding iterations, with three out of the four selected parameter configurations (Figure 10d). Hence, we conclude that discarding improves the solution performance and the improvement rates depend on the parameter configuration and the problem parameters. Reader is referred to Table 14 for detailed results.

Next we present the CPU time performance of *d*-SBPHA for 10 discarding iterations (Figure 11). Note that time plotted is cumulative over discarding iterations. Time of each discarding iteration is based on average of 10 replications and time is reported in seconds.

First observation is that the CPU time is linearly increasing or increasing at a decreasing rate. Further, the solution time is similar for all facility failure probabilities (Figure 11 a, b, c and d) with all parameter configurations.

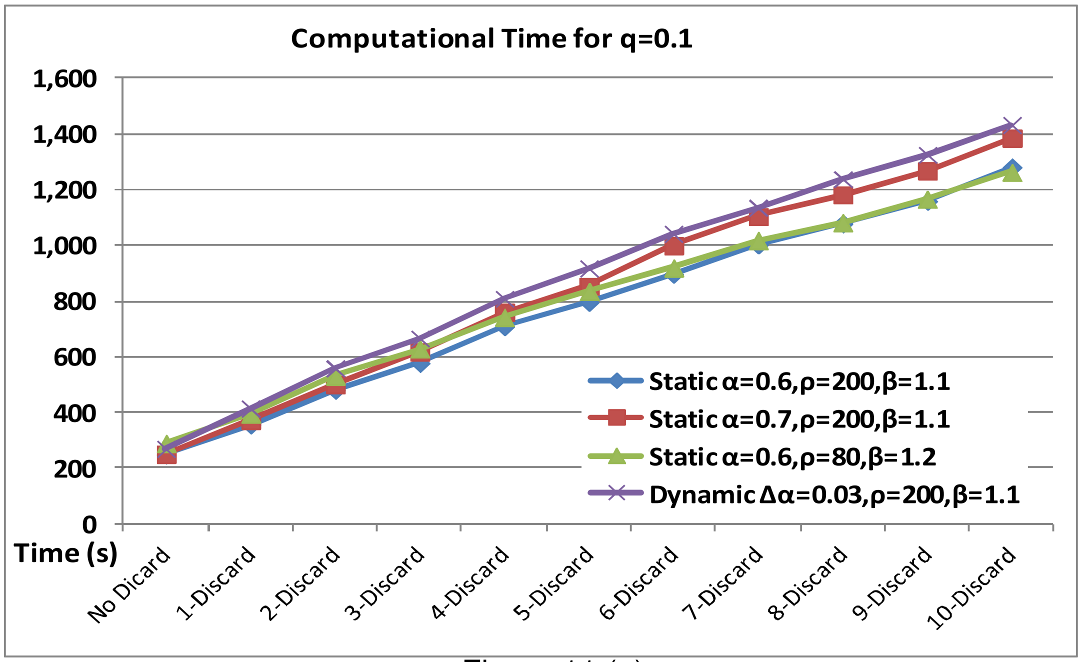


Figure 11 (a)

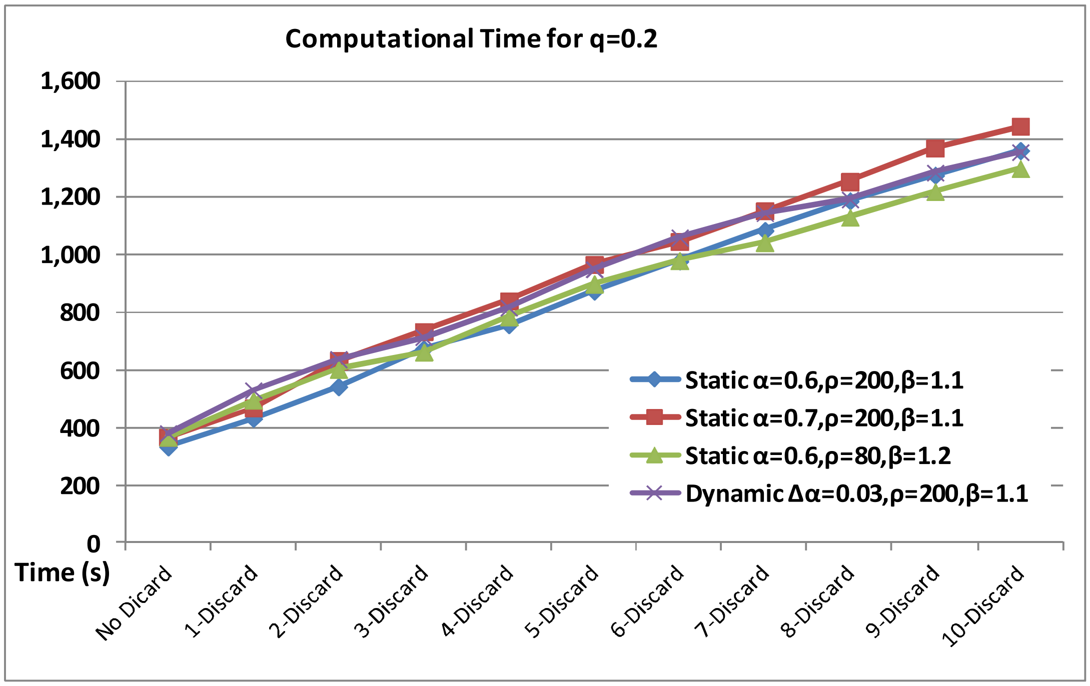


Figure 11(b)

Full results are provided in the Appendix in Table 15. One main reason for linearly increasing CPU time is that the *d*-SBPHA uses the carried over information. In particular, *d*-SBPHA does not test any first stage solution performance in the reference sample (N') if the first stage solution is already tested before.

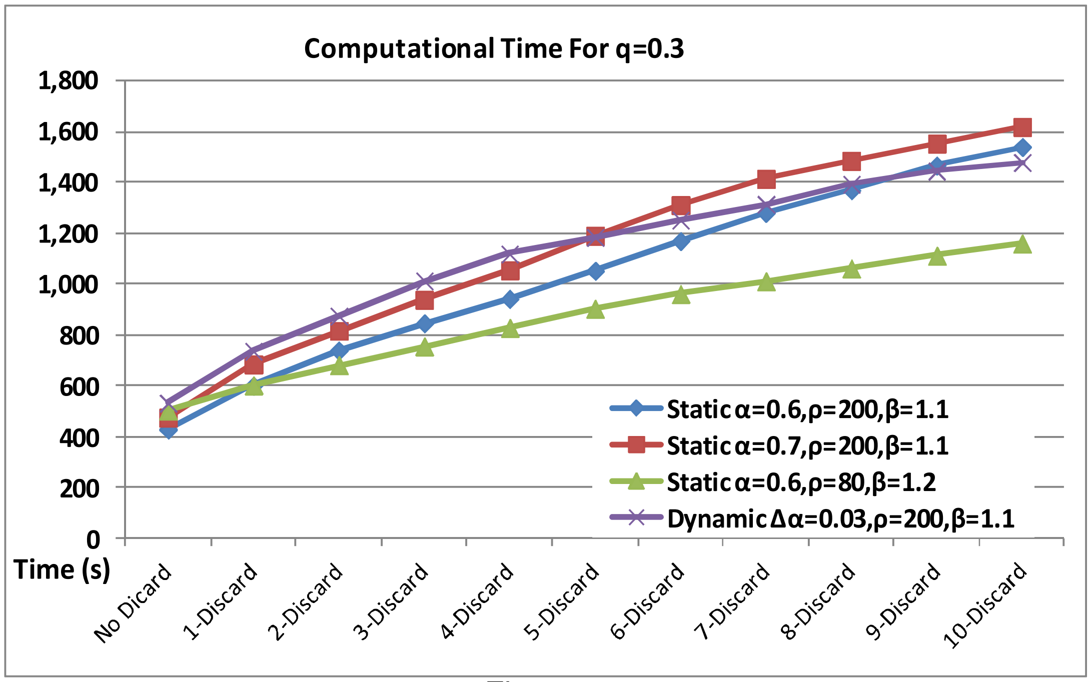


Figure 11
(a)

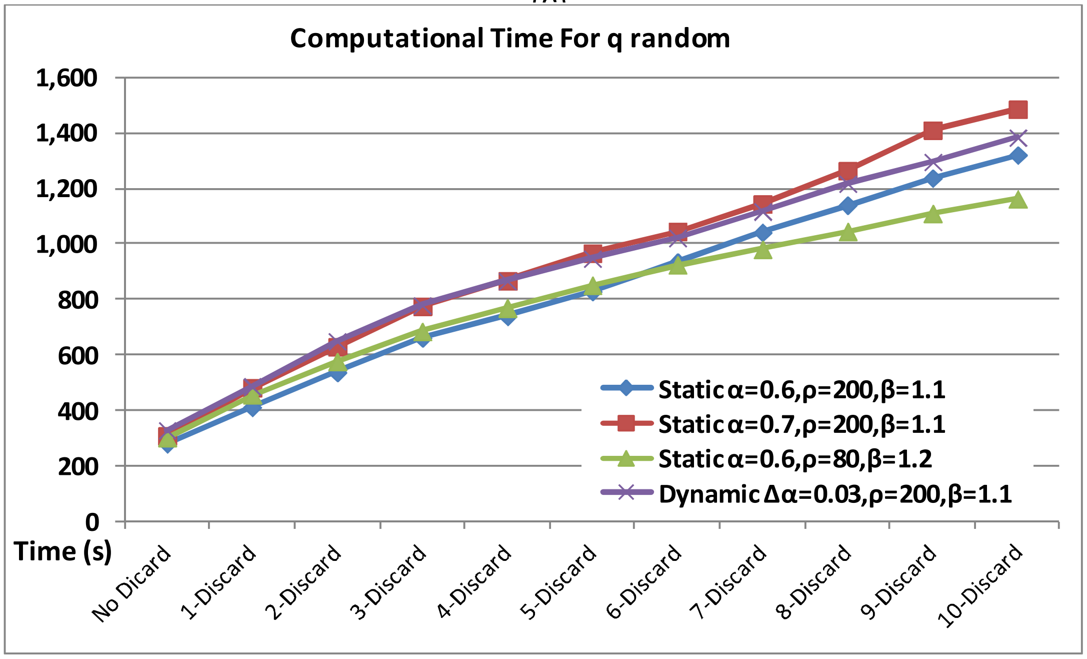


Figure 11

Figure 11: CPU Time Performance of Discarding Strategy for CRFLP with Facility Failure Probabilities (a) $q = 0.1$, (b) $q = 0.2$, (c) $q = 0.3$, and (d) when q is random.

3.1.3.2 SAA, SBPHA and d-SBPHA Tests, Comparisons

In this section, we compare the performances of the SBPHA, *d*-SBPHA, and SAA. First, we analyzed the performance of the proposed SBPHA and d-SBPHA with respect to that of the exact method and the SAA method with different sample sizes (N) and number of samples (M). We only use one of the parameter configuration, $\alpha = 0.7$, $\rho = 200$,and $\beta = 1.1$.

Table 7 and Table 8 illustrate these benchmark results for $q = \{0.1, 0.2, 0.3, random\}$ for one of the replications and average results across all replications are shown in Figure 12. The second column for SAA shows number of samples and sample size, i.e., (M, N). For *d*-SBPHA, it shows number of replications r, (M, N), and number of discarding iterations (d). Note that when the number of discarding iterations is $d = 0$, *d*-SBPHA becomes SBPHA. Third column, F^*, indicates the solution converged by each method, e.g., facilities opened. For instance with $q = 0.3$, the SAA's solution is to open facilities $F^* = \{1,2,8,10,12\}$ whereas the SBPHA opens facilities $F^* = \{1,2,4,10,12\}$, *2*-SBPHA and exact solution opens facilities $F^* = \{1,2,10,11,12\}$.

Fourth column presents the objective function value for SAA, SBPHA, d-SBPHA and exact method, $v^{SAA}, v^{SBPHA}, v_{best}^{d-SBPHA}$ and v^*. Fifth column presents CPU time and the sixth column shows the optimality gap (Gap_2) measures. Reported time for *d*-SBPHA is the average time of converged solution is found during the discarding iterations. Gap2 for SAA, SBPHA and d-SBPHA uses the optimal solution value v^*. It is defined as,

$$Gap_2 = \begin{cases} \frac{v^{SAA}-v^*}{v^*}\text{x}100\% & \text{for SAA,} \\ \frac{v^{SBPHA}-v^*}{v^*}\text{x}100\% & \text{for SBPHA,} \\ \frac{v_{best}^{d-SBPHA}-v^*}{v^*}\text{x}100\% & \text{for } d-\text{SBPHA.} \end{cases} \tag{3.9}$$

Table 7: Solution Quality and CPU Time Performances of the SAA, SBPHA and *d*-SBPHA for CRFLP with Facility Failure Probabilities $q = 0.1$ and $q = 0.2$.

		q=0.1				q=0.2			
Method	**M-N**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**
SAA	**5-10**	1,2,8,10,12	6,115	53	2.1	1,2,4,11,12	10,055	59	35.0
	5-25	1,2,8,10,12	6,115	132	2.1	1,2,7,8,10,12	7,856	152	5.5
	5-40	1,2,4,11,12	6,047	208	1.0	1,2,4,7,11,12	7,762	326	4.2
	5-50	1,2,4,11,12	6,047	307	1.0	1,2,8,10,11,12	7,649	325	2.7
	5-75	1,2,10,11,12	5,990	695	0.0	1,2,4,8,11,12	7,706	741	3.5
	10-10	1,2,8,10,12	6,115	94	2.1	1,3,4,8,10,12	8,465	122	13.7
	10-25	1,2,4,11,12	6,047	267	1.0	1,2,3,4,10,12	7,856	277	5.5
	10-40	1,2,4,11,12	6,047	445	1.0	1,2,4,7,11,12	7,762	646	4.2
	10-50	1,2,4,11,12	6,047	638	1.0	1,2,8,10,11,12	7,649	676	2.7
	10-75	1,2,10,11,12	5,990	1,430	0.0	1,2,8,10,11,12	7,649	1,636	2.7
	20-10	1,2,8,10,12	6,115	174	2.1	1,4,5,8,11,12	8,450	236	13.5
	20-25	1,2,4,11,12	6,047	526	1.0	1,2,7,8,10,12	7,856	558	5.5
	20-40	1,2,4,11,12	6,047	874	1.0	1,2,8,10,11,12	7,649	1,222	2.7
	20-50	1,2,4,11,12	6,047	1,340	1.0	1,2,4,10,11,12	7,614	1,368	2.2
	20-75	1,2,10,11,12	5,990	2,791	0.0	1,2,4,10,11,12	7,614	3,348	2.2
		Static α=0.7,ρ=200, β=1.1				**Static α=0.7,ρ=200, β=1.1**			
Method	**rxM-N,d**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**
SBPHA	**1x5-10, 0**	1,2,4,10,12	6,106	183	1.9	1,2,4,5,11,12	7,447	331	0.0
***d*-SBPHA**	**1x5-10, 1**	1,2,10,11,12	5,990	309	0.0	1,2,4,5,11,12	7,447	437	0.0
	1x5-10, 2	1,2,10,11,12	5,990	387	0.0	1,2,4,5,11,12	7,447	669	0.0
SBPHA	**2x5-10, 0**	1,2,10,11,12	5,990	323	0.0	1,2,4,5,11,12	7,447	976	0.0
***d*-SBPHA**	**2x5-10, 1**	1,2,10,11,12	5,990	652	0.0	1,2,4,5,11,12	7,447	1,211	0.0
	2x5-10, 2	1,2,10,11,12	5,990	810	0.0	1,2,4,5,11,12	7,447	1,713	0.0
SBPHA	**3x5-10, 0**	1,2,10,11,12	5,990	529	0.0	1,2,4,5,11,12	7,447	1,159	0.0
***d*-SBPHA**	**3x5-10, 1**	1,2,10,11,12	5,990	932	0.0	1,2,4,5,11,12	7,447	1,506	0.0
	3x5-10, 2	1,2,10,11,12	5,990	1,168	0.0	1,2,4,5,11,12	7,447	2,309	0.0
Exact	-	1,2,10,11,12	5,990	>>10800	-	1,2,4,5,11,12	7,447	>>14400	-

Table 8: Solution Quality and CPU Time Performances of the SAA, SBPHA and d-SBPHA for CRFLP with Facility Failure Probabilities $q = 0.3$ and q random.

		q=0.3				q random			
Method	**M-N**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**
SAA	**5-10**	1,2,4,7,10,12	11,877	119	32.0	2,4,5,11,12	7,609	52	12.2
	5-25	1,2,4,5,8,10,12	9,798	342	8.9	2,5,8,10,11,12	7,310	116	7.8
	5-40	1,2,7,8,10,11,12,	9,658	815	7.4	1,2,4,10,11,12	6,979	236	2.9
	5-50	1,2,4,5,10,11,12	9,645	1,378	7.2	2,4,5,8,11,12	7,169	291	5.7
	5-75	1,2,8,10,11,12,14	9,083	4,570	1.0	2,3,4,5,10,12	7,016	816	3.5
	10-10	1,2,4,11,12,14	9,839	236	9.4	2,4,5,11,12	7,609	99	12.2
	10-25	1,2,4,7,8,10,12	9,744	708	8.3	1,2,4,10,11,12	6,979	234	2.9
	10-40	1,2,11,12,14,18	9,096	2,044	1.1	1,2,4,5,10,12	6,887	474	1.6
	10-50	1,2,11,12,14,18	9,096	3,468	1.1	1,2,4,5,10,12	6,887	696	1.6
	10-75	1,2,8,10,11,12,14	9,083	7,746	1.0	2,4,5,10,11,12	6,780	1,550	0.0
	20-10	1,2,4,11,12,14	9,839	487	9.4	2,4,5,10,12	7,598	201	12.1
	20-25	1,2,7,8,10,11,12,	9,658	1,509	7.4	1,2,4,10,11,12	6,979	481	2.9
	20-40	1,2,11,12,14,18	9,096	3,960	1.1	2,4,5,10,11,12	6,780	893	0.0
	20-50	1,2,11,12,14,18	9,096	6,367	1.1	2,4,5,10,11,12	6,780	1,359	0.0
	20-75	1,2,4,10,11,12,14	8,995	13,867	0.0	2,4,5,10,11,12	6,780	2,824	0.0
		Static α=0.7,ρ=200, β=1.1				**Static α=0.7,ρ=200, β=1.1**			
Method	**rxM-N,d**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**	**F***	**Objective**	**Time(sec.)**	**Gap2(%)**
SBPHA	**1x5-10, 0**	1,2,4,10,11,12,20	9,024	422	0.3	2,4,5,7,10,12	7,263	250	7.1
***d*-SBPHA**	**1x5-10, 1**	1,2,4,10,11,12,20	9,024	456	0.3	2,4,5,10,11,12	6,780	467	0.0
	1x5-10, 2	1,2,4,10,11,12,20	9,024	490	0.3	2,4,5,10,11,12	6,780	590	0.0
SBPHA	**2x5-10, 0**	1,2,4,10,11,12,20	9,024	804	0.3	1,2,4,5,10,12	6,887	603	1.5
***d*-SBPHA**	**2x5-10, 1**	1,2,4,10,11,12,14	8,995	999	0.0	2,4,5,10,11,12	6,780	901	0.0
	2x5-10, 2	1,2,4,10,11,12,14	8,995	1,203	0.0	2,4,5,10,11,12	6,780	1,270	0.0
SBPHA	**3x5-10, 0**	1,2,4,10,11,12,14	8,995	1,503	0.0	2,4,5,10,11,12	6,780	1,000	0.0
***d*-SBPHA**	**3x5-10, 1**	1,2,4,10,11,12,14	8,995	2,093	0.0	2,4,5,10,11,12	6,780	1,470	0.0
	3x5-10, 2	1,2,4,10,11,12,14	8,995	2,442	0.0	2,4,5,10,11,12	6,780	1,951	0.0
Exact	-	1,2,4,10,11,12,14	8,995	>>21600	-	2,4,5,10,11,12	6,780	>>10800	-

Table 7 and

Table 8 show that with larger sample sizes, the objective function on the SAA's objective function is not always monotonously decreasing while the CPU time increases exponentially. The observation about the time is in accordance with those in Figure 12. SAA finds optimal solution only when N=75 for $q = 0.1$ and cannot find optimal solution for $q = 0.2$ with any of M-N configurations. SAA, also finds optimal solution for $q = 0.3$ only when M=20 and N=75 in more than 13,000 seconds and shows relatively good performance for random q when M=20. *d*-SBPHA finds optimal solution in all facility failure probabilities.

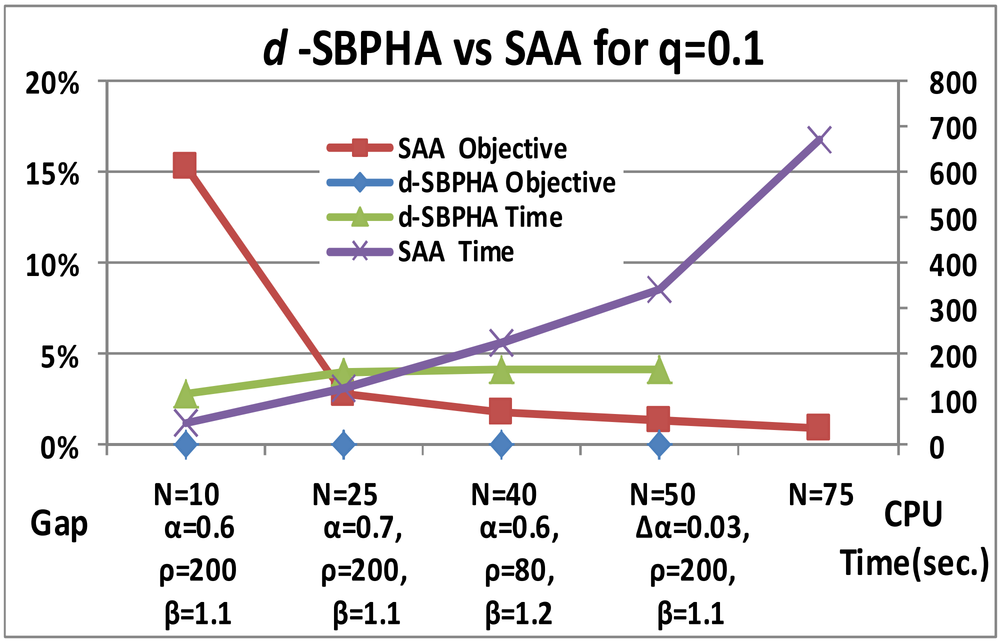


Figure 12 (a)

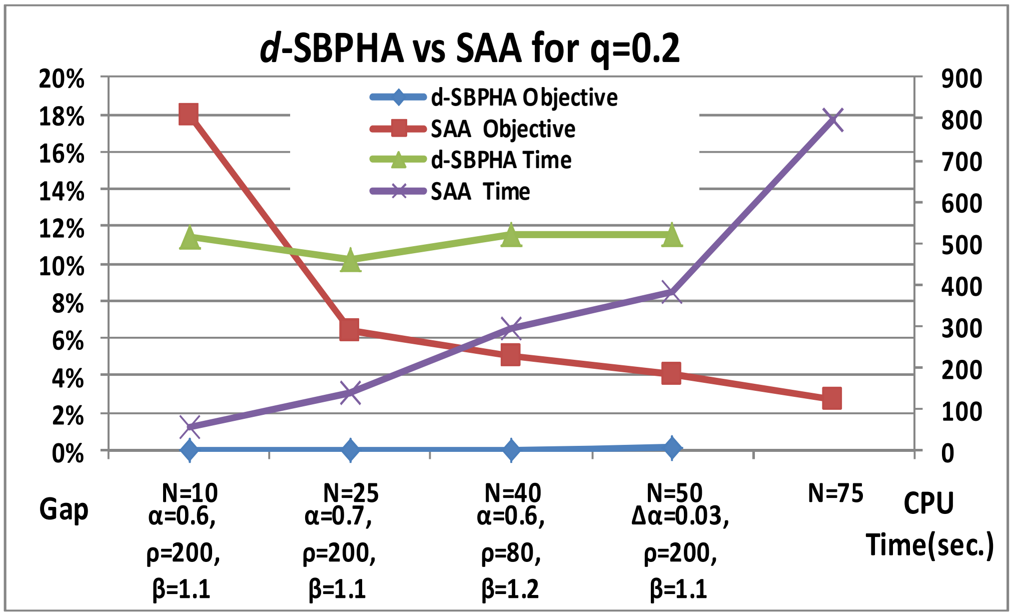


Figure 12 (b)

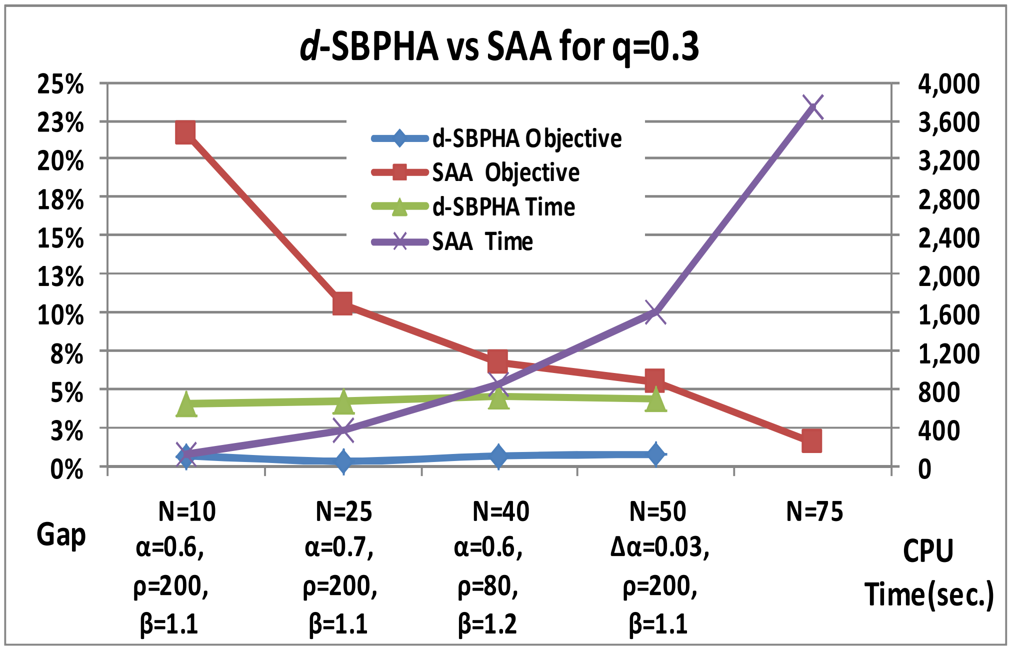


Figure 12 (c)

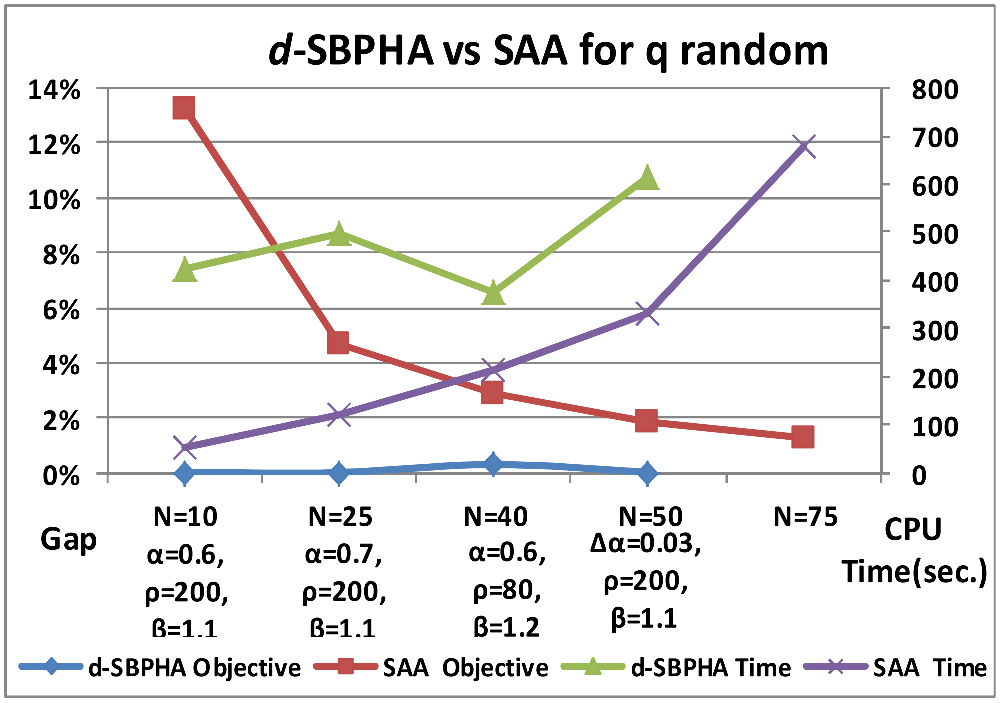


Figure 12 (d)

Figure 12: Effect of Sample Size on the Solution quality and CPU Time performance of SAA in Comparison with *d*-SBPHA for CRFLP with Facility Failure Probabilities (a) $q = 0.1$, (b) $q = 0.2$, (c) $q = 0.3$, (d) q random.

Results for *d*-SBPHA ($d = 10$) in Figure 12 are for all four parameter settings; first setting is for static $\alpha = 0.6, \rho = 200$, and $\beta = 1.1$, second is static $\alpha = 0.7, \rho = 200$, and $\beta = 1.1$, third is static $\alpha = 0.6, \rho = 200$, and $\beta = 1.2$, and fourth one is dynamic $\Delta_\alpha = 0.03$, $\rho = 200$, and $\beta = 1.1$.

In Figure 12, we present the CPU time and solution quality performances of the SAA for N=$\{10,25,40,50,75\}$ sample sizes and compare with that of the proposed *d*-SBPHA, in which $d = 10$, with N=10 in solving CRFLP with failure probabilities $\mathrm{q} = \{0.1, 0.2\ ,0.3, \mathrm{random}\}$. We use M=5 samples in both methods and four different parameter configurations in the proposed method. The results indicate that the sample size effect on the SAA's CPU time is high. For instance, the CPU time of the SAA is growing exponentially. In none of the failure probability cases, however, the solution quality performance of SAA has converged to that of *d*-SBPHA. The solution quality of all four configurations of the proposed d-SBPHA are either optimal or near optimal. The Gap in Figure 12 is calculated as in Figure 10 and the CPU time of *d*-SBPHA shows the average CPU time when the converged solution is found during the discarding iterations.

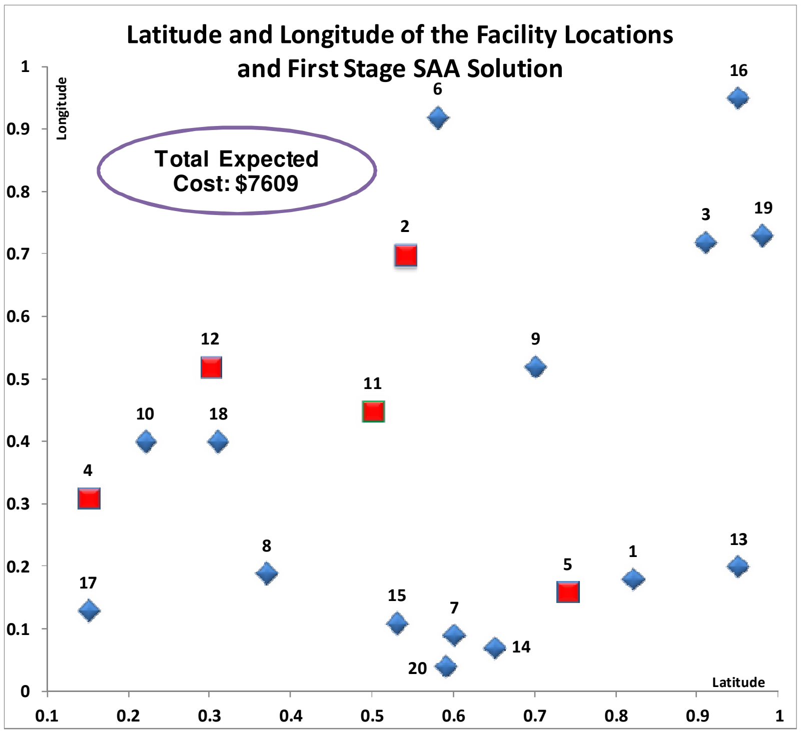


Figure 13: Latitude-Longitude of the Facilities and First Stage Solution of SAA

Now we are interested in presenting an illustrative example of the first stage decisions (F^{*}) start with SAA decisions, continue with SBPHA and end with *d*-SBPHA algorithms. Let's illustrate solution of one of the replications shown in

Table 8. The first stage solution of SAA, when random failure probability is selected, is $\mathrm{F}^{*}=\{2,4,5,11,12\}$. Please see latitude-longitude positions of the facilities (these are also demand points) and SAA' first stage solution in Figure 13. The total expected cost of the objective function is \$7,609 for the solution that SAA provides. SBPHA starts with this solution and improves iteratively. SAA opens 5 facilities while SBPHA opens 6 facilities. The solution that SBPHA provides is $\mathrm{F}^{*}=\{2,4,5,7,10,12\}$. SBPHA decide not to open facility 11, instead to open two new facilities $(7,10)$. The total expected cost of the objective function value is reduced to \$7,263. The solution

that SBPHA provides offers more than 5% saving in total expected cost. Graphically illustration of the SBPHA's solution is shown in Figure 14.

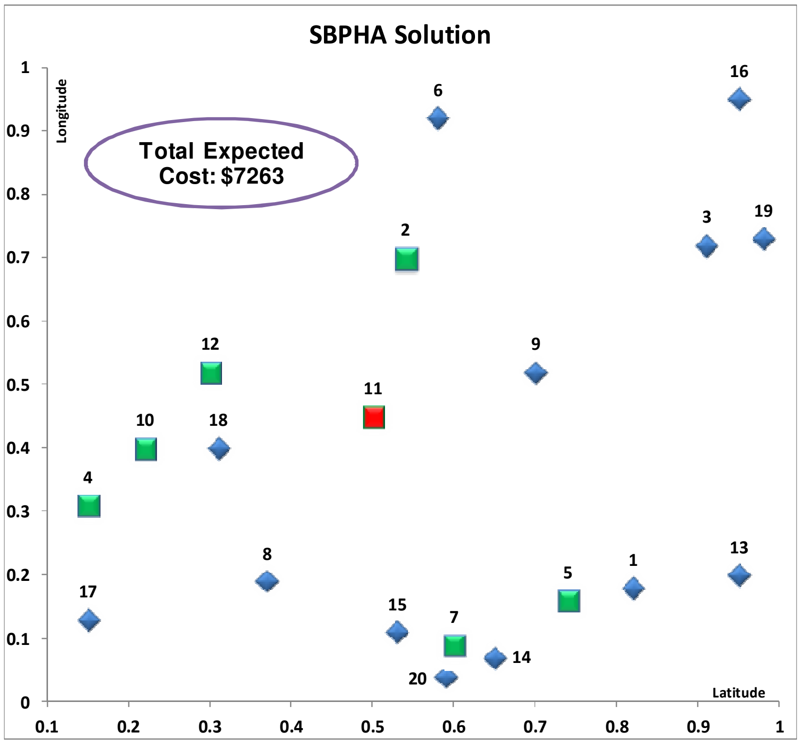


Figure 14: SBPHA's First Stage Solution

Next we start 1-SBPHA (d-SBPHA algorithm with one discarding iteration). The 1-SBPHA algorithm starts with discarding the solution that SBPHA provides and analyzes search space to come up with a better solution than that of SBPHA provides. 1-SBPHA decides not to open facility 7 and instead to reopen facility 11 that SAA decides to open at first. Then, the solution that 1-SBPHA provides is $F^* = \{2,4,5,10,11,12\}$ with a total expected cost of $6780. In total 1-SBPHA decides to open 6 facilities as SBPHA decides, opposed to SAA's decision. Graphically illustration of the *1*-SBPHA's solution is shown in Figure 15.

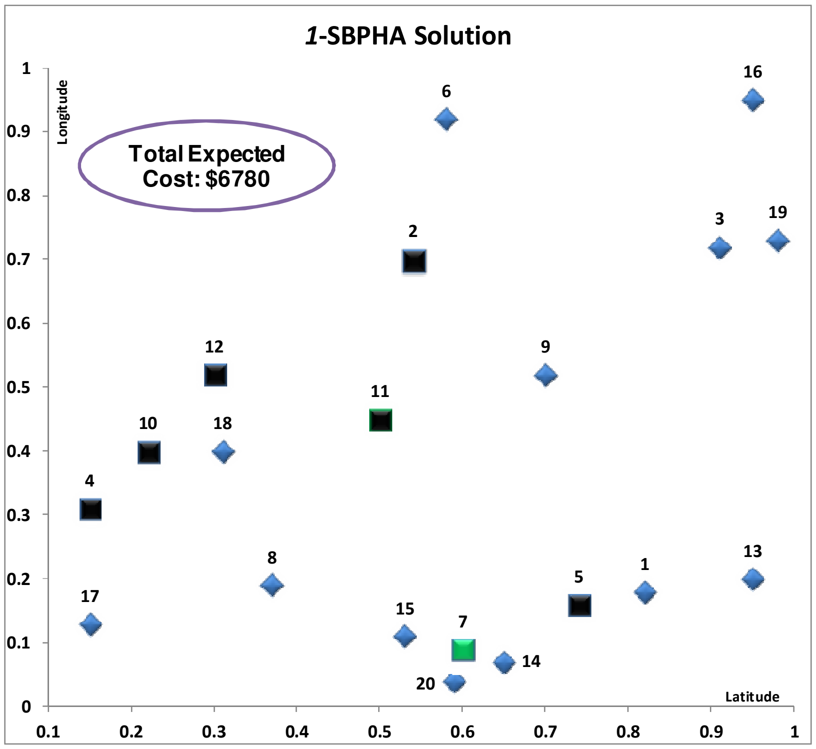


Figure 15: *1*-SBPHA's and Optimal Solution of the First Stage

1-SBPHA improves solution quality of SBPHA while SBPHA improves SAA's solution. 1-SBPHA reduced total expected cost of the objective function value that SBPHA provides by $483, which is more than 7% of the optimal objective function value. In total more than 12% of the total expected objective function value is reduced. The solution and the objective function value that 1-SBPHA provides is the optimal solution and objective function value of the problem. For this illustrative example, in both SBPHA and 1-SBPHA static $\alpha = 0.7, \rho = 200, \beta = 1.1$, parameters configuration is used.

3.2. Multi-Stage Stochastic Lot-Sizing Problem

In this section we describe the stochastic lot sizing problem that is used to test the SBPHA method proposed for multi-stage SP stochastic programming problems.

Production planning’s aim is to employ the resources in order to satisfy production goal such as demand, etc. over a time period (Karimi, Ghomi et al. 2003). Lot sizing models determine the best timing and level of production (Jans and Degraeve 2007). Lot sizing problems essentially can be categorized as: single or multiple products, with fixed cost or without fixed cost, with capacity or without capacity, with linear or nonlinear cost function, with bill of materials or without bill of materials, single period or multi period, with backlogging or without backlogging and stochastic or deterministic. Some solution methodologies that are applied to lot sizing problems are Lagrangean relaxation, Benders decomposition and column generation as decomposition methods, also Adhoc Heuristics, Meta Heuristics, Dynamic Programming and Approximation methods have been utilized to tackle lot sizing problems.

There are several good review papers on lot sizing problems such as,Kuik, Salomon et al. (1994) which discusses the production planning and lot sizing impacts. A good review on the history of the single item lot sizing is provided in (Wolsey 1995). Single item uncapacitated problem is the simplest structure of dynamic lot-sizing problem and first discussed by Wagner and Whitin (1958). Zangwill (1969) describe the lot sizing problem as a network problem.

Mathematical formulation of the scenario based stochastic lot-sizing problem can be formulated as follows (Haugen, Lokketangen et al. 2001):

<u>Notation for stochastic lot-sizing:</u>

$y_{s,t}$: indicator of the production for scenario s, in period t

$x_{s,t}$: amount to be produced in period scenario s, in period t

$I_{s,t}^{+}$: positive (carried over) ending inventory for scenario s, in period t

$I_{s,t}^{-}$: negative (backlogged) ending inventory for scenario s, in period t

I_0 : initial inventory

$I_{s,t}^{-}$: negative (backlogged) ending inventory for scenario s, in period t

f_t : set up cost in period t

c_t : production cost per product in period t

h_t^{+} : inventory holding cost in period t

h_t^{-} : shortage cost in period t

d_t : demand in period t

Scenario Based Stochastic Lot-Sizing Problem:

$$\text{Min} \sum_{s\in S} p_s \sum_{t=1}^{T} [f_t y_{s,t} + c_t x_{s,t} + h_t^{+} I_{s,t}^{+} + h_t^{-} I_{s,t}^{-}] \quad (3.10)$$

Subject to

$$x_{s,t} + I_{s,t-1}^{+} - I_{s,t-1}^{-} + I_{s,t}^{-} = d_{s,t} \ \forall s \in S, t \in \{1, \dots, T\} \quad (3.11)$$

$$x_{s,t} \geq 0 \ \forall s \in S, t \in \{1, \dots, T\} \quad (3.12)$$

$$I_{s,t}^{+} \geq 0 \ \forall s \in S, t \in \{1, \dots, T\} \quad (3.13)$$

$$I_{s,t}^{-} \geq 0 \ \forall s \in S, t \in \{1, \dots, T\} \quad (3.14)$$

$$y_{s,t} = \begin{cases} 0 \text{ if } x_{s,t} = 0 \\ 1 \text{ if } x_{s,t} > 0 \end{cases} \ \forall\, t \in \{1, \dots, T\} \quad (3.15)$$

$$x_{s,t} - x_{s',t} = 0, y_{s,t} - y_{s',t} = 0, I_{s,t}^{+} - I_{s',t}^{+} = 0, I_{s,t}^{-} - I_{s',t}^{-} = 0,$$

$$\forall \mathrm{s}, \mathrm{s}' \in \mathrm{S}: (\xi_1^{\mathrm{s}}, \dots, \xi_{\mathrm{t}}^{\mathrm{s}}) = (\xi_1^{\mathrm{s}'}, \dots, \xi_{\mathrm{t}}^{\mathrm{s}'}), \mathrm{t} = 1,2, \dots, \mathrm{T} \quad (3.16)$$

where (3.10) is the objective function which minimizes the total production, carrying over and shortage cost; constraints in (3.11) are the flow conservation constraints and allow carrying over inventory for the next periods' demand and backlogging if not all the demand is satisfied, (3.12)-(3.14) are non-negativity constraints. Constraints in (3.15) force the set up cost if any production done in the last period. Lastly, constraints in (3.16) ensure the non-anticipativity. Production and inventory decision variables are continuous and non-negative, while the set up decisions are binary type.

The more practical formulations of the lot sizing problems are those which consider capacity restriction and multiple items. The version of the capacitated lot sizing without set up cost studied by Karmarkar and Schrage (1985). Miller and Wolsey (2003) studied the discrete lot sizing problem with set ups but without startups. Mekler (1993) and Denizel, Erengüç et al. (1997) discussed the tradeoff between set up cost and benefit in terms of reducing set up time.

Capacity production tools can be considered as a constraint in production plants if company has limited tools and equipment. Jans and Degraeve (2004) model a production planning model which limits the production tools. Akturk and Onen (2002) combined the tool management problem and lot sizing problem. Machine capacity is studied various ways, such as decision on increasing number of machines or using different types (Clark and Clark 2000). Another important topic that is studied in lot sizing problems is inventory. Backlogging allows the company to satisfy demand at a later period. In other words, it is negative inventory which is going to be satisfied later. Clearly, backlogging has a cost, since the company cannot satisfy the customers' demands on time. There are two effects of backlogging: earning less than planned because of late demand satisfaction, discounts, and loosing customer in the future (Aksen, Altinkemer et al. 2003). If demand is not satisfied on time instead of backlogging lost sales occur. Backlogging case can be considered as single source fixed charge network problem (Zangwill 1969).

In real life production planning (schedules) is always applied according to rolling horizon methodology. They apply the plan for the closest period by considering 1-2 periods further. At the end of each period plans and schedules are updated by considering realization of the demand and supply. Baker (1977) indicated that small planning horizon plans can be perform optimally by heuristics and with the rolling horizon technique Wagner-Whitin algorithm is no longer optimal.

There are many studies considering stochastic lot sizing problems in the literature. In lot sizing problems, uncertainty can stem from many parameters such as, demand, capacity, supply, source, set up time, production time etc. Sox (1997) have studied static control policies underneath the non-stationary stochastic demand supposition in rolling horizon structure. Within this study a mixed integer lot-sizing problem with dynamic costs is presented. Alp, Erkip et al. (2003) introduced dynamic lot/batch sizing with deterministic demands and stochastic lead times. Ahmed, King et al. (2003) study the stochastic lot-sizing problem. In this study, demand follows a stochastic process which is described by a scenario tree. Guan, Ahmed et al. (2006) to solve the stochastic lot-sizing problem with zero lead times he proposed a branch-and-cut algorithm.

Lot-sizing has a wide literature in terms of problem types and many solution methodology and techniques have been applied to tackle with, such as, branch and bound, cut-generation (Barany, Roy et al. 1984). Other than exact methods (Barany, Roy et al. 1984; Belvaux and Wolsey 2002) also heuristic methods (Dogramaci, Panayiotopulos et al. 1982), are applied. These heuristics are; period by period heuristics, improvement heuristics, relaxation heuristics, branch and bound heuristics, set portioning and column generation heuristics (Karimi, Ghomi et al. 2003). The ways to obtain good formulations via extensions of basic lot sizing problems is discussed by Belvaux and Wolsey (2002). To solve large scale lot-sizing problems, some other heuristics methods have been applied by researchers (Staggemeier and Clark 2001; Stadtler 2003).

The problem we implemented the SBPHA is a multi-period stochastic lot sizing problem with demand uncertainty and backlogging. The problem is formulated in a scenario based formulation.

3.2.1. **Illustrative Example**

The lot sizing problem instance is uncapacitated and has four stages. The decision variables are: binary production decision (whether to produce or not at the first stage), amount to be produced, carrying over inventory and backlogged demand. The objective is to minimize total setup, production, inventory and shortage costs.

Table 9 illustrates the demand realizations where the first stage has only one possible demand realization and other stages have six possible demand realizations each. Total number of scenarios is $1\text{x}6\text{x}6\text{x}6 = 216$.

Table 9: Possible Demand for Lot-sizing Problem

	Demand			
	Stage 1		Stages 2-3-4	
Realization	Demand	Probability	Demand	Probability
1	300	1	500	0.15
2			400	0.2
3			300	0.2
4			200	0.2
5			100	0.15
6			0	0.1

Problem and SBPHA algorithm parameters are shown in Table 10.

Table 10: Lot-sizing Problem and SBPHA Algorithm Parameters

Problem Parameters	Value	SBPHA Parameters		Value
Initial pozitif Inventory:	0	Rho(ρ)	Initial	0
Initial Negative Inventory:	0		Start	10
Set Up Cost:	$ 400.0	Beta(β)	Update	1.2
Production Cost:	$ 3.0	Alpha (α)	Initial	1
Inventory Carrying Cost:	$ 0.6		Start	0.8
Shortage Cost:	$ 3.5		Update	0

The production, inventory carrying and shortage costs are per unit values.

3.2.2. Results and Comparison of SAA and SBPHA

The multi-stage lot-sizing problem is solved with SAA and SBPHA, then results are compared with optimal solution and objective function value. The samples are randomly selected and are identical for both the SAA and the SBPHA. We used the conditional sampling to obtain samples (Ruszczynski and Shapiro (2003); Shapiro (2003).

Results in Table 11 are first stage solutions and objective function values of multi-stage stochastic lot-sizing problem for SAA, SBPHA and exact method. Last column in the table presents gap of the objective function values from the optimal (exact) objective function value.

Table 11: SAA, SBPHA and Exact Solutions, Comparisons for Multi-stage Stochastic Lot-sizing Problem

Method	Solution				Objective	Gap
	Prodction Decision	Production amount	Inventory	Shortage		
SAA	1	600.0	300.0	0.0	4547.673	0.2%
SBPHA	1	700.3	400.3	0.0	4539.999	0.0%
Exact	1	700.0	400.0	0.0	4539.903	-

Note that solution of the SAA shows the solution that performed the best in the full scenario set. As seen in Table 11, both SAA and SBPHA decide to produce at first stage as in the exact solution. SAA decides on producing 600 units, while SBPHA produces 700.3 units close to the optimal solution of 700 units. Since there is no any backlogged demand to satisfy from previous stage and amount to be produced is higher than demand ($= 300$), shortage is $= 0$ in all three methods

The gap for objective function values are calculated via: $\frac{v^{SAA} - v^*}{v^*}$ for SAA and $\frac{v^{SBPHA} - v^*}{v^*}$ for SBPHA. According to gaps, SAA shows a good performance in terms of objective function value, and only 0.2% far from optimal objective function value. However, SBPHA improves the solution of SAA and converges to optimal solution (with 0.0% gap). Even SAA shows a good performance in terms of objective function value, does not show the same performance in decision variables. SAA decides to produce 100 units less than optimal which cause a 14.3% gap from the optimal solution.

CHAPTER IV: FLEXIBILITY to MITIGATE SUPPLY CHAIN DISRUPTIONS

4.1. Introduction and Related Literature

In this chapter, we apply the proposed SBPHA to solve a strategic supply chain problem which has been becoming more important with the increased disruption risks affecting today's global supply chains. The goal of this chapter is to analytically study the plant-product flexibility strategies for stylized problems and then perform an experimental study using SBPHA methodology to investigate the effect of different flexibility strategies.

Facility location and flexibility decisions (i.e., process, volume, product-mix flexibility etc.) are strategic supply chain decisions and require considerable investment affecting the firms' long term profitability. An important category of such uncertain events is the disruption of facilities which are critical for the ability to efficiently serve the customer demand (Schütz, Tomasgard et al. 2009). These disruptions can be natural disasters or man-made (such as terrorist attacks, labor strikes, etc.). Sheffi (2005) classifies disruption causes as direct (disruptions in supplies, problems in facility, and breakdowns in information system) and indirect reasons that can cause disruptions in supply chain. In numerous cases, the disruption at a region may extend or migrate through the network and affect other parts of the supply chain network (Masihtehrani 2011). Recent examples of such disruptions are the 2011 earthquake in Japan affecting Toyota's ability to ship parts and finished vehicles (Brennan 2011; TheGuardian 2011), hurricanes Katrina and

Rita in 2005 disrupting the nation’s oil refineries, and the 2011 fire at the Philips plant halting the production of Ericsson and Nokia (Snyder, Scaparra et al. 2006).

In practice, capacity decisions are considered jointly with the location decisions. Further, the capacity of facilities often cannot be changed in the event of a disruption. Following a facility failure, customers can be assigned to other facilities only if these facilities have sufficient available capacity and are designed (i.e. process flexibility) to be able to produce the products. Flexibility permits management to shift production among different facilities and suppliers to be able to cope with internal and external fluctuations (Chen, Egbelu et al. 1994). Thus, adding flexibility to the facilities becomes another strategy in managing disruptions (Elabassi, Benjaafar et al. 2010).

In order to develop a robust supply chain, managers need to consider not only which candidate facilities to open and their capacity levels but also the flexibility level of the facilities. Flexibility and the advantages can be obtained from flexibility have received significant attention from both the researchers and practitioners for decades. As Chou, Chua et al. (2010) state, the consumer markets show a switch toward more customized products and faster renovations in technology. This enforces companies to design more flexible facilities that are capable of producing variety of products in a facility to satisfy customer demand in case of any change in market or disruption in any of the manufacturing facilities or supplies.

Iravani, Oyen et al. (2005) motivating from “structural flexibility” concept, they proposed new flexibility measures and the applicability of the new methodology to manufacturing and service systems. In a recent study Chod, Rudi et al. (2009)

focused on three types of flexibility, “mix”, “volume” and “time” flexibility, to respond changes in market conditions. Bish, Muriel et al. (2005) studied the impact of flexibility on the supply chain. They showed that a certain flexibility level (partial flexibility) might seem reasonable but may end up in higher costs in terms of higher inventory levels what increases operational costs and decreases profitability. Muriel, Somasundaram et al. (2006) applied this concept to larger systems. In another research, Aksin and Karaesmen (2007) analyzed a network flow model to show interaction between flexibility and capacity and they analytically demonstrated that the expected throughput is concave in the degree of flexibility.

One of the most common flexibility types that are studied by researchers and the practitioners is process flexibility. The process flexibility concerns the number of product types that can be manufactured in each production facility (Sethi and Sethi 1990; Jordan and Graves 1995; Garavelli 2003; Chou, Chua et al. 2010). As Garavelli (2003) pointed, the costs associated to the facility capability of producing different types of products increase with the number of products. In most cases due to high flexibility investments producing all types of products in all facilities are not economic, so called “total flexibility” in Jordan and Graves (1995) and Garavelli (2003) and “full flexibility” in Chou, Chua et al. (2010) and Elabassi, Benjaafar et al. (2010). On the other hand, designing each facility to be capable of producing a few types of products is not efficient as well, because of higher transportation costs (Garavelli 2003), called “limited flexibility” in Jordan and Graves (1995) and Garavelli (2003).

Jordan and Graves (1995) introduced “chaining” concept as an intermediate level of process flexibility, where company gains most of the advantages of full flexibility only having limited flexibility, by creating the longest close loop between facilities and products so that all facilities and products are connected directly or indirectly. They called this form “1-chain” flexibility and showed that “1-chain” flexibility provides almost all the benefits of full flexibility when concern lost sales. Also Graves (2008) showed that when chaining form is set up right, provides the same benefits as full flexibility provides. Jordan and Graves (1995) compared the “1-chain” flexibility and full flexibility performances under uncertain demand environment. They also show some results on different flexibility level performance for different capacity levels. Garavelli (2003) proposed a simulation model to evaluate the performances of no flexibility, process flexibility and full flexibility levels in supply chain.

Chou, Chua et al. (2010) analytically studied the flexibility levels (process flexibility, full flexibility) for both symmetrical and asymmetrical demand (i.e., demand is not symmetrical around its mean) cases. Their model also can be applied to unbalanced cases (i.e., capacity is not equal to mean demand). To the best of our knowledge, the only work that considers facility disruptions when comparing flexibility level performances is Elabassi, Benjaafar et al. (2010). However, Elabassi, Benjaafar et al. (2010) focused only on balanced (i.e., capacity is equal or little higher than total demand) cases. In this study, we present different flexibility levels (i.e., no flexibility, process flexibility with chaining and without chaining, full flexibility) performances for both balanced and unbalanced capacity-demand cases. Our work is not limited to numerical studies; we also study analytically both balanced and unbalanced cases.

4.2. Problem Formulation

In this section, we first present Capacitated Reliable Facility Location and Product Allocation Problem (CRFLPAP) with assignment decisions. Next, we show flexibility configurations in location-allocation problems.

4.2.1. Capacitated Reliable Facility Location and Product Allocation Problem

We now introduce the notation used throughout this chapter. Let F denote the set of possible facility sites including the emergency facility and P denote the set of products (i.e., type of products). Let f_j be the fixed cost for facility $j \in \mathrm{F}$, which is incurred if the facility is opened, and p_i be the product $i \in P$. The v_{ij} denotes the cost of designing the facility j in order to be capable of producing product i. The c_{ij} denotes costs of satisfying each demand unit of product i from facility j and includes such variable cost drivers as transportation, production, and inventory. Whenever a demand unit of product i cannot be served by any of the opened facilities that can produce product i, then the demand for product i is assigned to the emergency facility and a large penalty cost h_i is incurred for each unit of unsatisfied demand. This penalty can be incurred due to finding an alternative source or due to the lost sale. Furthermore, the facility j has limited capacity and can serve at most b_j units of demand, and the total demand for product i is d_i. The u_{ij} denotes the usage rate from capacity of facility j for each demand unit of product i.

We formulate the CRFLPAP as a two-stage stochastic programming problem. In the first stage, the location and design decisions are made before random failures of the located facilities. In the second stage, following the facility failures, the product-facility assignment decisions are made for every product given the facilities that have not failed. The goal is to identify the set of facilities to be opened while minimizing the total cost of open facilities and the expected cost of meeting demand of products from the lasting facilities and the emergency facility. In the scenario based formulation of CRFLPAP, let s denote a failure scenario and the set of all failure scenarios is S, where $s \in S$. Let p_s be the probability that scenario s occurs and $\sum_{s \in S} p_s = 1$. Further let k_j^s denote whether the facility j survives (i.e., $k_j^s = 1$) and $k_j^s = 0$ otherwise. For instance, in case of independent facility failures, we have $|S| = 2^{|F|-1}$ possible failure scenarios for $|F| - 1$ facilities and the last facility is the emergency facility which is perfectly reliable. Note that our proposed method does not require any assumption on independence and distribution of each facility's failure.

The binary decision variable z_j specifies whether facility j is opened or not, and the binary variable y_{ij} specifies whether facility j designed to be capable of producing product i. Integer variable x_{ij}^s denotes the number of product i assigned to facility j in scenario s. Lastly, the scenario based formulation of the CRFLPAP (with assignment decisions) as a two-stage stochastic program is as follows.

$$\min \sum_{j \in F} f_j z_j + \sum_{i \in P} \sum_{j \in F} v_{ij} y_{ij} + \sum_{s \in S} p_s \sum_{i \in P} \sum_{j \in F} c_{ij} x_{ij}^s \tag{4.1}$$

subject to

$$\sum_{j \in F} x_{ij}^s = d_i, \forall i \in P, s \in S \tag{4.2}$$

$$y_{ij} \leq \mathrm{z}_j, \qquad \forall j \in F, i \in P \tag{4.3}$$

$$\sum_{j \in F} z_j \leq \mathrm{n}_{\max}, \qquad \forall j \in F \tag{4.4}$$

$$x_{ij}^s \leq \frac{b_j}{u_{ij}} y_{ij} \ , \qquad \forall i \in P, j \in F, s \in S \tag{4.5}$$

$$\sum_{i \in P} u_{ij}\, x_{ij}^s \leq k_j^s b_j z_j, \forall j \in \mathrm{F}, \mathrm{s} \in \mathrm{S} \tag{4.6}$$

$$x_{ij}^{\mathrm{s}} \geq 0, \forall i \in P, j \in F, s \in S \tag{4.7}$$

$$z_j \in \{0{,}1\}, \forall j \in F \tag{4.8}$$

$$y_{ij} \in \{0{,}1\}, \forall i \in P, j \in F \tag{4.9}$$

The objective function in formulation (4.1) minimizes the total fixed cost of opening facilities, designing facilities to produce different types of products and the expected second stage cost of satisfying demand through lasting and emergency facility. Constraints (4.2) ensure that demand for each product is satisfied. Constraints (4.3) prevent designing a facility to produce any product if it is not an opened facility. Constraints (4.4) assure the total number of opened facilities is not exceeding the number of facilities that applicant wants to open($\mathrm{n}_{\max}$). Constraints (4.5) prevent the assignment of any demand to a facility if it is not designed to produce that product and ensure that assigned amount does not exceed its capacity in every failure scenario. Constraints (4.6) prevent assignments to a failed or not opened facility and prevent total assignment does not exceed its capacity if it is survived and opened facility. Constraints (4.7), (4.8) and (4.9) are integrality constraints.

4.3. Analytical Analysis

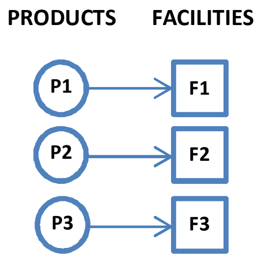
4.3.1. Flexibility Configurations

The level of flexibility and the allocation strategy directly affect the rate of benefit that can be gained from process flexibility. In case of facility disruptions, product assignment decisions (can be at plant level or line level), the level of process flexibility (number of links between a product and facilities), play important role to mitigate supply chain disruptions.

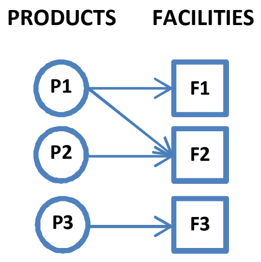
In order to make the right decision on where to add flexibility, the concept of “chaining” is very important (Jordan and Graves 1995). A "chain" can be explained as: a group of products and plants which are all connected. This connection can be either directly or indirectly, by product assignment decisions (Elabassi, Benjaafar et al. 2010). Within a chain, all products or plants are connected via links between them. All products are produced only by the plants that are members of the chain and these plants only can produce the products that are members of the chain.

4.3.1.1 No Disruption Case

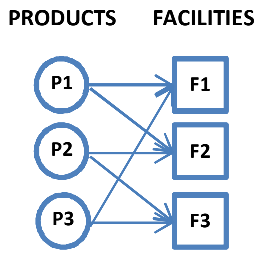
In order to demonstrate and compare the value of flexibility levels we consider the assignment of m-products to n–facilities (where $n = m$). The bipartite graph representation of flexibility structure is shown in Figure 16 for $n = m = 3$. Flexibility levels will be expressed by total number of links between products and facilities. A link between product i and facility j shows that the facility j is designed to be capable of producing product i, e.g., $y_{ij} = 1$.



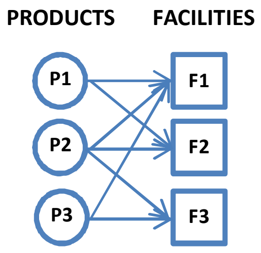
a) Dedicated with n links



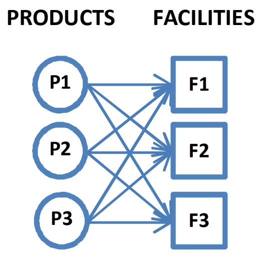
b) Low flexibility with 2n-t links, $1 \leq t \leq n+1$



c) 1-chain flexibility with 2n links



d) High flexibility with 2n+t links, $1 \leq t \leq n^2 - 2n - 1$



e) Full flexibility n^2 links

Figure 16: Different Flexibility Levels for $m = 3$ Products and $n = 3$ Facilities

In Figure 16, a) dedicated case; each facility is capable of producing only one product, and each product is assigned to a different facility. In b), one product is produced in more than one facility while others are produced in only one facility. This configuration is considered as low flexibility level. The configuration in c) one chain; each facility is able to produce two products and a long single chain created to be able to switch capacity from lower demand products to higher demand products (Jordan and Graves 1995). In e) full flexibility; all facilities are designed to be capable of produce all types of products.

4.3.1.2 Disruption Case

Let $\hat{n}$ represent the total number of products that can be satisfied if there were sufficient capacity; in the example (a) below, the $\hat{n} = 3$ since all three products can be served from $\mathrm{F2}$ and $\mathrm{F3}$. In contrast, in example (b) below, the product 1 cannot be served (due to elimination of all assignment links) hence only $\mathrm{P2}$ and $\mathrm{P3}$ can be served, e.g., $\hat{n} = 2$.

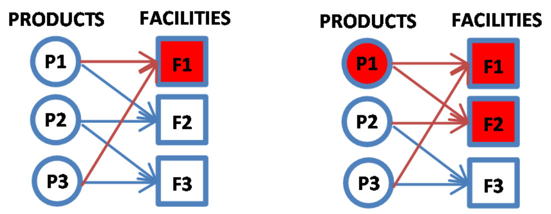


Figure 17: Examples of Failure Configurations in 1-chain flexibility for $n = m = 3$

If number of failed facilities (r) is less than number of total facilities (n), $1 \leq r < n$,:

i. In general: $1 \leq \hat{n} \leq n$

ii. 1-chain: $n - r + 1 \leq \hat{n} \leq n$,

iii. Full Flexibility: $\hat{n} = n$

iv. Dedicated: $\hat{\mathrm{n}} = \mathrm{n} - \mathrm{r}$

We next compare the performance of different flexibility levels to the full flexibility level performance both analytically and numerically under different capacity levels of facilities. All facilities are subject to fail with a given failure probability. In the next subsection, we compare “1-chain” flexibility with full flexibility analytically both for balanced and unbalanced demand-capacity cases.

4.3.2. Analytical Analysis – Homogenous Products and Plants

In this section, we first demonstrate comparison of 1-chain flexibility and full flexibility for balanced demand-capacity case when capacity of a facility is equal or higher than demand of a product up to a certain point. Then we compare these two levels of flexibility for unbalanced case when capacity of a facility is much higher than the demand of a product.

We consider “symmetric” cases where products and facilities are identical as discussed in Elabassi, Benjaafar et al. (2010) for facility disruption case and Chou, Chua et al. (2010) for uncertain demand case.

Assumptions:

Demand satisfying cost (service): $c_{\text{ij}} = \text{c} \;\; \forall j \in F, i \in P$.

Design cost: $v_{\text{ij}} = v \;\; \forall j \in F, i \in P$.

Capacity: $b_j = b \;\; \forall j \in F$.

Demand: $d_i = d \;\; \forall \; i \in P$.

Penalty/Outsourcing cost: $h_i = h \;\; \forall \; i \in P$.

Failure probability for facilities: $q_j = q \;\; \forall j \in F$.

Number of products, facilities: $m = n > 2$.

Capacity, demand relation: $b > d$.

4.3.2.1 Demand-Capacity Balanced Case

Under the assumption that capacity, b, is always greater than the demand, d, and they are balanced, then we can easily express following equation: $nd \geq (n-1)b$.

Proposition 1: 1-chain flexibility always satisfy as the same amount of demand as full flexibility and is superior to full flexibility in terms of total cost (TC) because of high designing cost in full the flexibility case.

Proof: Total Cost (TC) function of CRFLPAP as shown above in (4.1):

$$\text{Minimize} \qquad \sum_{j\in F} f_j\, z_j + \sum_{i\in P}\sum_{j\in F} v_{ij} y_{ij} + \sum_{s\in S} p_s \sum_{i\in P}\sum_{j\in F} c_{ij} x_{ij}^{s}$$

There are three terms in the cost function. First term is total open cost, second term is total designing cost, and the third term is the expected total demand satisfying cost. Total open cost is equal for both 1-chain and full flexibility since we open all facilities in both cases.

$$C_{open}^{1-chain} = C_{open}^{full} = \sum_{j \in F} f_j = nf \tag{4.10}$$

In 1-chain flexibility case each facility is capable of producing 2 products, in other word each product is assigned to two facilities. Since there are $m\ (= n)$ products, total designing cost for 1-chain flexibility is:

$$C_{design}^{1-chain} = \sum_{i \in P} \sum_{j \in F} v_{ij} y_{ij} = 2nv \tag{4.11}$$

In full flexibility case, all facilities are deigned to produce all products, then total designing cost for full flexibility case is:

$$C_{design}^{full} = \sum_{i \in P} \sum_{j \in F} v_{ij} y_{ij} = n^2 v \tag{4.12}$$

Let us now calculate demand satisfying service cost for 1-chain flexibility case. Let r facilities are failed in the system, then available capacity is $(n - r)b$, and number of products that can be served is $(n - r + 1)$, then total demand that can be served is $(n - r + 1)d$.

In any failure scenario, $\min(nd + d(1 - r), nb - rb)$ of demand can be satisfied (see Proposition 2). As it is proved above $nd + d(1 - r) \geq nb - rb$, total satisfied demand is $(n - r)b$, and unsatisfied demand is $nd - (n - r)b$.

Since facility failures are independent, the occurrence probability of a scenario with r failures is $q^r(1-q)^{n-r}$. Demand satisfying service cost can be expressed as follows:

$$C_{service}^{1-chain} = ndc(1-q)^n + \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \left(nd - (n-r)b\right)h$$

$$+ \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} (n-r)bc \tag{4.14}$$

In full flexibility case, all facilities can serve all products. Total satisfied demand, under r failed facilities scenario, is determined via $\min(nd, (n-r)b)$. Because of the assumption in balanced case: $nd \geq (n-1)b$ and $r \geq 1$, total satisfied demand is $(n-r)b$ and unsatisfied demand is $nd-(n-r)b$. Then total demand satisfying service cost for full flexibility can be expressed as:

$$C_{service}^{full} = ndc(1-q)^n + \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \left(nd - (n-r)b\right)h$$

$$+ \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} (n-r)bc \tag{4.15}$$

Let us compare 1-chain and full flexibilities performances in terms of Total Cost (TC) by subtracting TCs.

$$TC^{full} - TC^{1-chain} = C^{full}_{open} + C^{full}_{design} + C^{full}_{service} - C^{1-chain}_{open} - C^{1-chain}_{design} - C^{1-chain}_{service}$$

$$= nf + n^2v + \quad ndc(1-q)^n + \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \, (nd - (n-r)b)h$$

$$+ \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \, (n-r)bc - nf - 2nv$$

$$- \left(ndc(1-q)^n + \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \, (nd - (n-r)b)h \right.$$

$$\left. + \sum_{r=1}^{n} \binom{n}{r} q^r (1-q)^{n-r} \, (n-r)bc \right)$$

$$= \; n^2v - 2nv \tag{4.16}$$

<u>If $v = 0$</u> →$TC^{full} = TC^{1-chain}$

<u>If $v > 0$</u> →$TC^{full} > TC^{1-chain}$, since $n > 2$. □

Proposition 2: Because of the balanced assumption ($\boldsymbol{nd \geq (n-1)b}$ and, $\boldsymbol{d < b}$, then it can be proved that total available capacity is less than total demand:

$$(n - r + 1)d > (n - r)b \tag{4.13}$$

Proof: Given that $nd \geq (n-1)b$ and, $d < b$,

$$nd + b \geq nb$$

When rb is subtracted from both sides of equation:

$$nd + b - rb \geq nb - rb \text{ or } nd + (1-r)b \geq nb - rb$$

Since $r \geq 1$, and $(1-r)b \leq (1-r)d$ (both sides of equation is “0” or negative)

When $(1-r)b$ is replaced by $(1-r)d$, then we conclude that $nd + (1-r)d \geq nb - rb$. □

Please note that proofs above are modified from Elabassi, Benjaafar et al. (2010). We want to use proofs above for the clarity of the concept and the proof for unbalanced case.

4.3.2.2 Demand-Capacity Unbalanced Case

In this subsection, we compare 1-chain and full flexibility level performances in terms of TC for unbalanced demand and capacity case.

Proposition-3: The assumption ($\boldsymbol{nd} \geq (\boldsymbol{n}-\mathbf{1})\boldsymbol{b}$) is made in balanced case, but this assumption is not valid for the unbalanced case. Let us reverse the assumption that is made in balanced case as: $\boldsymbol{nd} \leq (\boldsymbol{n}-\mathbf{1})\boldsymbol{b}$. Then it can be argued that $\boldsymbol{nd} + \boldsymbol{d}(\mathbf{1}-\boldsymbol{r}) \geq \boldsymbol{nb} - \boldsymbol{rb}$ is not always true.

Proof: Given that $nd \leq (n-1)b$ and, $d < b$

$$nd + b \leq nb$$

If rb is subtracted from both sides of the equation

$$nd + b - rb \leq nb - rb$$

Since $r \geq 1$, and $(1-r)b \leq (1-r)d$ (both sides of equation is "0" or negative)

When $b - rb$ is replaced by $d - rd,$ then it can be concluded that $nd + d(1-r) \gtreqless nb - rb.$ □

The proof above shows that the total satisfied demand varies and is dependent on the capacity of facilities and the demand for products. Thus we cannot conclude 1-chain or full flexibilities are superior to one another.

In order to compare total satisfied demand and Total Cost (TC) for 1-chain and full flexibilities, we introduce ***block failure concept***. Let us consider a system for 1-

chain flexibility with 6 facilities and products. Some failure scenarios are shown in Figure 18.

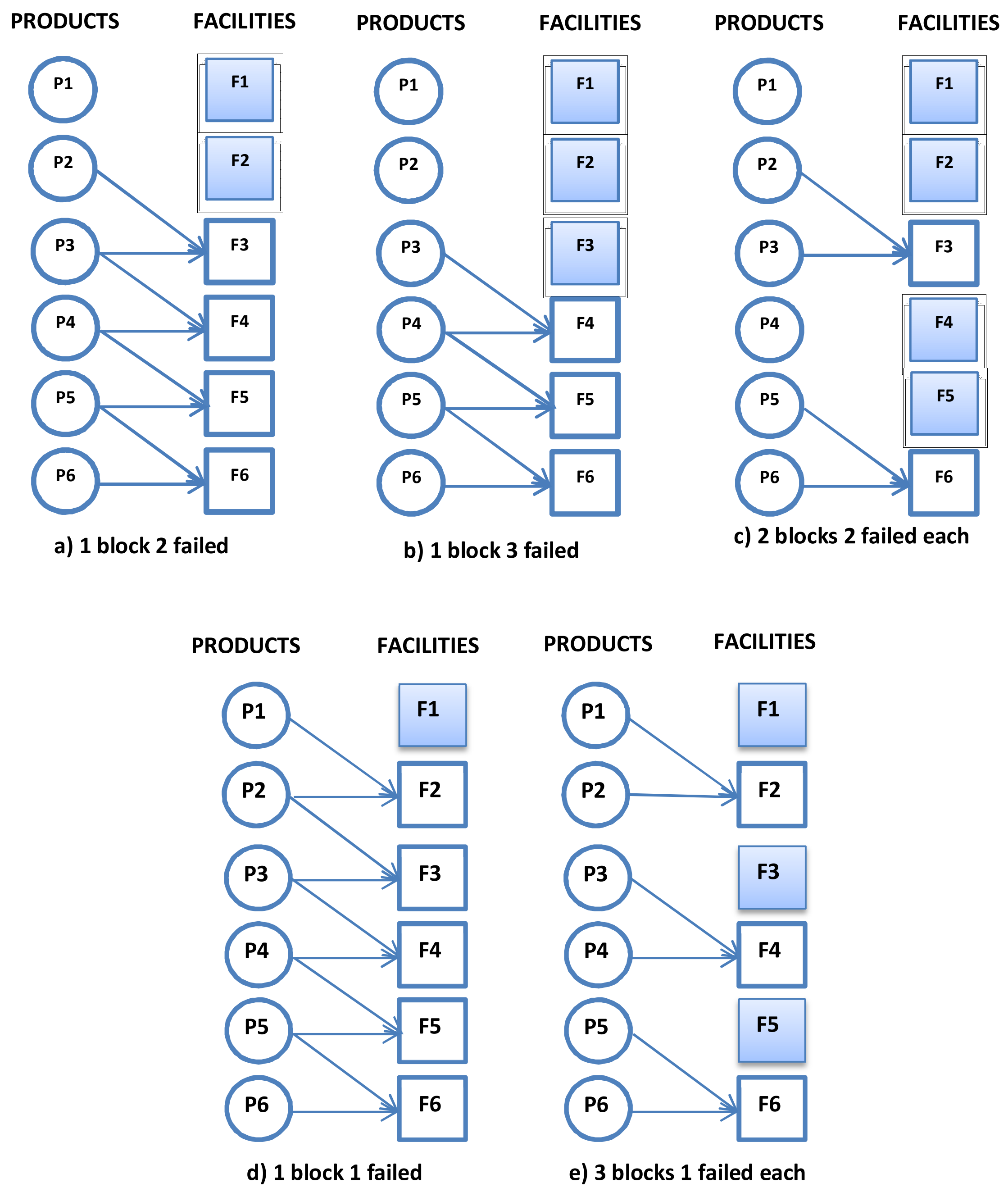


Figure 18: Examples of Failure Configurations for 1-Chain Flexibility

In Figure 18, a) $B_{l,r} = 1$, and $con_a \rightarrow con_1 = 2$, in b) $B_{l,r} = 1$, and $con_a \rightarrow con_1 = 3$, in c) $B_{l,r} = 2$, and $con_a \rightarrow con_1 = 2, con_2 = 2$, in d) $B_{l,r} = 1$, and $con_a \rightarrow con_1 = 1$ and in d) $B_{l,r} = 3$, and $con_a \rightarrow con_1 = 1, con_2 = 1, con_3 = 1$.

Notation and description that is used in this subsection is as follows:

Block: A set of chained facilities that failed. Single failed facility can also be a block. Any two facilities are chained if production capacity in one facility can be used to increase the available capacity in another facility by shifting production levels of several products. Jordan and Graves (1995) have a similar description. Also, any two facilities are chained if they share at least one product. If facility pairs $(\mathrm{F1}, \mathrm{F2})$ and $(\mathrm{F2}, \mathrm{F3})$ are chained, then facility pair $(\mathrm{F1}, \mathrm{F3})$ are also chained.

Notation:

Sc: set of failure scenarios

Sc_r: set of failure scenarios with r failed facilities, i.e., $Sc = \cup_r Sc_r$

l: index for failure configuration

$B_{l,r}$: number of blocks in l^{th} failure scenario of r failures

con_a^{lr}: number of consecutive failed facilities in a^{th} block of l^{th} failure scenario of r failures

Total number of failures:

$$r = \sum_{a=1}^{B_{l,r}} con_a^{lr} \tag{4.17}$$

1-chain flexibility

$(con_a^{lr} - 1)$: number of products that cannot be served due to failures in block a

No flexibility

(con_a^{lr}): number of products that cannot be served due to facility failures in block a

Full flexibility

$(con_a^{lr} - con_a^{lr})$: number of products that cannot be served due to facility failures in block a

For instance, $\sum_{a=1}^{B_{l,r}}(con_a^{lr} - 1)d$: total demand that cannot be served due to facility failures in all blocks of failure scenario l of r facility failures in 1-chain flexibility case.

$S_{l,r}^{full}$: Total satisfied demand for full flexibility level when r facilities are failed as configuration l

$S_{l,r}^{1-chain}$: Total satisfied demand for 1-chain flexibility level when r facilities are failed as configuration l

Total open cost and designing cost in unbalanced case will be identical to one in balanced case for both 1-chain and full flexibility levels as shown in (4.10)-(4.12). Since all facilities are opened $(z_j = 1, j = 1, 2, \dots, F)$.

$$C_{open}^{1-chain} = C_{open}^{full} = \sum_{j \in F} f_j = nf$$

$$C_{design}^{1-chain} = \sum_{i \in P} \sum_{j \in F} v_{ij} y_{ij} = 2nv$$

$$C_{design}^{full} = \sum_{i \in P} \sum_{j \in F} v_{ij} y_{ij} = n^2 v$$

Given a failure scenario, where r out of n facilities are failed; the total demand that can be served as a percentage of the total demand

$$\rho(n,r) \leq \frac{\min\{\hat{n}d, (n-r)b\}}{nd} \tag{4.18}$$

Recall that $\hat{n}$ represents the total number of products that can be satisfied if there were sufficient capacity.

$$\hat{n} = n - \sum_{a=1}^{B_{l,r}} (con_a^{lr} - 1) \tag{4.19}$$

Total satisfied demand when r facilities are failed in 1-chain flexibility is

$$S_{l,r}^{1-chain} \leq min\left\{(n-r)b, \left(n - \sum_{a=1}^{B_{l,r}} (con_a^{lr} - 1)\right) d\right\} \tag{4.20}$$

, and total satisfied demand when r facilities are failed in full flexibility is

$$S_{rl,r}^{full} = min\{(n-r)b, nd\} \tag{4.21}$$

Since $nd \geq nd - \sum_{a=1}^{B_{l,r}} (con_a^{lr} - 1)d$, it can be concluded that total satisfied demand in full flexibility is more than 1-chain flexibility in some failure scenarios, what may cause 1-chain flexibility be more costly than full flexibility:

$$S_{l,r}^{1-chain} \leq S_{l,r}^{full} \tag{4.22}$$

Let us now calculate demand satisfying cost for 1-chain and full flexibility levels.

$$\begin{aligned} C_{service}^{1-chain} \quad &= ndc(1-q)^n + c\sum_{r=1}^{n} q^r (1-q)^{n-r} \sum_{l=1}^{\binom{n}{r}} S_{l,r}^{1-chain} \\ &+ h\sum_{r=1}^{n} q^r (1-q)^{n-r} \sum_{l=1}^{\binom{n}{r}} (nd - S_{l,r}^{1-chain}) \end{aligned} \tag{4.23}$$

$$C_{service}^{full} = ndc(1-q)^n + c\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}} S_{l,r}^{full}$$

$$+ h\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(nd - S_{l,r}^{full}\right) \tag{4.24}$$

Let us now compare 1-chain and full flexibilities performances in terms of Total Cost (TC).

$$TC^{full} - TC^{1-chain} = C_{open}^{full} + C_{design}^{full} + C_{service}^{full} - C_{open}^{1-chain} - C_{design}^{1-chain} - C_{service}^{1-chain} \tag{4.25}$$

$$= nf + n^2v + ndc(1-q)^n + c\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}} S_{l,r}^{full}$$

$$+ h\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(nd - S_{l,r}^{full}\right) - nf - 2nv - ndc(1-q)^n$$

$$- c\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}} S_{l,r}^{1-chain} - h\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(nd - S_{l,r}^{1-chain}\right)$$

$$= n^2v - 2nv + c\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(S_{l,r}^{full} - S_{l,r}^{1-chain}\right)$$

$$- h\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(S_{l,r}^{full} - S_{l,r}^{1-chain}\right)$$

$$= \underbrace{n^2v - 2nv}_{\text{Term1}} + \underbrace{(c-h)}_{\text{Term2}}\left(\underbrace{\sum_{r=1}^{n} q^r(1-q)^{n-r}\sum_{l=1}^{\binom{n}{r}}\left(S_{l,r}^{full} - S_{l,r}^{1-chain}\right)}_{\text{Term3}}\right)$$

If $v = 0$

$Term1 = 0$,

Since $h > c$,$Term2 < 0$,

Since

$$\sum_{l=1}^{\binom{n}{r}} S_{l,r}^{full} \geq \sum_{l=1}^{\binom{n}{r}} S_{l,r}^{1-chain}$$

, $Term3 \geq 0$

Then, it can be easily concluded that $C_{full} - C_{1chain} \leq 0$;

$$TC^{full} \leq TC^{1-chain}$$

If $v > 0$: Possible Outcomes:

i. If $TC^{full} - TC^{1-chain} = 0$, both full flexibility and 1-chain flexibility show the same performance,

ii. If $TC^{full} - TC^{1-chain} > 0$, 1-chain flexibility shows better performance than full flexibility,

iii. If $TC^{full} - TC^{1-chain} < 0$, full flexibility shows better performance than 1-chain flexibility,

We can argue that the performance of full flexibility and 1-chain flexibility depends on value of v, failure probability(q), capacity(b), demand(d), service cost(c), and lost sale cost(h). Put in another way, there will be some situations that full flexibility performs better, while in others 1-chain flexibility performs better.

An example for 3 facilities, and 3 products case:

There will be four situations; no fail, 1, 2, or 3 failed facilities.

No Failure: $r = 0$, $L = 1$. In this situation there will be only one configuration, $(l = 1)$. Writing the equations (4.20) and (4.21):

$S_{1,0}^{full} = \min((n-0)b, nd) = \min(nb, nd)$, since $\mathrm{b} > d, nb > nd, S_{1,0}^{full} = nd$

$S_{1,0}^{1-chain} = \min((n-0)b, nd) = \min(nb, nd)$, since $\mathrm{b} > d, nb > nd, S_{1,0}^{1-chain} = nd$

1 Failure: $r = 1$, $L = 3$. In this situation there will be three configurations, $(l = 3)$.

a) Facility 1 fails, 2 and 3 survive

b) Facility 2 fails, 1 and 3 survive

c) Facility 3 fails, 1 and 2 survive

$S_{1,1}^{full} = S_{2,1}^{full} = S_{3,1}^{full} = \min((n-1)b, nd) = \min(2b, 3d)$

$S_{1,1}^{1-chain} = S_{2,1}^{1-chain} = S_{3,1}^{1-chain} \leq \min((n-1)b, nd - 0) = \min(2b, 3d)$

Since there is no failures other than one block with 1 failed facility,

$S_{1,1}^{1-chain} = S_{2,1}^{1-chain} = S_{3,1}^{1-chain} = \min((n-1)b, nd - 0) = \min(2b, 3d)$

2 Failure: $r = 2$, $L = 3$:In this situation there will be three configurations, $(l = 3)$.

a) Facility 1 and 2 fails, 3 survives

b) Facility 1 and 3 fails, 2 survives

c) Facility 2 and 3 fails, 1 survives

$S_{1,2}^{full} = S_{2,2}^{full} = S_{3,2}^{full} = \min((n-2)b, nd) = \min(b, 3d)$

$S_{1,2}^{1-chain} = S_{2,2}^{1-chain} = S_{3,2}^{1-chain} \leq \min((n-2)b, nd - (2-1)d) = \min(b, 2d)$

Since there is no failures other than one block (with 2 failed facilities),

$S_{1,2}^{1-chain} = S_{2,2}^{1-chain} = S_{3,2}^{1-chain} = \min((n-2)b, nd - (2-1)d) = \min(b, 2d)$

3 Failure: $r = 3$, $L = 1$:In this situation there will be three configuration, $(l = 3)$.

$S_{1,3}^{full} = \min\big((n-3)b, nd\big) = \min(0, 3d) = 0$

$S_{1,3}^{1-chain} = \min\big((n-3)b, nd-(3-1)d\big) \leq \min(0, d) = 0$

Total Cost Difference:

$$TC^{full} - TC^{1-chain} = 3f + 9v + c[(1-q)^3 nd + 3q(1-q)^2 \min(2b, 3d)$$
$$+3q^2(1-q)\min(b, 3d) + q^3 0] + h[\,(1-q)^3 0 + 3q(1-q)^2\,(3\text{d} - \min(2b, 3d))$$
$$+3q^2(1-q)(3d - \min(b, 3d) + q^3 3d] - 3f - 6v - c[(1-q)^3 nd$$
$$+3q(1-q)^2 \min(2b, 3d) + 3q^2(1-q)\min(b, 2d) + q^3 0] - h[\,(1-q)^3 0$$
$$+3q(1-q)^2\,(3\text{d} - \min(2b, 3d)) + 3q^2(1-q)(3d - \min(b, 2d) + q^3 3d]$$
$$= 3v + 3q^2(1-q)(c-h)\,[\min(b, 3d) - \min(b, 2d)]$$

<u>If $b < 2d$</u>

$TC^{full} - TC^{1-chain} = 3v$, 1-chain flexibility is superior to full flexibility

<u>If $2d < b < 3d$</u> , then

$TC^{full} - TC^{1-chain} = 3v + 3q^2(1-q)(c-h)\,(b-2d)$, their performances depend on the value of $v, b, d, c, h,$ and q.

If $v = |q^2(1-q)(c-h)\,(\text{b} - 2d)|$, 1-chain and full flexibility show the same performances

If $v > |q^2(1-q)(c-h)\,(\text{b} - 2d)|$, 1-chain flexibility is superior to full flexibility

If $v < |q^2(1-q)(c-h)\,(\text{b} - 2d)|$, full flexibility is superior to 1chain flexibility

<u>If $3d < b$</u> , then

$C_{full} - C_{1chain} = 3v + 3q^2(1-q)(c-h)(3d-2d)$, their performances depend on the value of $v, b, d, c, h,$ and q.

If $v = |q^2(1-q)(c-h)\ d|$, 1-chain and full flexibility show the same performances

If $v > |q^2(1-q)(c-h)\ d|$, 1-chain flexibility is superior to full flexibility

If $v < |q^2(1-q)(c-h)\ d|$, full flexibility is superior to 1chain flexibility

According to the proofs and analysis above, it can be concluded that performances of 1-chain and full flexibility depend on the parameters of the problem in unbalanced demand-capacity case.

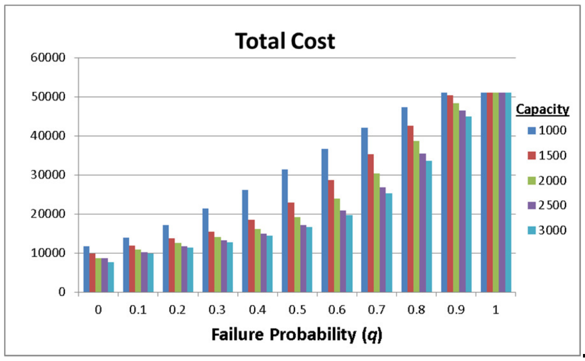
4.4 Experimental Tests

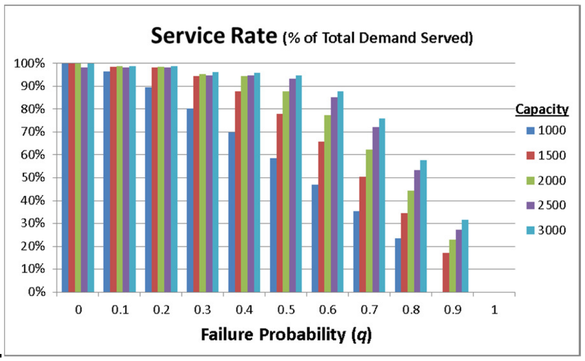
In this section, we have performed an experimental study to understand the impact of various problem parameters on the total cost, service rate (demand satisfaction level), degree of plant-product assignment, and average number of products assigned per plant. All the problem instances are solved using SBPHA algorithm. Note that while SBPHA is not guaranteed to be exact for any given problem, we have extensively tested the degree of optimality in calibrating the SBPHA's parameters.

4.4.1. Experimental setting

In these experiments, we have considered 7 plant locations (including the emergency facility) and 6 customers. In total there are 64 plant failure scenarios. The assignment cost is taken as 100; each customer has 850 units of demand; unit cost is 1; cost of unit unmet demand is 10; cost of operating a plant is 1000. We have varied the remainder of the problem parameters, e.g., plant failure probability, plant capacity, utilization threshold and loss of capacity due to increased set ups with additional product assignments. In all instances, we consider the homogenous plants and customers, e.g. plants all have identical capacity and failure probability.

Figure 15 illustrates the effect failure probability and plant capacity level on the total cost and the service rate when there are no minimum utilization requirements and no loss of capacity due to assigning more than one product. Total cost results indicate that the effect of capacity increase is most apparent when the failure probabilities are between 0.5 and 0.8. At the extreme failure probabilities, the effect of capacity on total cost is not as significant. When the capacity levels are tight (with respect to demand), the total cost increases linearly with the increasing failure rate. However, the total cost increases at an increasing rate when there is excess capacity. The same is also true for the service rate. As in total cost results, the service rate decreases linearly with tighter capacity and nonlinearly with excess capacity configuration.

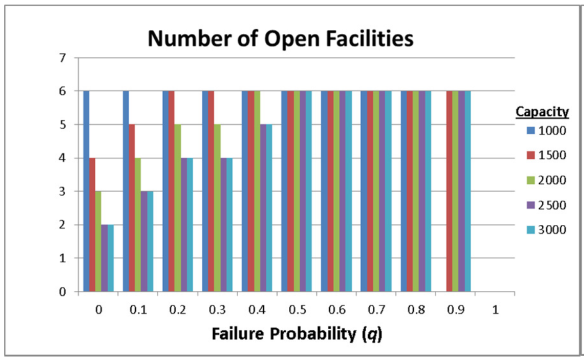


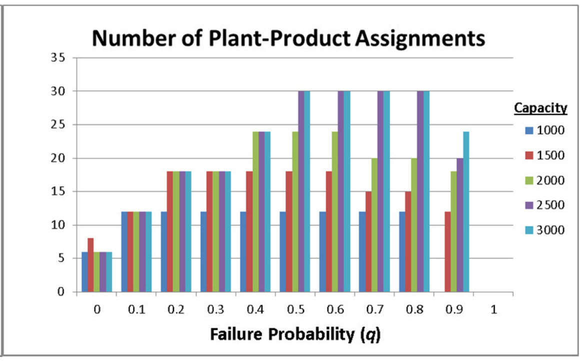


(a) (b)

Figure 19: Total cost and service rate for no utilization requirement and no loss of capacity case.

Figure 16 illustrates the effect on the number of facilities and total number of plant-product assignments. Clearly, increasing failure probability leads to increased number of facilities. This increase is most discernable when there is excess capacity. The total number of plant-product assignments first increases and then decreases with the increasing failure rate.





(a) (b)

Figure 20 Number of open facilities and product-plant assignments for no utilization requirement and no loss of capacity case.

Figure 17 illustrates the effect on the average number of product assignments per plant. Again, this statistic is first increasing and then decreasing with the increasing failure rate. This indicates that there is no single dominating flexibility level (e.g., 1-chain or full-flexibility) that provides the best performance. Further, the optimal flexibility configuration depends on the failure probability which supports the analytical results in the preceding section.

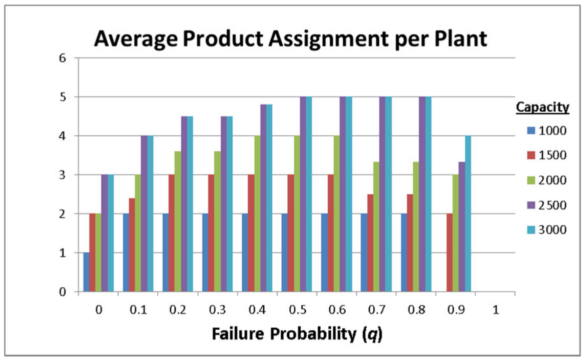
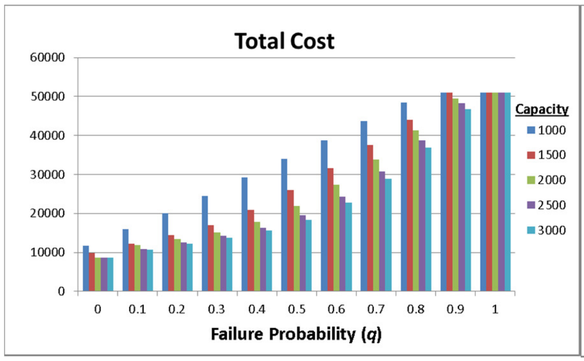
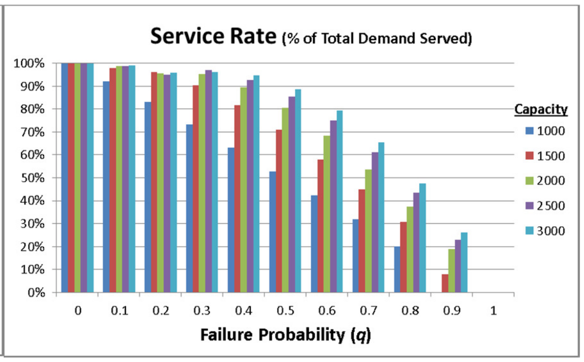


Figure 21: Average number of products plant assignments for no utilization requirement and no loss of capacity case.

The figures 18 to 20 shows the similar observation as above when we incorporate the capacity loss with additional product assignments. One visible effect is the dampening of the capacity effect.

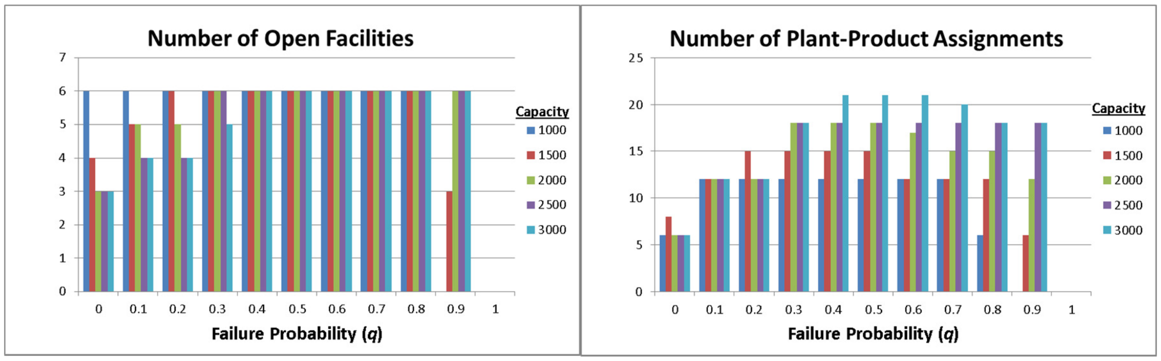




.

(a) (b)

Figure 22: Total cost and service rate for no utilization requirement and 10% loss of capacity per additional product.



(a) (b)

Figure 23: Number of open facilities and product-plant assignments for no utilization requirement and 10% loss of capacity per additional product.

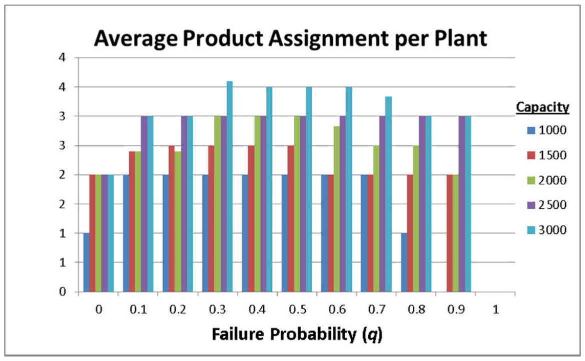
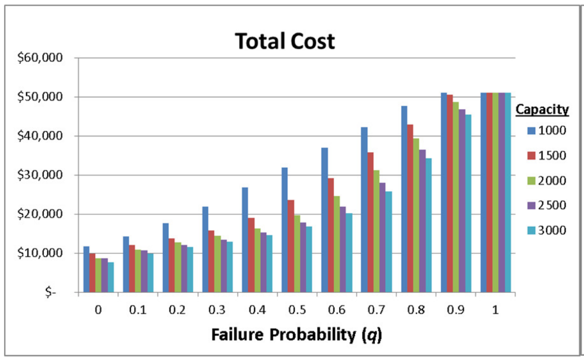
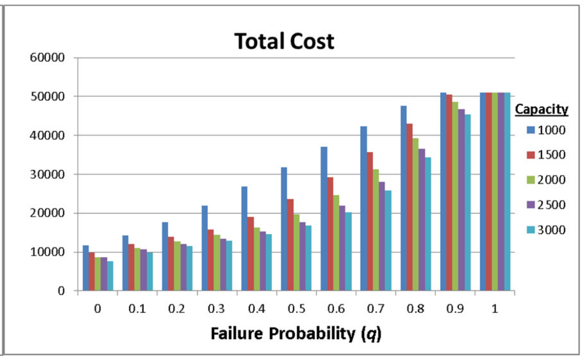


Figure 24: Average number of products plant assignments for no utilization requirement and 10% loss of capacity per additional product.

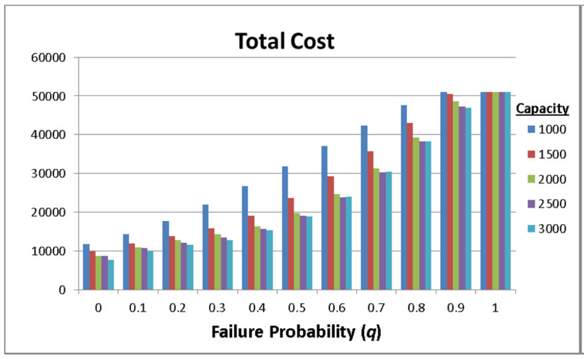
Figure 21 shows the effect of capacity utilization requirement on the total cost. Clearly, total cost differences among different capacity levels decrease with tightening utilization requirements.



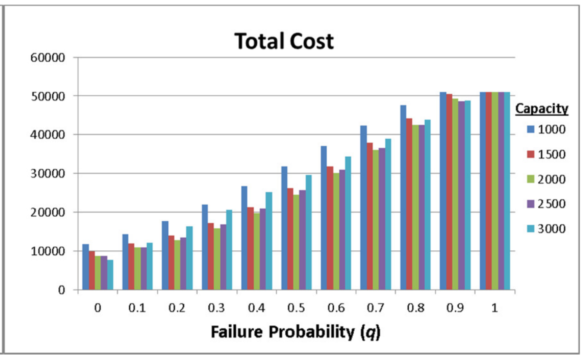
(a)



(b)



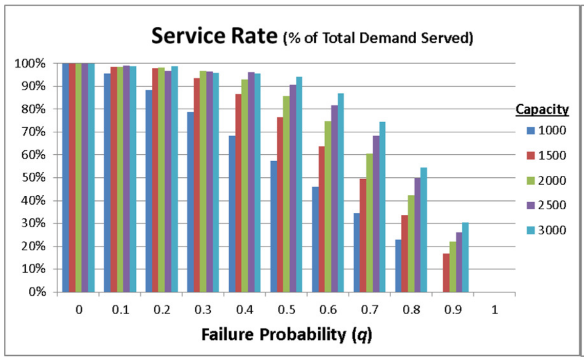
(c)



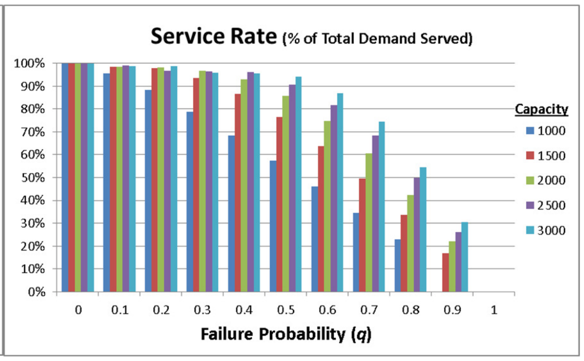
(d)

Figure 25: Effect of plant utilization requirement on total cost (a) no utilization, (b) 20% utilization, (c) 40% utilization, and (d) 60% utilization requirement.

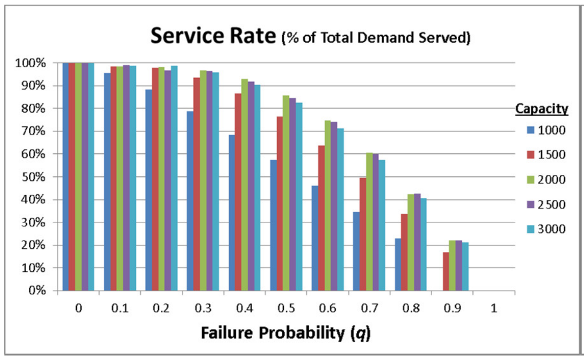
Figure 22 shows the effect of capacity utilization requirement on the service rate. Interestingly, the total demand met is no longer monotone with the capacity level under restrictive utilization requirements.



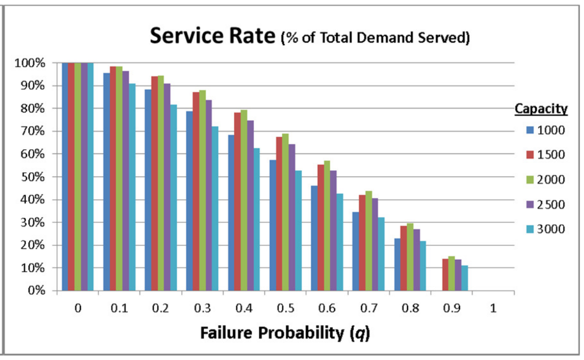
(a)



(b)



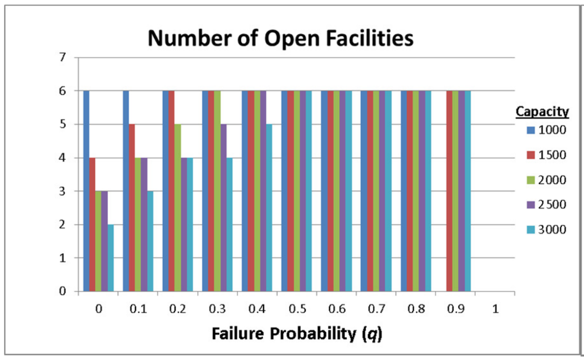
(c)

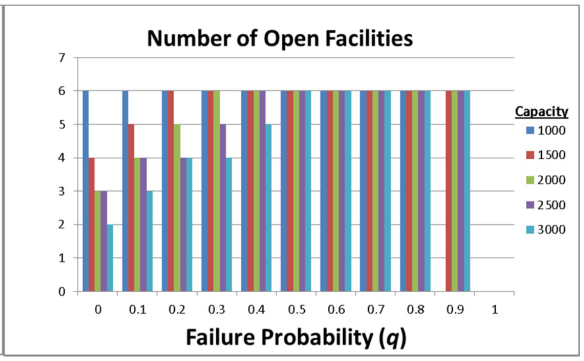


(d)

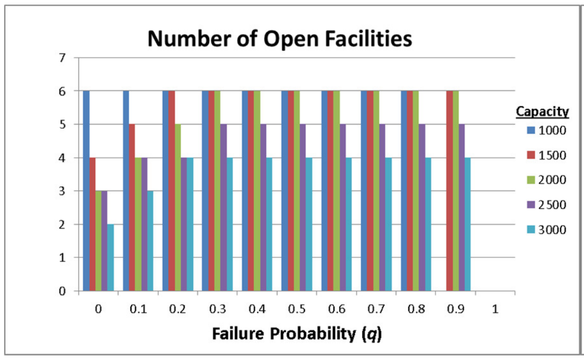
Figure 26: Effect of plant utilization requirement on total cost (a) no utilization, (b) 20% utilization, (c) 40% utilization, and (d) 60% utilization requirement.

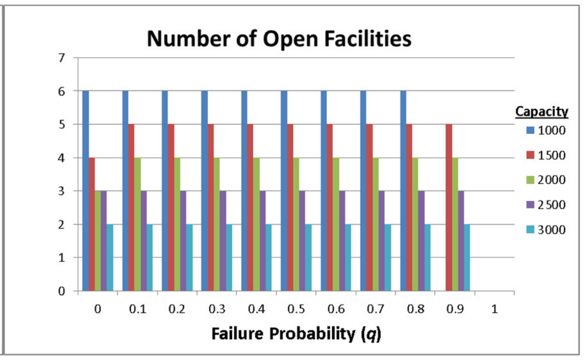
Figure 23 shows the effect of capacity utilization requirement on the number of plants.





(a) (b)

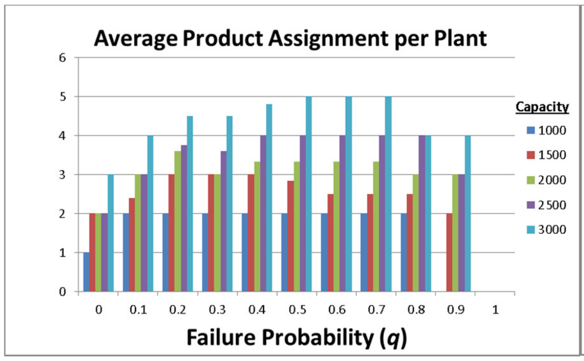


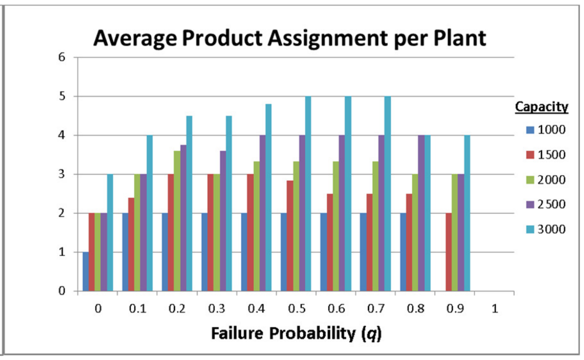


(c) (d)

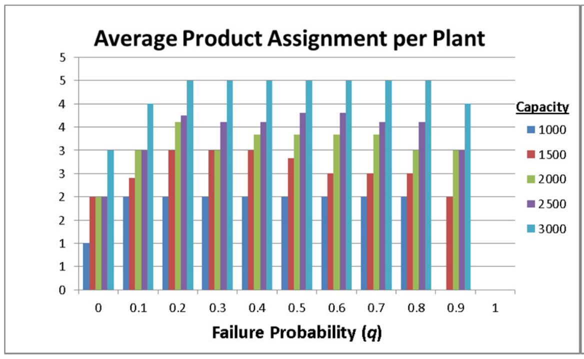
Figure 27: Effect of plant utilization requirement on number of plants open (a) no utilization, (b) 20% utilization, (c) 40% utilization, and (d) 60% utilization requirement.

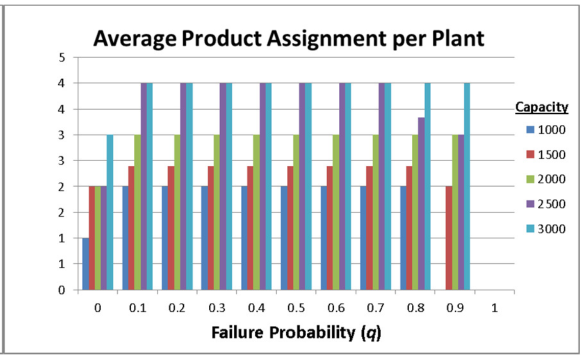
Figure 24 shows the effect of capacity utilization requirement on the average product assignment per plant.





(a) (b)





(c) (d)

Figure 28: Effect of plant utilization requirement on average number of product assignment per plant (a) no utilization, (b) 20% utilization, (c) 40% utilization, and (d) 60% utilization requirement.

CHAPTER V: CONCLUSIONS and FUTURE STUDIES

In this dissertation, we have developed hybrid algorithms for stochastic programming problems. The proposed algorithm is a hybridization of two existing methods. The first one is the Monte Carlo sampling based algorithm, which is called Sample Average Approximation (SAA). Sample Average Approximation method provides an attractive approximation for stochastic programming problems when the number of uncertain parameters increases. The second algorithm is Progressive Hedging Algorithm (PHA) which is an exact solution methodology for stochastic programming problems. The research presented in this dissertation mainly addresses two issues that arise when using SAA and PHA methods individually; lack of effectiveness in solution quality of SAA and lack of efficiency in computational time of PHA.

5.1 Summary of Study and Contributions

The first proposed algorithm is called Sampling Based Progressive Hedging Algorithm (SBPHA), which is the integration of SAA and PHA. This integration considers each sample as a small deterministic problem and employs the SAA algorithm iteratively. In each iteration, the non-anticipativity constraints is injected into the solution process by introducing penalty terms in the objective that guides the solution of each sample to the samples' balanced solution and to ensure that non-anticipativity constraints are satisfied. The two key parameters of SBPHA are the weight of the best incumbent solution and the penalty factor.

The weight for the best incumbent solution adjusts the importance given to samples' best found solution and to the most recent average sample solution in calculating the balanced solution. The penalty factor modulates the rate at which the sample solutions converge to the samples' balanced solution. Given that the best found solution improves over time, we propose two strategies for the weight of the best incumbent solution: static versus dynamic strategy.

We first conducted experiments for sensitivity analysis of the algorithm with respect to the parameters. The results show that the SBPHA's solution quality performance is relatively insensitive to the choice of strategy for the weight of the best incumbent solution, i.e., both the static and dynamic strategies are able to converge to the optimum solution. SBPHA is able to converge to the optimal solution even with small number of samples and small sample sizes.

In addition to the sensitivity experiments, we compared the performances of SBPHA and *d*-SBPHA with SAA's. These results show that the SBPHA and d-SBPHA are able to improve the solution quality noticeably with reasonable computational effort compared to SAA. Further, increasing SAA's sample size to match the solution quality performance of SBPHA requires significant computational effort which is not affordable in many practical instances.

The contributions of this research are as follows:

Contribution 1: Developed SBPHA which provides a configurable solution method that improves the sampling based methods' accuracy and PHA's efficiency for two-stage and multi-stage stochastic programming problems.

Contribution 2: Enhanced the SBPHA for SPs with binary first stage decision variables. The improved algorithm is called Discarding-SBPHA (*d*-SBPHA). It is analytically proved that SBPHA guarantee optimal solution to the mentioned problems when the number of discarding iterations approaches to infinity.

There are three possible avenues of future research on SBPHA.

First opportunity is to investigate the integration of alternative solution methodologies in order to improve the convergence rate and solution quality, such as Stochastic Decomposition (SD), Stochastic Dual Dynamic Programming (SDDP), L-Shaped decomposition.

Second extension is the research on the application of the *d*-SBPHA to the Stochastic Programs that have linear first stage decision variables.

Another extension is the investigations on the development of a general strategy for the SBPHA and d-SBPHA specific parameters in order reduce the computational effort spent on the parameter sensitivity steps.

APPENDIX A: RESULTS for SBPHA and d-SBPHA

Table 12: Objective Function Values of Test Samples

Alpha(α)	Rho		Replication (r=1,…,Rep)										Objective			
Strategy/ Parameter	Start (ρ)	Update Parameter (β)	1	2	3	4	5	6	7	8	9	10	Average	Gap	Min	Max
Dynamic/ Δα=0.02	1	1.8	9,362	9,167	9,254	9,703	9,490	9,713	9,024	9,254	9,161	9,287	**9,341**	**3.9%**	**9024**	**9713**
	20	1.2	9,825	9,167	9,124	9,083	9,448	9,713	8,995	9,083	9,167	9,254	**9,286**	**3.2%**	**8995**	**9713**
	20	1.5	9,319	9,167	9,266	9,344	9,545	9,183	8,995	9,266	9,161	9,167	**9,241**	**2.7%**	**8995**	**9545**
	40	1.3	9,528	9,024	8,995	9,547	9,478	9,362	8,995	9,083	9,161	9,138	**9,231**	**2.6%**	**8995**	**9547**
	40	1.4	9,319	9,167	9,266	9,828	9,490	9,347	8,995	9,271	9,161	9,839	**9,368**	**4.1%**	**8995**	**9839**
	80	1.1	9,825	9,032	8,995	9,208	9,478	9,221	9,024	9,083	9,124	9,287	**9,228**	**2.6%**	**8995**	**9478**
	80	1.2	9,825	9,167	8,995	9,345	9,478	9,221	8,995	9,083	9,161	9,096	**9,236**	**2.7%**	**8995**	**9478**
	200	1.1	9,345	9,167	8,995	9,398	9,448	9,333	8,995	9,083	9,161	9,316	**9,224**	**2.5%**	**8995**	**9448**
Dynamic/ Δα=0.03	1	1.8	9,528	8,995	9,024	9,637	9,490	9,713	9,024	9,288	9,471	9,287	**9,346**	**3.9%**	**8995**	**9713**
	20	1.2	9,362	9,024	8,995	9,528	9,412	9,370	9,024	9,288	9,221	9,280	**9,250**	**2.8%**	**8995**	**9528**
	20	1.5	9,825	9,292	9,266	9,467	9,490	9,713	8,995	9,254	9,161	9,138	**9,360**	**4.1%**	**8995**	**9713**
	40	1.3	9,528	9,167	9,032	9,112	9,490	9,383	8,995	9,254	9,161	9,287	**9,241**	**2.7%**	**8995**	**9490**
	40	1.4	9,825	9,292	9,032	9,467	9,478	9,713	8,995	9,254	9,221	9,326	**9,360**	**4.1%**	**8995**	**9713**
	80	1.1	9,825	9,024	8,995	9,362	9,448	9,369	8,995	9,083	9,083	9,167	**9,235**	**2.7%**	**8995**	**9448**
	80	1.2	9,528	9,104	8,995	9,944	9,478	9,221	8,995	9,112	9,161	9,287	**9,283**	**3.2%**	**8995**	**9944**
	200	1.1	9,825	9,167	8,995	9,112	9,448	8,995	9,024	9,083	9,161	9,138	**9,195**	**2.2%**	**8995**	**9448**
Dynamic/ Δα=0.05	1	1.8	9,825	9,347	9,083	10,006	9,700	9,362	9,024	9,715	9,292	9,693	**9,505**	**5.7%**	**9024**	**10006**
	20	1.2	9,362	9,104	9,210	9,715	9,539	9,515	9,024	9,161	9,292	9,183	**9,310**	**3.5%**	**9024**	**9715**
	20	1.5	9,825	9,032	9,288	10,006	9,562	9,362	8,995	9,838	9,167	9,136	**9,421**	**4.7%**	**8995**	**10006**
	40	1.3	9,679	9,266	9,210	10,006	9,490	9,183	8,995	9,838	9,471	9,167	**9,430**	**4.8%**	**8995**	**10006**
	40	1.4	9,825	9,266	9,271	9,788	9,562	9,221	8,995	9,254	9,221	9,136	**9,354**	**4.0%**	**8995**	**9788**
	80	1.1	9,825	8,995	8,995	9,362	9,292	9,221	8,995	9,838	9,221	9,287	**9,303**	**3.4%**	**8995**	**9838**
	80	1.2	9,362	9,104	8,995	10,006	9,545	9,498	9,024	9,838	9,161	9,167	**9,370**	**4.2%**	**8995**	**10006**
	200	1.1	9,825	9,167	8,995	9,658	9,490	9,221	9,024	9,254	9,167	9,083	**9,288**	**3.3%**	**8995**	**9658**

Table 12 continues…

Static/ α=0.6	**1**	**1.8**	9,825	9,032	9,266	10,006	9,208	9,362	8,995	9,083	9,292	9,515	**9,358**	**4.0%**	**8995**	**10006**
	20	**1.2**	9,751	9,104	8,995	10,006	9,161	9,362	8,995	9,083	9,221	9,483	**9,316**	**3.6%**	**8995**	**10006**
	20	**1.5**	9,547	9,271	9,203	10,006	9,208	9,713	9,024	9,083	9,221	9,332	**9,361**	**4.1%**	**9024**	**10006**
	40	**1.3**	9,547	8,995	9,124	10,006	9,161	9,713	9,024	9,083	9,292	9,404	**9,335**	**3.8%**	**8995**	**10006**
	40	**1.4**	9,528	9,271	9,124	10,006	9,161	9,362	9,024	9,083	9,221	9,586	**9,337**	**3.8%**	**9024**	**10006**
	80	**1.1**	9,362	8,995	8,995	9,528	9,161	9,319	9,024	9,254	9,024	9,112	**9,177**	**2.0%**	**8995**	**9528**
	80	**1.2**	9,547	9,032	8,995	8,995	9,161	9,167	9,024	9,083	9,167	9,167	**9,134**	**1.5%**	**8995**	**9547**
	200	**1.1**	9,362	9,287	8,995	10,006	9,124	9,221	8,995	9,254	9,167	9,096	**9,251**	**2.8%**	**8995**	**10006**
Static/ α=0.7	**1**	**1.8**	9,528	9,326	8,995	9,775	9,490	9,516	8,995	9,083	9,471	9,287	**9,347**	**3.9%**	**8995**	**9775**
	20	**1.2**	9,751	9,024	9,024	9,344	9,161	9,221	8,995	9,083	9,161	8,995	**9,176**	**2.0%**	**8995**	**9751**
	20	**1.5**	9,825	9,032	9,254	9,344	9,448	9,713	9,024	9,254	9,221	9,341	**9,346**	**3.9%**	**9024**	**9825**
	40	**1.3**	9,825	9,024	9,104	10,006	9,161	9,713	9,024	9,083	9,161	9,124	**9,322**	**3.6%**	**9024**	**10006**
	40	**1.4**	9,825	8,995	9,266	9,513	9,161	9,691	8,995	9,254	9,167	9,024	**9,289**	**3.3%**	**8995**	**9825**
	80	**1.1**	9,547	8,995	8,995	9,528	9,161	9,221	9,024	9,083	9,221	9,083	**9,186**	**2.1%**	**8995**	**9547**
	80	**1.2**	9,825	8,995	9,203	10,006	9,161	9,370	9,024	9,254	9,161	9,024	**9,302**	**3.4%**	**8995**	**10006**
	200	**1.1**	9,024	9,167	8,995	10,006	9,083	9,713	8,995	9,254	9,083	9,104	**9,242**	**2.7%**	**8995**	**10006**
Static/ α=0.8	**1**	**1.8**	9,825	9,167	9,254	9,765	9,833	9,320	9,024	9,254	9,471	9,713	**9,463**	**5.2%**	**9024**	**9833**
	20	**1.2**	9,825	8,995	9,124	9,344	9,448	9,221	8,995	9,083	9,161	9,254	**9,245**	**2.8%**	**8995**	**9825**
	20	**1.5**	9,825	9,167	9,254	9,645	9,545	9,221	8,995	9,254	9,161	9,326	**9,339**	**3.8%**	**8995**	**9825**
	40	**1.3**	9,319	9,167	9,254	9,485	9,448	9,713	8,995	9,254	9,161	9,138	**9,293**	**3.3%**	**8995**	**9713**
	40	**1.4**	9,319	8,995	9,254	9,645	9,490	9,353	8,995	9,254	9,221	9,287	**9,281**	**3.2%**	**8995**	**9645**
	80	**1.1**	9,319	8,995	8,995	9,319	9,161	9,326	8,995	9,254	9,161	9,096	**9,162**	**1.9%**	**8995**	**9326**
	80	**1.2**	9,825	9,167	9,254	9,456	9,448	9,353	8,995	9,254	9,161	9,167	**9,308**	**3.5%**	**8995**	**9825**
	200	**1.1**	9,345	9,032	8,995	9,370	9,448	9,221	8,995	9,254	9,161	9,138	**9,196**	**2.2%**	**8995**	**9448**

Table 13: Computational Time of Test Samples

Alpha(α)	Rho (ρ)		Replication (r=1,…,Rep)										Time (s)
Strategy/ Parameter	**Start**	**Update Parameter(β)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Average**
Dynamic/ Δα=0.02	**1**	**1.8**	371	518	641	546	445	534	663	448	360	394	**492.04**
	20	**1.2**	568	777	1,183	823	753	907	992	738	654	833	**822.79**
	20	**1.5**	346	428	620	472	447	590	547	596	376	440	**486.06**
	40	**1.3**	395	517	761	580	566	630	505	592	467	549	**556.10**
	40	**1.4**	370	452	624	495	464	556	505	579	384	450	**487.83**
	80	**1.1**	551	746	1,090	895	815	952	959	827	779	795	**840.79**
	80	**1.2**	386	518	799	657	574	695	625	604	484	577	**591.98**
	200	**1.1**	435	458	720	613	581	742	503	667	488	588	**579.48**
Dynamic/ Δα=0.03	**1**	**1.8**	379	468	629	489	415	512	664	426	349	421	**475.14**
	20	**1.2**	575	782	1,135	696	767	914	995	653	680	742	**794.04**
	20	**1.5**	319	447	613	411	424	512	555	468	346	431	**452.71**
	40	**1.3**	391	484	715	552	515	706	616	502	446	484	**541.12**
	40	**1.4**	337	447	629	394	426	508	505	459	382	466	**455.29**
	80	**1.1**	515	729	1,115	823	771	1,089	863	726	768	710	**810.94**
	80	**1.2**	388	546	767	531	501	649	612	568	479	568	**560.92**
	200	**1.1**	346	401	696	579	537	777	449	556	515	595	**544.99**
Dynamic/ Δα=0.05	**1**	**1.8**	393	439	646	356	387	499	648	384	329	381	**446.36**
	20	**1.2**	641	688	1,026	784	653	876	914	704	568	656	**751.03**
	20	**1.5**	348	398	586	405	385	469	531	372	354	418	**426.60**
	40	**1.3**	409	524	702	458	451	581	584	427	415	471	**502.21**
	40	**1.4**	340	466	613	523	366	523	497	391	379	423	**452.01**
	80	**1.1**	477	660	1,051	836	630	794	814	520	649	652	**708.24**
	80	**1.2**	398	491	701	478	467	603	577	402	499	488	**510.43**
	200	**1.1**	304	376	639	483	439	509	403	472	398	508	**453.18**

Table 13 Continues…

Static/ α=0.6	**1**	**1.8**	399	533	638	385	390	519	627	444	349	419	**470.38**
	20	**1.2**	649	740	1,229	554	709	797	967	710	675	837	**786.60**
	20	**1.5**	391	462	632	315	408	426	508	410	876	410	**483.75**
	40	**1.3**	439	475	754	346	463	518	588	485	425	491	**498.48**
	40	**1.4**	379	424	633	306	366	456	457	410	369	945	**474.61**
	80	**1.1**	670	665	1,206	741	720	413	692	688	816	1,402	**801.07**
	80	**1.2**	450	430	789	410	438	599	508	1,023	449	554	**564.96**
	200	**1.1**	421	390	698	269	370	499	373	400	427	582	**442.95**
Static/ α=0.7	**1**	**1.8**	393	452	658	451	440	562	634	456	349	441	**483.58**
	20	**1.2**	641	726	1,209	706	762	1,046	976	681	721	768	**823.52**
	20	**1.5**	355	407	625	373	435	510	525	386	364	404	**438.48**
	40	**1.3**	442	476	780	412	464	531	578	481	441	571	**517.60**
	40	**1.4**	385	403	631	432	389	449	510	364	383	436	**438.24**
	80	**1.1**	683	673	1,235	804	704	903	802	795	764	976	**833.96**
	80	**1.2**	415	473	832	375	458	618	535	447	508	563	**522.35**
	200	**1.1**	491	385	730	311	536	482	390	420	558	601	**490.37**
Static/ α=0.8	**1**	**1.8**	384	461	654	458	469	540	671	433	356	416	**484.20**
	20	**1.2**	623	816	1,251	746	838	996	1,017	732	695	885	**859.79**
	20	**1.5**	459	414	642	503	462	596	565	410	405	417	**487.37**
	40	**1.3**	405	506	737	580	558	582	623	464	460	555	**546.96**
	40	**1.4**	459	411	630	517	458	561	510	417	376	412	**475.13**
	80	**1.1**	744	682	1,302	891	929	1,210	910	661	796	1,046	**917.22**
	80	**1.2**	452	523	750	615	590	726	618	461	490	623	**584.84**
	200	**1.1**	458	421	753	600	546	601	452	471	477	589	**536.55**

Table 14: Average Objective Function Value of 10 Replications for Each Parameter Configuration over Discarding

	q=0.1				q=0.2			
	Static α			Dynamic α	Static α			Dynamic α
	α=0.6	α=0.7	α=0.6	Δα=0.03	α=0.6	α=0.7	α=0.6	Δα=0.03
	ρ=200	ρ=200	ρ=80	ρ=200	ρ=200	ρ=200	ρ=80	ρ=200
Replications	β=1.1	β=1.1	β=1.2	β=1.1	β=1.1	β=1.1	β=1.2	β=1.1
No Dicard	6,002	6,007	6,002	6,015	7,554	7,511	7,557	7,541
1-Discard	5,990	5,996	5,990	6,008	7,540	7,502	7,520	7,487
2-Discard	5,990	5,993	5,990	5,990	7,520	7,451	7,484	7,476
3-Discard	5,990	5,993	5,990	5,990	7,463	7,447	7,464	7,476
4-Discard	5,990	5,990	5,990	5,990	7,458	7,447	7,447	7,476
5-Discard	5,990	5,990	5,990	5,990	7,458	7,447	7,447	7,476
6-Discard	5,990	5,990	5,990	5,990	7,447	7,447	7,447	7,470
7-Discard	5,990	5,990	5,990	5,990	7,447	7,447	7,447	7,470
8-Discard	5,990	5,990	5,990	5,990	7,447	7,447	7,447	7,470
9-Discard	5,990	5,990	5,990	5,990	7,447	7,447	7,447	7,470
10-Discard	5,990	5,990	5,990	5,990	7,447	7,447	7,447	7,459
Exact Solution	5,990				7,447			

	q=0.3				q random			
	Static α			Dynamic α	Static α			Dynamic α
	α=0.6	α=0.7	α=0.6	Δα=0.03	α=0.6	α=0.7	α=0.6	Δα=0.03
	ρ=200	ρ=200	ρ=80	ρ=200	ρ=200	ρ=200	ρ=80	ρ=200
Replications	β=1.1	β=1.1	β=1.2	β=1.1	β=1.1	β=1.1	β=1.2	β=1.1
No Dicard	9,251	9,242	9,134	9,195	6,851	6,938	6,862	6,987
1-Discard	9,220	9,059	9,109	9,116	6,842	6,859	6,861	6,927
2-Discard	9,137	9,057	9,081	9,113	6,822	6,859	6,861	6,801
3-Discard	9,100	9,057	9,081	9,110	6,801	6,812	6,820	6,790
4-Discard	9,067	9,033	9,076	9,081	6,790	6,801	6,810	6,790
5-Discard	9,057	9,033	9,067	9,081	6,790	6,801	6,800	6,790
6-Discard	9,057	9,033	9,067	9,065	6,790	6,790	6,800	6,790
7-Discard	9,057	9,033	9,067	9,065	6,790	6,790	6,800	6,790
8-Discard	9,057	9,033	9,059	9,065	6,790	6,780	6,800	6,790
9-Discard	9,057	9,033	9,059	9,065	6,780	6,780	6,800	6,780
10-Discard	9,057	9,024	9,056	9,065	6,780	6,780	6,800	6,780
Exact Solution	8,995				6,780			

Table 15: Average CPU Time of 10 Replications for Each Parameter Configuration over Discarding

	q=0.1				q=0.2			
	Static α			Dynamic α	Static α			Dynamic α
	α=0.6	α=0.7	α=0.6	Δα=0.03	α=0.6	α=0.7	α=0.6	Δα=0.03
	ρ=200	ρ=200	ρ=80	ρ=200	ρ=200	ρ=200	ρ=80	ρ=200
Replications	β=1.1	β=1.1	β=1.2	β=1.1	β=1.1	β=1.1	β=1.2	β=1.1
No Dicard	248	250	288	268	333	366	366	380
1-Discard	108	121	106	146	98	101	128	149
2-Discard	124	130	137	143	110	164	108	107
3-Discard	97	116	97	108	131	102	60	77
4-Discard	131	143	114	143	85	106	122	102
5-Discard	88	93	94	109	117	126	114	133
6-Discard	101	146	85	125	103	80	81	110
7-Discard	106	105	98	93	106	106	63	85
8-Discard	75	78	64	103	103	102	88	47
9-Discard	82	86	84	87	89	116	89	93
10-Discard	122	117	97	108	86	76	79	71
Total Time	1,282	1,386	1,263	1,434	1,360	1,444	1,297	1,354
Time (Best Solution found)	114	160	165	165	515	458	521	523

	q=0.3				q random			
	Static α			Dynamic α	Static α			Dynamic α
	α=0.6	α=0.7	α=0.6	Δα=0.03	α=0.6	α=0.7	α=0.6	Δα=0.03
	ρ=200	ρ=200	ρ=80	ρ=200	ρ=200	ρ=200	ρ=80	ρ=200
Replications	β=1.1	β=1.1	β=1.2	β=1.1	β=1.1	β=1.1	β=1.2	β=1.1
No Dicard	430	476	503	533	279	308	302	329
1-Discard	172	210	100	204	133	173	155	157
2-Discard	138	132	78	139	126	148	119	163
3-Discard	107	122	75	137	124	147	109	130
4-Discard	96	116	72	106	78	91	85	88
5-Discard	110	136	76	66	89	100	82	80
6-Discard	117	119	57	68	106	79	73	76
7-Discard	110	102	50	62	108	98	57	95
8-Discard	89	72	50	78	98	120	64	101
9-Discard	97	67	50	52	97	147	65	78
10-Discard	73	65	47	35	83	75	52	88
Total Time	1,540	1,618	1,159	1,479	1,320	1,485	1,162	1,384
Time (Best Solution found)	648	686	727	692	423	499	373	614

APPENDIX B: DATA SET USED in CHAPTER III for CRFLP

First column of Table 16 shows whether the possible facility site is reliable or not. Second column shows the facility no. Third and fourth columns show the location (lat-long) of the facility sites. Fifth column presents the demand of the location and sixth column presents the fix opening cost that is going to be applied if a facility is opened on the specified location. Lastly the seventh column presents the failure probability if it is a random failure case otherwise all values in this column (only rows 1-12) are equal, e.g., $q = q_i = 0.1$. Emergency cost (demand satisfying cost if the demand is not satisfied from an opened facility but from emergency, e.g., dummy, facility) is 20 and is equal for all facility sites. Capacity for all facilities is taken 2000.

Table 16: Data set for CRFLP

Facility		**lat**	**long**	**Demand**	**Fixed Open Cost**	**Failure Probability**
Type	**No**					
Unreliable	**1**	0.82	0.18	957	938	0.24
	2	0.54	0.70	202	642	0.12
	3	0.91	0.72	186	1,230	0.13
	4	0.15	0.31	635	1,008	0.11
	5	0.74	0.16	737	1,279	0.08
	6	0.58	0.92	953	1,431	0.25
	7	0.60	0.09	450	1,187	0.29
	8	0.37	0.19	188	1,044	0.30
	9	0.70	0.52	206	1,466	0.26
	10	0.22	0.40	995	989	0.17
	11	0.50	0.45	429	948	0.17
	12	0.30	0.52	528	585	0.24
Reliable	**13**	0.95	0.20	570	2,000	0.00
	14	0.65	0.07	938	2,000	0.00
	15	0.53	0.11	726	2,000	0.00
	16	0.95	0.95	533	2,000	0.00
	17	0.15	0.13	565	2,000	0.00
	18	0.31	0.40	322	2,000	0.00
	19	0.98	0.73	326	2,000	0.00
	20	0.59	0.04	663	2,000	0.00

REFERENCES

Ahmed, S., A. J. King, et al. (2003). "A multi-stage stochastic integer programming approach for capacity expansion under uncertainty." Journal of Global Optimization **26**(3-24).

Ahmed, S. and A. Shapiro (2002). The sample average approximation method for stochastic programs with integer recourse, Georgia Institute of Technology.

Aksen, D., K. Altinkemer, et al. (2003). "The single-item lot-sizing problem with immediate lost sales." European Journal of Operational Research **147**(558-566).

Aksin, O. and F. Karaesmen (2007). "Characterizing the performance of process flexibility structures." Operations Research Letters **35**(4): 477-484.

Akturk, M. S. and S. Onen (2002). "Dynamic lot sizing and tool management in automated manufacturing systems." Computers and Operations Research **29**: 1059-1079.

Alp, O., N. K. Erkip, et al. (2003). "Optimal lot-sizing/vehicle-dispatching policies under stochastic lead times and stepwise fixed costs." Operations Research **51**(1): 160-166.

Baker, K. R. (1977). "An experimental study of the effectiveness of rolling schedules in production planning." Decision Sciences **8**(19-27).

Barany, I., T. J. V. Roy, et al. (1984). "Strong formulations for multi-item capacitated lot sizing." Management Science **30**(10): 1255-1261.

Bayraksan, G. and D. P. Morton (2009). Assessing solution quality in stochastic programs via sampling. Tutorials in Operations Research. INFORMS**:** 102-122.

Belvaux, G. and L. A. Wolsey (2002). "Modeling practical lot-sizing problems as mixed integer programs." Management Science **47**(7): 993-1007.

Benders, J. F. (1962). "Partitioning procedures for solving mixed variable programming problems." Numerical Math **4**: 283-252.

Birge, J. R. (1985). "Decomposition and partitioning methods for multistage stochastic linear programs." Operations Research **33**(5): 989-1007.

Birge, J. R. and F. V. Louveaux (1997). Introduction to Stochastic Programming. New York.

Bish, E., A. Muriel, et al. (2005). "Managing flexible capacity in a make to- order environment." Management Science **51**(2): 167-180.

Brennan, P. (2011) "Lessons learned from the Japan earthquake." Disaster Recovery Journal **24**.

Caroe, C. C. and J. Tind (1997). "L-shaped decomposition of two-stage stochastic programs with integer recourse." Mathematical Programming **83**: 451-464.

Chen, C.-F., P. J. Egbelu, et al. (1994). "Production planning models for a central factory with multiple satellite factories." International Journal of Production Research **32**(6): 1431-1450.

Chiralaksanakul, A. (2003). Monte Carlo Methods for Multi-stage Stochastic Programs. Doctor of Philosophy.

Chod, J., N. Rudi, et al. (2009). Mix, time, and volume flexibility: Valuation and corporate diversification. Chicago, USA, Northwestern University.

Chou, M. C., G. A. Chua, et al. (2010). "Design for process flexibility: Efficiency of the long chain and sparse structure." Operations Research **58**: 43-58.

Chow, Y. S. and H. Robbins (1965). "On the asymptotic theory of fixed-width sequential confidence intervals for the mean." Annals of Mathematical Statistics **36**: 457-462.

Clark, A. R. and S. J. Clark (2000). "Rolling-horizon lot-sizing when set-up times are sequence dependent." International Journal of Production Research **38**(10): 2287-2307.

Crainic, T. G., X. Fu, et al. (2009). "Progressive hedging-based meta-heuristics for stochastic network design." CIRRELT **3**: 1-20.

Dantzig, G. B. (1955). "Linear programming under uncertainty." Management Science **1**(3-4): 197-206.

Denizel, M., S. Erengüç, et al. (1997). "Dynamic lot-sizing with setup cost reduction." European Journal of Operational Research **100**: 537-549.

Dogramaci, A., J. C. Panayiotopulos, et al. (1982). "The dynamic lot-sizing problem for the multiple items under limited capacity." AIIE Transactions **13**(4): 294-303.

Elabassi, S. B., S. Benjaafar, et al. (2010). On the performance of flexibility chaining in location-allocation problems under disruption. 8th International Conference of Modeling and Simulation-MOSIM'10, Hammamet, Tunisia.

Fan, T. and C. Liu (2010). "Solving stochastic transportation network protection problem using the progressive hedging-based method." Network and Spatial Economics **10**(2): 193-208.

Gade, D. (2007). Capacitated facilities location problems with unreliable facilities. Master's Thesis, University of Arkansas.

Garavelli, A. C. (2003). "Flexibility configurations for the supply chain management." International Journal of Production Economics **85**: 141-153.

Graves, S. C. (2008). Flexibility Principles. Chapter 3 in Building Intuition: Insights from basic operations management models and principles. C. a. L. Eds, Springer US.

Guan, Y., S. Ahmed, et al. (2006). "A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem." Mathematical Programming **105**(1): 55-84.

Haugen, K. K., A. Lokketangen, et al. (2001). "Progressive hedging as a meta-heuristic applied to stochastic lot-sizing." European Journal of Operational Research **132**: 116-122.

Helgason, T. and S. W. Wallace (1991). "Approximate scenario solutions in the progressive hedging algorithm: A numerical study with an application to fisheries management." Annals of Operations Research **31**: 425-444.

Higle, J. L. (2005). Stochastic programming: Optimization when uncertainty matters. In: Tutorials in Operations Research.

Higle, J. L. and S. Sen (1991). "Stochastic decomposition: an algorithm for two-stage linear programs with recourse." Mathematics of Operations Research **16**: 650–669.

Infanger, G. (1992). "Monte Carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs." Annals of Operations Research **39**(69-95).

Iravani, S. M., M. P. V. Oyen, et al. (2005). "Structural flexibility: A new perspective on the design of manufacturing and service operations." Management Science **51**(2): 151-166.

Jans, R. and Z. Degraeve (2004). "An industrial extension of the discrete lot sizing and scheduling problem." IIE Transactions **36**(1): 47-58.

Jans, R. and Z. Degraeve (2007). "Meta-heuristics for dynamic lot-sizing: a review and comparison of solution approaches." European Journal of Operational Research **177**: 1855-1875.

Jordan, W. C. and S. C. Graves (1995). "Principles on the benefits of manufacturing process flexibility." Management Science **41**: 577-594.

Kall, P. and S. W. Wallace (1994). Stochastic Programming. New York, Wiley.

Karimi, B., S. M. T. F. Ghomi, et al. (2003). "The capacitated lot-sizing problem: a review of models and algorithms." Omega **31**: 365-378.

Karmarkar, U. S. and L. Schrage (1985). "The deterministic dynamic product cycling problem." Operations Research **33**(2): 326-345.

Kleywegt, A. J., A. Shapiro, et al. (2001). "The Sample Average Approximation Method for Stochastic Discrete Optimization." SIAM Journal on Optimization **12**(2): 479:502.

Kuik, R., M. Salomon, et al. (1994). "Batching decisions: structure and models." European Journal of Operational Research **75**: 243-263.

Laporte, G. and F. V. Louveaux (1993). "The integer L-shaped method for stochastic integer programs with complete recourse." Operations Research Letters **13**: 133-142.

Law, A. M. and W. D. Kelton (1982). "Confidence intervals for steady-state simulations II: A survey of sequential procedures." Management Science **28**: 550-562.

Law, A. M., W. D. Kelton, et al. (1981). "Relative Width Sequential Confidence Intervals For the Mean." Communications in Statistics **B10**: 29-39.

Lokketangen, A. and D. L. Woodruff (1996). "Progressive hedging and tabu search applied to mixed Integer (0,1) multi stage stochastic programming." Journal of Heuristics **2**(2): 111-128.

Louveaux, F. V. and M. H. v. d. Vlerk (1993). "stochastic Programming with simple integer recourse." Mathematical Programming **61**: 301-325.

Masihtehrani, B. (2011). Stochastic analysis of disruption in supply chain networks. Doctor of Philosophy, Pennsylvania State University.

Mekler, V. A. (1993). "Setup cost reduction in the dynamic lot-size model." Journal of Operations Management **11**: 35-43.

Miller, A. J. and L. A. Wolsey (2003). "Tight formulations for some simple mixed integer programs and convex objective integer programs." Mathematical Programming **Serial B**(98): 73-88.

Morton, D. and E. Popova (2001). Monte Carlo simulations for stochastic optimization. Encyclopedia of Optimization. I. C. A. Floudas and P. M. Pardalos, Kluwer Academic Publishers.

Mulvey, J. M., D. P. Rosenbaum, et al. (1997). "Strategic financial risk management and operations research." European Journal of Operational Research **97**: 1-16.

Mulvey, J. M. and A. Ruszczynski (1995). "A new scenario decomposition method for large-scale stochastic optimization." Operations Research **43**: 477-490.

Mulvey, J. M. and H. Vladimirou (1991). "Applying the progressive hedging algorithm to stochastic generalized networks." Annals of Operations Research **31**: 399-424.

Mulvey, J. M. and H. Vladimirou (1992). "Stochastic network programming for financial planning problems." Management Science **39**(11): 1642-1664.

Muriel, A., A. Somasundaram, et al. (2006). "Impact of partial manufacturing flexibility on production variability." Manufacturing Service Operations Management **8**(2): 192-205.

Norkin, V. I., G. C. Pflug, et al. (1998). "A branch and bound method for stochastic global optimization." Mathematical Programming **83**: 425-450.

Olsen, P. (1976). "Discretization of multistage stochastic programming problems." Mathematical Programming Study **6**: 111-124.

Peng, P., L.-V. Snyder, et al. (2011). "Reliable logistics networks design with facility disruptions." Transportation Research, Part B **45**: 1190-1211.

Pennanen, T. (2005). "Epi-Convergent discretizations of multistage stochastic programs." Mathematical Operations Research **30**: 245-256.

Penuel, J., J. C. Smith, et al. (2010). "An integer decomposition algorithm for solving a two-stage facility location problem with second-stage activation cost." Naval research Logistics **57**: 391-402.

Robinson, S. (1991). "Extended scenario analysis." Annals of Operations Research **31**: 385-398.

Rockafellar, R. T. and R. J.-B. Wets (1991). "Scenarios and policy aggregation in optimization under uncertainty." Mathematics and Operations Research **16**: 119-147.

Rosa, C. H. and A. Ruszczynski (1996). "On augmented Lagrangean decomposition methods for multistage stochastic programs." Annals of Operations Research **64**: 289-309.

Rubinstein, R. Y. and A. Shapiro (1990). "Optimization of static simulation models by the score function method." Mathematics and Computers in Simulation **32**: 373-392.

Ruszczynski, A. and A. Shapiro (2003). Monte Carlo Sampling. Handbooks in OR and MS, Elsevier Science. **10:** 353-403.

Ruszczynski, A. and A. Shapiro (2003). Stochastic Programming Models. Stochastic Programming, Handbooks in Operations Research and Management Science. **10**.

Schütz, P., A. Tomasgard, et al. (2009). "Supply chain design under uncertainty using sample average approximation and dual decomposition." European Journal of Operational Research **199**(2): 409-419.

Sethi, A. and S. Sethi (1990). "Flexibility in manufacturing: A survey." International Journal of Flexible Manufacturing Systems **2**: 289-328.

Shapiro, A. (2002) "Statistical inference of multistage stochastic programming problems." Optimization Online. www.optimization-online.org.

Shapiro, A. (2003). "Inference of statistical bounds for multistage stochastic programming problems." Mathematical Methods of Operations Research **58**: 57-68.

Shapiro, A. (2005). Complexity of Two and Multi-stage Stochastic Programming Problems. Tutorial Notes for School of Industrial and Systems Engineering. Atlanta, Georgia 30332-0205, USA, Georgia Institute of Technology**:** 1-27.

Shapiro, A. (2008). "Stochastic programming approach to optimization under uncertainty." Mathematical Programming **ser. B , 112**: 183-220.

Shapiro, A. and T. Homem-de-Mello (2001). "On the rate of convergence of Monte Carlo approximations of stochastic programs." SIAM Journal on Optimization **11**: 76-86.

Sheffi, Y. (2005). The resilient enterprise: Overcoming vulnerabilities for competitive advantage. MIT Press. Cambridge, MA.

Shen, Z.-J. M., R. L. Zhan, et al. (2011). "The reliable facility location problem: Formulations, heuristics, and approximation algorithms." Informs Journal on Computing **23**(3): 470-482.

Slyke, R. M. V. and R. J.-B. Wets (1969). "L-shaped linear programs with applications to optimal control and stochastic programming." SIAM Journal on Applied Mathematics **17**: 638–663.

Snyder, L.-V. and M. S. Daskin (2005). "Reliability models for facility location: The expected failure cost case." Transportation Science **39**: 400-416.

Snyder, L.-V., M.-P. Scaparra, et al. (2006). Planning for disruptions in supply chain networks. Informs. F. i. T. i. O. Research. Baltimore, MD, USA.

Snyder, L.-V. and N. Ülker (2005). A model for locating capacitated, unreliable facilities. IERC Conference. Atlanta, GA, USA.

Solak, S. (2007). Efficient solution procedures for multistage stochastic formulations of two problem classes. Doctor of Philosophy, Georgia Institute of Technology.

Sox, C. A. (1997). "Dynamic lot-sizing with random demand and non-stationary costs." Operations Research Letters **20**(15-164).

Stadtler, H. (2003). "Multilevel lot-sizing with set-up times and multiple constrained resources: internally rolling schedules with lot-sizing windows " Operations Research **51**(3): 487-502.

Staggemeier, A. T. and A. R. Clark (2001). A survey of lot-sizing and scheduling models. 23rd Annual Symposium of the Brazilian Operational Research Society (SOBRAPO).

Takriti, S., J. R. Birge, et al. (1996). "A stochastic model for the unit commitment problem." IEEE Transactions on Power Systems **11**: 1497-1508.

TheGuardian (2011) "Toyota profit slides on Japan earthquake disruption."

Verweij, B., S. Ahmed, et al. (2003). "The sample average approximation method applied to stochastic routing problems: A computational study." Computational Optimization and Applications **24**: 289-333.

Wagner, H. M. and T. M. Whitin (1958). "Dynamic version of the economic lot size model." Management Science **5**(1): 89-96.

Wallace, S. W. and T. Helgason (1991). "Structural properties of the progressive hedging algorithm." Annals of Operations Research **31**: 445-456.

Wang, W. (2007). Sample average approximation of risk-averse stochastic programs. Doctor of Philosophy, Georgia Institute of Technology.

Watson, J. P. and D. L. Woodruff (2011). "Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems." Computational Management Science **8**: 355-370.

Wolsey, L. A. (1995). "Progress with single-item lot-sizing." The European Journal of Operational Research **86**: 395-401.

Woodruff, D. L. and S. Vob (2006). Planning for a big-bang in a supply chain: fast hedging for production indicators. Proceedings of the 39th Hawaii International Conference on System Sciences.

Zangwill, W. I. (1969). "Backlogging model and a multi-echelon model of a dynamic economic lot size production system– A network approach." Management Science **15**(9): 506-527.

Zhan, R.-L. (2007). Models and algorithms for reliable facility location problems and system reliability optimization. Doctor of Philosophy, University of Florida.

ABSTRACT

SAMPLING BASED PROGRESSIVE HEDGING ALGORITHMS FOR STOCHASTIC PROGRAMMING PROBLEMS

by

NEZIR AYDIN

August 2012

Advisor: Dr. Alper E. Murat

Major: Industrial and Systems Engineering

Degree: Doctor of Philosophy

Many real-world optimization problems have parameter uncertainty. For instances where the uncertainties can be estimated to a certain degree, stochastic programming (SP) methodologies are used to identify robust plans. Despite advances in SP, it is still a challenge to solve real world stochastic programming problems, in part due to the exponentially increasing number of scenarios. For two-stage and multi-stage problems, the number of scenarios increases exponentially with the number of uncertain parameters, and for multi-stage problems also with the number of decision stages.

In the case of large scale mixed integer stochastic problem instances, there are usually two common approaches: approximation methods and decomposition methods. Most common sampling-based approximation (SAA) SP technique is the Monte Carlo sampling-based method. The Progressive Hedging Algorithm (PHA) on the other hand can optimally solve large problems through the decomposition into smaller problem instances. The SAA, while effectively used in many applications, can

lead to poor solution quality if the selected sample sizes are not sufficiently large. With larger sample sizes and multi-stage SPs, however, the SAA method is not practical due to the significant computational effort required. In contrast, PHA suffers from the need to solve many sub-problems iteratively which is computationally expensive.

In this dissertation, we develop novel SP algorithms integrating sampling based SAA and decomposition based PHA SP methods. The proposed integrated methods are novel in that they marry the complementary aspects of PHA and SAA in terms of exactness and computational efficiency. Further, the developed methods are practical in that they allow the analyst to calibrate the tradeoff between the exactness and speed of attaining a solution.

We demonstrate the effectiveness of the developed integrated approaches, Sampling Based Progressive Hedging Algorithm (SBPHA) and Discarding SBPHA (*d*-SBPHA), over the pure strategies (i.e. SAA or PHA) as well as other commonly used SP methods through extensive experimentation. In addition, we develop alternative hybridization strategies and present results of extensive experiments for these strategies under different uncertainty models. The validation of the methods is demonstrated through Capacitated Reliable facility Location Problem (CRFLP) and Multi-stage stochastic lot-sizing problems.

AUTOBIOGRAPHICAL STATEMENT

Nezir Aydin was born in Batman, Turkey on June 26, 1981, the son of Ibrahim and Medine Aydin. He received the Bachelor of Science in 2005 from Yildiz Technical University, Istanbul/Turkey. When he was an undergraduate student he worked for Lider Lighting Co. as a Quality Department Manager/ISO Quality Standards practitioner in 2004 and 2005. After receiving his B.S degree he worked at Yildiz Technical University as a Research Assistant. He received Master of Science degree in Industrial Engineering from the Yildiz Technical University, Istanbul, Turkey, in 2007.

In 2007, he resumed his studies to earn a Ph.D. in the Industrial and Systems Engineering at the Wayne State University, Detroit, Michigan/USA. Upon graduation, he plans to work as an assistant professor at the Industrial Engineering Department at the Yildiz Technical University.

During his studies at Wayne State University and Yildiz Technical University, he made a number of technical presentations at INFORMS, SAE and several conferences. His articles have been published and/or are under review in journals like International Journal of Production Economics and Computers & Industrial Engineering. He is a member of INFORMS and IIE.