ALGORITHMS FOR STOCHASTIC INTEGER PROGRAMS USING FENCHEL CUTTING PLANES

A Dissertation

by

SARAVANAN VENKATACHALAM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Lewis Ntaimo
Committee Members,	Sergiy Butenko
	Kiavash Kianfar
	Subodha Kumar
Head of Department,	César O. Malavé

August 2014

Major Subject: Industrial Engineering

ABSTRACT

This dissertation develops theory and methodology based on Fenchel cutting planes for solving stochastic integer programs (SIPs) with binary or general integer variables in the second-stage. The methodology is applied to auto-carrier loading problem under uncertainty. The motivation is that many applications can be modeled as SIPs, but this class of problems is hard to solve. In this dissertation, the underlying parameter distributions are assumed to be discrete so that the original problem can be formulated as a deterministic equivalent mixed-integer program. The developed methods are evaluated based on computational experiments using both real and randomly generated instances from the literature. We begin with studying a methodology using Fenchel cutting planes for SIPs with binary variables and implement an algorithm to improve runtime performance.

We then introduce the stochastic auto-carrier loading problem where we present a mathematical model for tactical decision making regarding the number and types of auto-carriers needed based on the uncertainty of availability of vehicles. This involves the auto-carrier loading problem for which actual dimensions of the vehicles, regulations on total height of the auto-carriers and maximum weight of the axles, and safety requirements are considered. The problem is modeled as a two-stage SIP, and computational experiments are performed using test instances based on real data.

Next, we develop theory and a methodology for Fenchel cutting planes for mixed-integer programs with special structure. Integer programs have to be solved to generate a Fenchel cutting plane and this poses a challenge. Therefore, we propose a new methodology for constructing a reduced set of integer points so that the generation of Fenchel cutting planes is computationally favorable. We then present the computational results based on randomly generated instances from the literature

and discuss the limitations of the methodology. We finally extend the methodology to SIPs with general integer variables in the second-stage with special structure, and study different normalizations for Fenchel cut generation and report their computational performance.

DEDICATION

To my wife, Suja, and daughters, Aditi and Ananya.

ACKNOWLEDGEMENTS

First and foremost, my sincere thanks to my advisor, Dr. Lewis Ntaimo for introducing me to the subject, and for his patience and unconditional support thorough out my research. I would like to thank my committee members, Dr. Sergiy Butenko, Dr. Kiavash Kianfar, and Dr. Subodha Kumar, for their valuable inputs and for introducing me to different facets of academia research. I would also like to thank Dr. Guy L. Curry who inspired me during my graduate days to pursue a career in the field of operations research. I would like to thank the department for their continued support. I am indebted and eternally grateful to my parents, my in-laws and my brother for their patience and support throughout my period of education.

NOMENCLATURE

ALC	Auto-Carrier logistic company
BAB	Branch-and-bound
BAC	Branch-and-cut
CPU	Central processing unit
DEP	Deterministic equivalent problem
EV	Expected value
FCG	Fenchel cut generation
FD	Fenchel decomposition
IP	Integer programming
ISG	Integer set generation
LP	Linear programming
max	Maximum value for the given parameters
Max	Maximize the objective function
min	Minimum value for the given parameters
Min	Minimize the objective function
MIP	Mixed-integer programming
MP	Master problem
SACP	Stochastic auto-carrier problem
SFP	Starting feasible point
SIP	Stochastic integer programming
SMIP	Stochastic mixed-integer programming
SP	Sub problem
ST-FD	Stage-wise Fenchel decomposition
VSS	Value of stochastic solution

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation and Problem Statement

Uncertainty is a key ingredient in many decision making problems. Financial planning, airline scheduling, unit commitment in power systems, and supply chain network planning are a few examples of areas in which ignoring uncertainty often results in sub-optimal decisions. Recent advancements in the availability of computing power and mathematical techniques have made decision making under uncertainty an important field of study. In this research, we study stochastic integer programs (SIPs). SIPs are a class of optimization problems in which some of the input parameters for the model are not known with certainty. We make the assumption that we know the probability distribution of the 'uncertain' parameters, and hence the objective function will explicitly include all the outcomes of the uncertain parameter rather than using an expected value for the uncertain parameters. In a two-stage SIP, first-stage decisions are made here-and-now before the future outcomes are known. In the second-stage, the outcomes of the uncertain parameters are considered, and if necessary, corrective or recourse actions are made. The uncertainties in the SIPs are represented by probability distributions.

In the last two decades, there has been a steady growth in the development of efficient solution methods for SIPs. Though there is a considerable amount of literature on solving stochastic programming models with continuous variables in the second-stage, the developments for general integer variables are very limited. This is due to the fact that SIPs are difficult to solve in general. This research develops a new methodology for solving stochastic programs with general integer variables.

1.1.1 Stochastic Mixed-Integer Programming

Following is a two-stage SIP formulation where we minimize the sum of first-stage costs and expected second-stage costs:

$$\begin{aligned} \text{SIP2: Min } & c^\top x + \mathcal{Q}_E(x) \\ \text{s.t. } & Ax \geq b \\ & x \in X. \end{aligned} \tag{1.1}$$

In the above formulation, $\mathcal{Q}_E(x)$ denotes the expected second-stage cost based on the first-stage decision x. The set X imposes binary restrictions on all or some components of x. The objective function coefficients, technology matrix and righthand side are assumed to be stochastic. Therefore, the function $\mathcal{Q}_E(x)$ is given as

$$\mathcal{Q}_E(x) = \mathbb{E}_\omega \Phi(q(\omega), h(\omega) - T(\omega)x, \omega). \tag{1.2}$$

The second-stage value function Φ is given as

$$\Phi(\rho, \tau, \omega) = \text{Min}\{\rho^\top y(\omega) : Wy(\omega) \leq \tau, 0 \leq y(\omega) \leq u, y(\omega) \in Y\}. \tag{1.3}$$

In problem SIP2, x denotes the first-stage decision vector, $c \in \mathbb{R}^{n_1}$ is the first-stage cost vector, $b \in \mathbb{R}^{m_1}$ is the first-stage righthand side, and $A \in \mathbb{R}^{m_1 \times n_1}$ is the first-stage constraint matrix. In the second-stage formulation (1.3), $y(\omega)$ denotes the recourse decision vector, $q(\omega) \in \mathbb{R}^{n_2}$ is the cost vector, $h(\omega) \in \mathbb{R}^{m_2}$ is the righthand side, $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$ is the technology matrix, and $W \in \mathbb{R}^{m_2 \times n_2}$ is the

fixed recourse matrix. We also assume that $T(\omega) : \Omega \mapsto \mathbb{R}^{m_2 \times n_1}$, $h(\omega) : \Omega \mapsto \mathbb{R}^{m_2}$ and $q(\omega) : \Omega \mapsto \mathbb{R}^{n_2}$ are measurable mappings defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. The function $\mathcal{Q}_E(x)$ is the expected recourse function, where ω is a realization of a multivariate random variable $\tilde{\omega}$, and $\mathbb{E}_\omega$ denotes the mathematical expectation operator satisfying $\mathbb{E}_\omega[\,|\Phi(q(\omega), h(\omega) - T(\omega)x), \omega|\,] < \infty$ for all $x \in \{Ax \geq b, x \in X\}$. This requirement is the relatively recourse assumption. The set Y is the restriction for the second-stage variables. Finally, the vector $u \in \mathbb{Z}^{n_2}$ defines the upper bound for the second-stage variables. Subproblem (1.3) is generally referred as the *scenario* problem.

A scenario ω will have a corresponding probability p_ω, and $\sum_{\omega \in \Omega} p_\omega = 1$. If the underlying probability distribution of $\tilde{\omega}$ is discrete with a finite number of realizations (scenarios), then the formulation (1.1) - (1.3) can also be written in extensive form, also known as deterministic equivalent problem (DEP) as follows:

$$\begin{aligned} \text{DEP} : \text{Min } & c^\top x + \sum_{\omega \in \Omega} p_\omega q(\omega)^\top y(\omega) \\ \text{s.t. } & Ax \geq b \\ & T(\omega)x + Wy(\omega) \leq h(\omega) \\ & x \in X, y(\omega) \in Y. \end{aligned} \tag{1.4}$$

In formulation (1.4), $y(\omega)$ is the recourse decision variable vector, and the other dimensions are as stated before.

Even for a reasonable number of scenarios in Ω, DEP is a large scale MIP. With integer variables in both first and second stages, a moderate sized DEP is difficult to solve using a direct solver like CPLEX [40]. This makes a decomposition approach a necessity for most practical sized problems. In SIP2, the type of decision variables (continuous, binary, integer) and in which stage they appear greatly influences

algorithm design. The complexity of the solution methodologies depends on the definitions of the sets X and Y. When $X \in \mathbb{R}^+$ and $Y \in \mathbb{R}^+$ for a given $\omega \in \Omega$, the recourse function $\Phi(\rho, \tau, \omega)$ is a well-behaved piecewise linear and convex function of x. Thus, Benders' decomposition [16] is applicable in this case [108] and the L-shaped method [104] can be used to solve the problems. Assuming fixed recourse (i.e, the recourse matrix W is independent of the scenario ω), the value function of $\Phi(\rho, \tau, \omega)$ will be a piecewise linear function in x. Hence, the L-shaped method works by approximating the linear functions from the subproblems by constructing optimality cuts in the first-stage based on the dual values from the subproblems. However, when $X \in \mathbb{Z}^+$ and $Y \in \mathbb{Z}^+$, the linear approximation procedure by L-shaped method is not viable, as the value function is discontinuous, and more precisely the function is lower semicontinuous [19]. Also, the function is non-convex and sub-additive [87].

General efficient methods like the L-shaped method or Dantzig-Wolfe decomposition methods are not applicable to SIP. With the integrality restrictions on the second-stage decision variables, dual values from the second-stage program cannot be used to approximate the value function of the second-stage program. Hence, new algorithms or extensions of the L-shaped method are required to handle integer variables in the second or in both of the stages.

1.1.2 Fenchel Cutting Planes

Within a generic framework like L-shaped method, the approach to solve a SIP will be to approximate the value function of the subproblems by solving them as relaxed linear programs (LPs). Based on the dual solution of the relaxed subproblems, the optimality cuts will be constructed to approximate the second-stage value function. However, in the event of second-stage problems giving non-integer solutions, suitable separation problems are constructed to remove the non-integer solution from further iterations. Construction of valid inequalities via solving sep-

aration problems is an important aspect of this research, where we try to exploit the special structure of the subproblem. In this research, Fenchel cuts will be used as valid inequalities in the second-stage relaxed subproblems. Under certain normalizations, Fenchel cuts are the deeper cuts which guarantee to give facets for the polyhedron. However, for getting the Fenchel cuts, the cut generation procedure involves solving the subproblem as an integer program (IP). In the current literature, Fenchel cuts are used only for subproblems with binary variables. In this research, theory and methodology are devised to generate Fenchel cuts for subproblems with general integer variables.

1.1.3 Stochastic Auto-Carrier Loading Problem

The last mile delivery of cars, trucks and vans to dealerships is one of the most expensive logistics part of vehicle distribution. The final leg of the delivery of vehicles to the dealer lot is invariably carried out by a special type of trucks called auto-carriers. These carriers are specialized trucks with a tractor and a trailer, with upper and lower loading ramps (platforms) as shown in Figure 1.1. An auto-carrier can have anywhere between one and four ramps in each loading level, and a typical version used in delivery of new vehicles has about nine loading ramps. Although vehicle delivery is a specialized transportation problem, this is a very important part of the trucking industry. In 2013 alone, 15.6 million new vehicles were sold in the US [57]. A conservative estimate of $100 per vehicle for the last mile delivery (in general the average cost for the final leg is between $250 and $400) puts the expected expenditure in auto transportation in excess of $15 billion in 2013. The American Trucking Association accords this industry with a special status in their group called Automotive Carrier Conference (ACC).

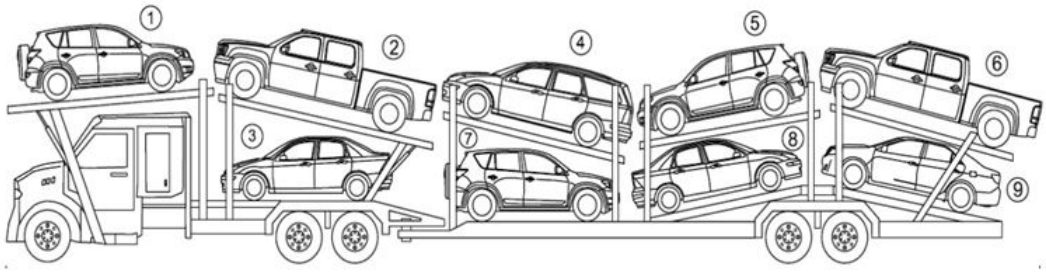

Figure 1.1: An auto-carrier with nine loading ramps

The auto-manufacturers ship their finished vehicle to a distribution center (DC) through ships and by train. From the staging areas in a DC, the vehicles are shipped to auto dealerships through auto-carriers. Typically a DC receives about 20,000 to 40,000 vehicles a month, and they schedule the delivery to the auto dealerships on a weekly basis. The main objective of a logistics company operating a DC is to reduce the number of trips they make each week to deliver the vehicles. Each trip is subject to loading constraints such as height, length and shape of the vehicle, and also restrictions on maximum length, height and weight of cargo set by local and government organizations such as the U.S. Department of Transportation. Every year, new vehicle types with various dimensions and weights are introduced and this complicates the already difficult loading problem. Table 1.1 shows the wide range of vehicle types, and their dimensions sold by some of the auto-manufacturers in the US. As seen in the table, the heaviest vehicle type is at least three times as heavy as the lightest one, and it requires two loading ramps to transport it. The large quantities of new vehicles being sold each year, the rising cost of fuel, and increasing variety of vehicle types have made this problem very difficult to solve.

	Honda			Toyota			Ford		
	Ridgeline	Accord	Fit	Tundra	Camry	Yaris	F350	Focus	Fiesta
	Truck	Sedan	HB	Truck	Sedan	HB	Truck	Sedan	HB
Weight (lbs)	6,050	3,216	2,496	6,800	3,190	2,295	9,900	2,097	3,620
Length (inches)	207	195	162	229	189	154	233	179	160
Height (inches)	70	58	60	76	58	59	77	58	58
Width (inches)	78	73	67	80	72	67	80	72	68

Table 1.1: Sample vehicle types dimensions (HB- Hatchback)

Many approaches in current literature use approximation and rule of thumb for auto-carrier loading process. This research presents a tactical planning regarding the number and type of auto-carriers required based on uncertainty in demand for vehicle types. The tactical planning includes an auto-carrier loading problem, which considers actual dimensions of the vehicles, regulations on total height of the auto-carriers, and maximum weight of the axles, and safety requirements. The problem is modeled as a two-stage SIP, and computational experiments using real data are performed.

1.2 Research Contributions

Research contributions include devising of theory and algorithms towards solving SIP2 models based on Fenchel cutting planes. The existing approach of using Fenchel cutting planes for stochastic programs with binary variables exploits the special structure in binary problems. Unfortunately, such direct exploitation is not applicable for SIP with general integer variables. We develop a new algorithm to effectively address the stochastic programs with general integer variables for problems

with special structure. The proposed methodology is tested on randomly generated instances from the literature. The specific research contributions (RC) are as follows:

- **RC1:** *Theory and algorithm for SIP2 with general integer variables in the second-stage based on Fenchel cutting planes.* The methods for generating Fenchel cutting planes require to solve IPs, which may be difficult in general. Therefore a new algorithm is developed to overcome this challenge.

- **RC2:** *Investigation of normalization for the cut co-efficients in the Fenchel cutting planes for SIP2.* The normalization provides the alignment of the Fenchel cuts, hence the *norms* give the ability for the Fenchel cuts to separate a relaxed solution from the solution space. We investigate the usefulness of different norms, both in terms of computation time and ability to recover integer solutions.

- **RC3:** *Implementation of algorithms from RC1 and RC2.* Test the implementations with randomly generated instances and standard test instances from the literature.

- **RC4:** *Computational study for Fenchel decomposition algorithm SIP2 with special structure.* Perform a computational study based on randomly generated instances from the literature. Also, propose and implement techniques to improve the run time for computational experiments.

- **RC5:** *Formulation for SACP.* Generate the instances using real data, and perform computational experiments using the implementation of the ST-FD algorithm.

This research provides a new methodology to solve SIP2 with general integer variables in second-stage. Also, the proposed methods will give a new direction for

future research extensions. The computational study using standard and stochastic auto-carrier instances will provide insights on the practical applicability and limitations of the proposed methodology. In a stochastic setup, other general applications with special structure using general integer variables can also benefit directly from using the proposed methodology.

1.3 Dissertation Organization

This dissertation is organized as follows: Literature review for SIP, Fenchel cutting planes, and stochastic auto-carrier loading problem are provided in Section 2. Section 3 details the ST-FD for SIP with binary second-stage recourse problems. Section 4 presents the mathematical model and computational results for SACP. Theory, methodology and computational studies are presented for ST-FD for IP, and SIP with second-stage general integer variables in Section 5. Finally, Section 6 summarizes the contributions of this dissertation, and presents the avenues for future research.

2. LITERATURE REVIEW

This dissertation focuses on using Fenchel cutting planes for SIPs with special structure. This section reviews theory necessary for later sections. It also summarizes current state-of-the art approaches for solving SIPs, generating Fenchel cutting planes, and modeling auto-carrier loading problem.

2.1 Stochastic Mixed-Integer Programming

Mathematical programming deals with optimization problems to seek a best solution from the given alternatives. We consider problems with linear constraints and objective function. When all the decision variables of a mathematical program are allowed to take continuous values, and the problem data are known precisely, then the problem is a LP ([42], [33] and [14]). If some or all of the decision variables are restricted to take discrete values then the problem is an IP. Some of the good references for IP are [109], [86] and [71]. A LP with uncertain parameters is a stochastic LP ([41], [58] and [91]). When the data are unknown for the parameters, and some of the variables have integrality restrictions, then it is a SIP. A good introduction to stochastic programming can be found at [18], [58] and [79], and for surveys on SIP, the reader can refer to [65], [98], [59], [94] and [90]. To aid the discussion on the literature, we use a classification scheme to represent the various classes of two-stage SIPs. This scheme is based on the variable restrictions represented by X and Y. We use the sets F and S to denote the first and second stages of the stochastic program, respectively, and B, C, D for binary, continuous and discrete variables, respectively. For example, the class of problems considered in this literature have $F = \{B, C, D\}$ and $S = \{B, C, D\}$, i.e., binary, continuous, and general integer variables may appear in both stages.

The integer L-shaped algorithm [64] is a pioneering methodology for solving SIP. The algorithm solves problems with $F = \{B\}; S = \{B, C, D\}$. In the algorithm, the second-stage objective function values are used to construct cuts for the first-stage. However, solving second-stage MIPs to optimality is a challenging task for large scale problems. The work in [27] and [28] uses Lagrangian dual and BAB for $F = \{B, C, D\}; S = \{B, C, D\}$. In the algorithm, a non-anticipativity constraint is relaxed, and the corresponding Lagrangian dual is solved to get the Lagrange multipliers. Furthermore, a BAB scheme is used for non-integer solutions. This is a pioneering work to suggest a BAB scheme for solving SIPs. However, the implementation needs very careful devising for choosing appropriate Lagrangian multipliers.

The IP duality in L-shaped framework and Gomory cuts are used in [30] for problems of type $F = \{B, C, D\}; S = \{B, D\}$. The subproblems are solved to optimality for a given $x \in X$ in a cutting plane algorithm using Gomory cuts. Furthermore, the optimality cuts for master problem are linear functions with integer variables. This is computationally challenging due to the presence of integer variables, and they are required to solve to optimality in the first-stage. In a closely related work [49], a decomposition algorithm for two-stage stochastic programs with binary first-stage and integer second-stage variables is proposed. The second-stage cost function, technology and recourse matrices are allowed to be random. Since this decomposition method exploits the property that the first-stage variables are binary to derive valid cuts for the second-stage, it is not directly extendable to SIPs with general integer variables in the first-stage.

Cutting plane methods that can partially approximate the second-stage problems within the L-shaped method have been proposed for SIPs with integer variables in the second-stage. In the literature, such methods for two-stage problems have been

almost exclusively restricted to disjunctive cut-generation schemes. In [29], lift-and-project cutting planes approach based on the ideas from [12] is used to solve problems with $F = \{B, C\}; S = \{B, C\}$. Cutting planes are used to separate non-integer solutions from the relaxed LPs. In [92], for problems with $F = \{B\}; S = \{B, C\}$, disjunctive cuts are developed for the second-stage. Furthermore, the cuts can be made valid across all the other scenarios by calculating an appropriate righthand side function. This has a computational advantage as the cuts need not be generated independently for each scenario. Additionally, the value function is sequentially approximated using linear cutting planes in the first-stage. The work in [95] and [96] uses the framework of reformulation linearization technique. The algorithm in [92] is extended in [93] for problems with $S = \{B, C, D\}$. A combination of disjunctive programming and a partial BAB tree is used in the second-stage. Computational studies are reported in [[74] and [75]], [110] for the algorithms of [92] and [93], respectively. Furthermore, the algorithm in [92] is extended in [72] for problems with random recourse and fixed technology matrices. This ensures that the cuts for the second-stage have common coefficients. Another approach using reformulation linearization technique cuts is introduced in [97] for problems with $F = \{B, C\}; S = \{B, C\}$.

Method	Authors	FS-Binary	FS-Integer	SS-Binary	SS-Integer
1. Integer L-shaped	Laporte and Louveaux	X		X	X
2. Parametric Gomory Cuts	Gade et al.	X			X
3. Cutting Plane Approach	Caroe and Tind	X		X	X
4. Disjunctive Decomposition	Sen and Higle	X		X	
5. Disjunctive Decomposition and Branch-and-Cut	Sen and Sherali	X		X	X
6. Enumeration Algorithm	Schultz et al.			X	X
7. Finite branch-and-bound Algorithm	Ahmed et al.	X		X	X
8. Fenchel Decomposition	Ntaimo	X		X	
9. Dual Decomposition	Caroe and Schultz	X	X	X	X
10. Super-additive Dual Approach	Kong et al.	X	X	X	X
11. L-shaped Decomposition	Caroe and Tind	X	X	X	X

Table 2.1: Literature review

Continuous variables in the first-stage $F = \{C\}$ present more difficult problems, as their solutions dictate the orientation of the cutting planes in the second-stage. An enumeration algorithm is developed in [88], and the algorithm is based on polynomial ideal theory (Gröbner bases) for problems with $F = \{C\}; S = \{B, D\}$. Unfortunately, Gröbner bases are notoriously difficult to compute [68]. A finite BAB algorithm is developed in [4] for problems with $F = \{B, C\}; S = \{B, D\}$ and fixed technology matrix. A formulation to obtain value functions in both the stages is proposed in [62], and problems with $F = \{B, D\}; S = \{B, D\}$ are studied. Furthermore, a BAB framework in combination with a level-set approach is used as solution methodology. Table 2.1 briefly lists the contributions from the literature.

2.2 Fenchel Cutting Planes

Several types of cutting planes have been proposed in IP. The types of cutting planes include split cuts ([36], [39] and [7]), intersection cuts ([8], [9] and [35]), disjunctive cuts ([11], [12] and [38]) and Fenchel cuts. For a relatively recent review on cutting planes, the reader is referred to [37]. This research work is based on a type of valid inequalities called Fenchel cutting planes. Fenchel cutting planes are a class of deep cutting planes derived using Fenchel duality in convexity theory [81], and they take advantage of the maximum separation/minimum distance duality. Fenchel cuts are suggested in [24], and a number of characteristics are derived in [23], [25] and [26]. The most important results from [24], [25] and [26] are that Fenchel cutting planes are facet defining under certain conditions, and the use of Fenchel cuts in a cutting plane approach yields an algorithm with finite convergence. The work also highlights the fact that generating a Fenchel cut for binary programs is computationally expensive in general; therefore, problems with special structure are desirable to achieve faster convergence. Computational experiments demonstrating the effectiveness of Fenchel cuts are presented for knapsack polyhedra in [22] and for pure binary problems in [25]. Fenchel cuts are derived for two-stage SIPs under a stage-wise decomposition setting in [73]. In [73], considering x as first-stage decision variable, and y as second-stage decision variable, two forms of cuts called Fenchel decomposition (FD) cuts are derived: one based on the (x, y) space, and the other derived based on the y space, and then lifted ([10] and [13]) to the (x, y) space. However, a direct extension of the current methodology to general integer variables may not be scalable, since solving the subproblems as IPs may be computationally expensive. Also, appropriate normalization should be used in the cut generation process. In this research, we study the effect of different normalizations for the cut-generation procedure.

After the pioneering work in [24], only a few have adopted Fenchel cuts in their work. In [83], Fenchel cuts are used to improve the bounds obtained from MIPs using Lagrangian relaxation. More recently, Fenchel cuts are used to solve deterministic capacitated facility location problems [80]. This work compares Fenchel cuts to Lagrangian cuts in finding good relaxation bounds for their problem. In [20], Fenchel cutting planes are used for finding p median nodes in a graph using a cut and branch approach.

2.3 Stochastic Auto-Carrier Loading Problem

To the best of our knowledge this is the first time SIP has been considered in auto-carrier loading problem. In this section, we survey the literature related to auto-carrier loading problem in deterministic setup, and we present on how our approach and features differ from the current literature. Though loading problems are studied in combination with routing, we restrict ourselves to the literature for loading problems. The literature is considered only for the auto-carriers used for loading vehicles for delivery as the loading problems are available in 2-dimensional and 3-dimensional spaces in other areas of applications ([48], [56]). One such example is [55], where the authors have used meta-heuristics for routing with two-dimensional and three-dimensional loading constraints.

The pioneering work on auto-carrier loading problem is presented in [2]. This work is extended in [3]. A quadratic assignment model is presented for auto-carrier loading problem. Vehicle-slot and pairwise incompatibilities are considered for vehicles in the adjacent slots. Furthermore, a BAB is presented to solve the quadratic assignment model.

The loading and routing problem is formulated as an IP model in [99]. This work also shows that the problem is NP-hard. Furthermore, a heuristic that considers loading, routing and vehicle selection is proposed. The use of vehicle dimensions

is substituted by a parameter proposed by transportation companies. Based on the parameters from the companies, the total length of the loaded vehicles is constrained.

In [69], loading process is simplified so that vehicle dimensions are not considered. The loads are considered in two flat loads with an assumption that any two vehicles can be assigned to the two flat beds. The work proposes a construction heuristics, and presents limited computational results.

A quadratic assignment model very similar to the model suggested in [3] is used in [32]. Furthermore, computational results are presented. The reported results show that the quadratic assignment model produces an exact optimal solution. Similar to the work in [3], compatibility indicators are used for vehicle-slot and pairwise incompatibilities for vehicles in adjacent slots.

An iterated local search approach for both routing and loading of auto-carriers is presented in [44]. Instead of vehicle dimensions, reduction coefficients are used, which are based on auto-carrier type, vehicle type, and slot used for loading. The reduction coefficients are proposed by logistics companies. The reduction coefficients are used to construct restrictions on length of a platform used for loading the vehicles. For each generated route, loading constraints are checked for feasibility. The work also presents extensive computational results.

In all of the works mentioned above, the assignment of vehicles to the slots are constrained based on a parameter value obtained from logistics or transportation companies. These parameters are assumed based on the experience of loaders or loading rules generally maintained in the logistics companies. However, this approach can be cumbersome whenever there are large number of vehicles for loading or vehicles are new to the market. Whenever there are new vehicle types introduced in the market for loading, then generating a reasonable representative parameter for a vehicle is a challenge. Also, in the US we have government regulations on the

maximum weight for each of the axles of an auto-carrier [103]. An approximate estimation for the weight of an auto-carrier's axle based on the position of vehicle types in the slots is not trivial to estimate. These challenges motivated us to consider the actual dimensions of vehicle types in our mathematical model. Our model considers actual physical dimensions for the vehicle types to estimate the overall length and height, and weight on each axle of the auto-carriers based on the vehicle types loaded in the respective slots.

We presented literature review for SIP and SACP. We reviewed the decomposition approaches available for SIP. SACP is an application of SIP. Hence, a scalable algorithm for SIP will be useful to solve the instances of SACP.

3. FENCHEL DECOMPOSITION FOR STOCHASTIC MIXED 0-1 PROGRAMS WITH SPECIAL STRUCTURE

3.1 Introduction

Decomposition approaches for SIP traditionally fall under one of two categories: *stage-wise* decomposition or *scenario-wise* decomposition. Stage-wise decomposition strategies are usually based on Benders' decomposition ([16], [104]). Scenario-wise decomposition involves variable splitting on the first-stage decision variables to create nonanticipativity constraints to enforce the first-stage solution to be the same for all scenarios ([28], [82] and [65]). Applications of scenario-wise decomposition can be found in [46], [53], [106] and [70]. In this research, we consider the stage-wise decomposition approach for solving SIP2.

3.2 Fenchel Decomposition Cut Generation

Stage-wise Fenchel decomposition (ST-FD) adopts the Benders' decomposition setting with x as the first-stage decision variable in the master problem, and y as the second-stage decision variable in the subproblem. In SIP2, instead of working with the IP subproblem directly, ST-FD seeks to find the optimal solution via a cutting plane approach on a partial LP-relaxation of SIP2 where only the subproblems are relaxed. Fenchel cuts are sequentially generated to recover (at least partially) the convex hull of integer points for each scenario subproblem feasible set. If a subproblem LP has a non-integer solution, a Fenchel cut is generated and added to cut off the fractional solution. Fenchel cuts are capable of recovering faces of the convex hull of binary programs, which is the special structure for SIP2. The goal is to construct the convex hull of integer points in the neighborhood of the optimal solution so that by solving subproblems LPs with enough Fenchel cuts added, we

can find the optimal solution without having to use BAB to guarantee optimality.

At a given iteration k of the ST-FD cutting plane algorithm, the master problem takes the following form:

$$
\begin{aligned}
z^k = \text{Min } & c^\top x + \theta \\
\text{s.t. } & Ax \geq b \\
& (\eta^t)^\top x + \theta \geq \gamma^t, \ t \in 1, ..., k \qquad (3.1a) \\
& x \in \{0, 1\}.
\end{aligned}
$$

Constraints (3.1a) are the *optimality* cuts, which are computed based on the optimal dual solution of all the subproblems. Optimality cuts approximate the value function of the second-stage subproblems. For a first-stage solution x^k from the master problem (3.1), the subproblem for each scenario $\omega \in \Omega$, denoted SP(ω), is given as follows:

$$
\begin{aligned}
\text{SP}(\omega) : \Phi^k_{LP}(\rho, \tau, \omega) = \text{Min } & \rho^\top y(\omega) \\
\text{s.t. } & Wy(\omega) \leq \tau \\
& \beta^t(\omega)^\top y(\omega) \leq g(\omega, \beta^t(\omega)), t \in \Theta(\omega) \qquad (3.2a) \\
& y(\omega) \geq 0.
\end{aligned}
$$

Constraints (3.2a) are the Fenchel cuts, and $\Theta(\omega)$ is the index set for algorithm iterations at which a Fenchel cut is generated for each $\omega \in \Omega$. Next, we describe how these cuts are generated.

We start with the preliminaries for Fenchel cut generation (FCG) for SIP2. Consider a methodology for FCG to solve problems of type (1.3), especially when

second-stage variables have integer restrictions. We will restrict our discussion to problems of type (1.3) in this section. Also in this section we consider the set Y to have only binary restrictions for some of its components. For ease of exposition, we will ignore the parameter ω in the following derivation. We will start with the definitions.

Let the objective function value of the LP-relaxation to (1.3) be given as,

$$\Phi^{LP}(\rho, \tau) = \text{Min}\{\rho^\top y : Wy \leq \tau, 0 \leq y \leq u, y \in \mathbb{R}^{n_2}\}. \tag{3.3}$$

Feasible set for the problem (3.3) be given as:

$$F^{LP} = \{y : Wy \leq \tau, 0 \leq y \leq u, y \in \mathbb{R}^{n_2}\}. \tag{3.4}$$

Let the feasible set for the problem (1.3) be given as:

$$F^{IP} = \{y \in F^{LP} : y \in Y\}. \tag{3.5}$$

The convex hull of feasible integer points for F^{IP} is represented as $C(F^{IP})$.

Let $\hat{y} \in F^{LP}$ and $\hat{y} \notin C(F^{IP})$ be given. Then the separation problem is to find a valid inequality $\beta^\top y \leq \beta_0$ for the problem (1.3) such that $\beta^\top \hat{y} > \beta_0$, where β and β_0 are the vectors with appropriate dimensions.

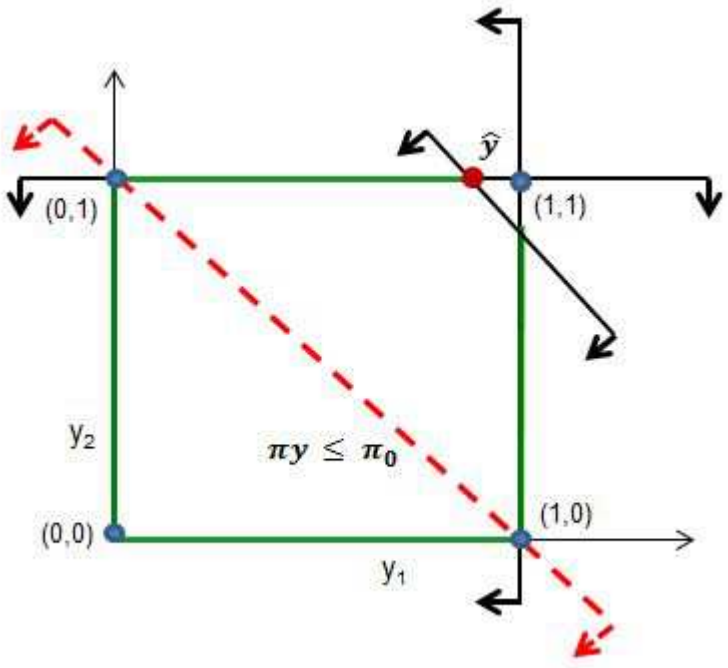

Figure 3.1: Separation problem for binary variables

For illustration, let $\hat{y} \in F^{LP}$ be the optimal solution to the LP-relaxation (3.3) as depicted in Figure 3.1. Using a separation problem, we generate a cut $\pi^\top y \leq \pi_0$ such that $\pi^\top \hat{y} > \pi_0$.

In FCG, the objective is to derive such valid inequalities called Fenchel cuts. We start devising the FCG procedure for IPs of type (1.3) with the following theorem.

THEOREM 3.1. *Let $\hat{y} \in F^{LP}$ be given. Define $g(\beta) = Max\ \{\beta^\top y \mid y \in C(F^{IP})\}$ and let $\delta(\beta) = \beta^\top \hat{y} - g(\beta)$. Then there exists a vector β for which $\delta(\beta) > 0$ if and only if $\hat{y} \notin C(F^{IP})$.*

Theorem 3.1 is based on generating a Fenchel cut for IPs as proposed in [24], so we omit the proof. The result of Theorem 3.1 is that given a $\hat{y} \in F^{LP}$, if $\delta(\beta) > 0$, then there exists a valid inequality that will separate $\hat{y}$ from the convex hull of

integer feasible points $C(F^{IP})$. The inequality derived in such a way is of the form $\beta^\top y \leq g(\beta)$ and is called a Fenchel cut. When generating a Fenchel cut, it is desirable to maximize the distance between $\hat{y}$ and the hyperplane $\beta^\top y \leq g(\beta)$ without cutting off any integer points in $C(F^{IP})$. This requires maximizing $\delta(\beta)$.

While any β vector that gives a positive $\delta(\beta)$ will provide a valid Fenchel cut, finding such a β vector requires a search of the β space constrained to a convex set Π^β. Maximizing the function $\delta(\beta)$ provides such a search and returns the deepest cutting plane possible. This maximization provides a Fenchel cut, and separates $\hat{y}$ from $C(F^{IP})$. To generate a Fenchel cut, a solution to the following optimization problem is required:

$$\delta = \underset{\beta \in \Pi^\beta}{\text{Max}} \left\{ \beta^\top \hat{y} - g(\beta) \right\}. \tag{3.6}$$

where the maximization is done over a domain Π^β and

$$g(\beta) = \underset{y \in C(F^{IP})}{\text{Max}} \left\{ \beta^\top y \right\}. \tag{3.7}$$

Once found, the Fenchel cut separating the non-integer point $\hat{y}$ from $C(F^{IP})$ is:

$$\beta^\top y \leq g(\beta). \tag{3.8}$$

Note that the cut (3.8) must pass through a point in $C(F^{IP})$ (found in (3.7)), and could be potentially a facet of $C(F^{IP})$.

Solving (3.6) is not a trivial task. For this work, a generalized programming method based on Benders' decomposition is used. The method uses a master problem

(given below) to construct a linear approximation of the subproblem space while the subproblem returns feasible integer points from F^{IP}.

$$\begin{aligned} \delta^{(t)} = \ & \underset{\beta \in \Pi^{\beta}}{\text{Max}} \ \ \theta \\ & \text{s.t.} \quad -\theta + (\hat{y} - y^{(\nu)})^{\top} \beta^{(\nu)} \geq 0, \ \nu = 1, \cdots, t. \end{aligned} \tag{3.9}$$

Let t be the number of iteration, then $\beta^{(t)}$ represents the value of β in iteration t, and let β_i represent a component of β. Given an optimal solution $(\theta^{(t)}, \beta^{(t)})$ to (3.9) at iteration t, where $y^{(t)}$ is the optimal solution to the following subproblem:

$$\begin{aligned} g(\beta^{(t)}) = \text{Max} \ & {\beta^{(t)}}^{\top} y \\ & \text{s.t.} \ y \in F^{IP}. \end{aligned} \tag{3.10}$$

It should be noted that solving (3.10) is generally difficult. A straight forward approach will be to solve (3.10) as MIP, however this may not be trivial for larger instances. Adopting a Benders' decomposition framework, a method for generating Fenchel cuts is stated in Algorithm 1.

In step [1] we initialize the parameters for the algorithm. Initially, each component of $\beta^{(0)}$ is set to 0.5. Since problem (3.6) has to be solved many times to generate Fenchel cuts, a linearly constrained domain for Π^{β} such as the L^1 unit sphere,

$$\Pi^{\beta} = \{\beta \in R_{+}^{n2} : 0 \leq \beta \leq 1, \sum \beta \leq 1\}$$

provides a better choice in terms of solution time. However, an L^2 unit sphere,

$$\Pi^{\beta} = \{\beta \in R_{+}^{n2} : 0 \leq \beta \leq 1, \sum_{i} \beta_i^2 \leq 1\}$$

can also be used. Step [2] uses $\beta^{(0)}$ as co-efficients, then subproblem (3.7) is solved, and the corresponding objective value stored. It should be noted that problem (3.7) is solved as an IP, and this solution $y^{(t)}$ is integral. The bounds and incumbent solutions are updated in step [2]. Based on the solution $y^{(t)}$ from (3.7), the cut is added to master problem (3.9). In step [3], master problem (3.9) is solved and the termination condition is checked. Based on the termination condition, the algorithm either stops or continues.

3.3 Stage-Wise Fenchel Decomposition Algorithm

We extend the algorithm by using L-shaped algorithm to solve the LP-relaxation of SIP2, and then carefully choosing a starting solution for ST-FD algorithm to yield better results. This version of ST-FD algorithm is formally stated in Algorithm 2.

The ST-FD algorithm starts by initializing data in step [1] and getting an initial solution by solving the LP-relaxation of SIP2 in step [2]. If the initial solution satisfies the integrality restrictions for all subproblems in step [3], i.e., $x \in X$ and $y(\omega) \in Y$, $\forall\, \omega \in \Omega$, then the solution is optimal, and the algorithm stops. Otherwise, the algorithm continues by calculating and storing the optimality cut coefficients for all subproblems with an integer solution in step [4].

Algorithm 1 Fenchel Cut Generation Procedure (FCG)

[1] **Initialization:** Set $t \leftarrow 0, \epsilon > 0, LB \leftarrow -\infty, UB \leftarrow \infty$, and get an initial point $\beta^{(0)} \in \Pi^{\beta}$.

[2] **Solve subproblem:**

Use $\beta^{(t)}$ to solve (3.7) and get solution $y^{(t)}$ and the corresponding objective value $g(\beta^{(t)})$.

Compute lower bound:

Let $d^{(t)} \leftarrow (\hat{y} - y^{(t)})^{\top}\beta^{(t)}$.

Set $l^{(t+1)} \leftarrow max\{d^{(t)}, l^{(t)}\}$.

if $l^{(t+1)}$ is updated **then**

Update incumbent solution:

Set $\mu \leftarrow d^{(t)}$ and $(\beta^*, g(\beta^*)) \leftarrow (\beta^{(t)}, g(\beta^{(t)}))$.

end if

Use $\hat{y}$ and solution $y^{(t)}$ from (3.7) to form and add constraint to the problem (3.9).

[3] Solve problem (3.9) to get an optimal solution $(\theta^{(t)}, \beta^{(t)})$.

Compute upper bound:

Set $u^{(t+1)} \leftarrow min\{\theta^{(t)}, u^{(t)}\}$.

if $u^{(t+1)} - l^{(t+1)} \leq \epsilon'$ **then**

The incumbent solution is optimal.

Stop.

else

Set $t \leftarrow t + 1$ and go to [2].

end if

Algorithm 2 Stage-Wise Fenchel Decomposition (ST-FD) Algorithm

[1] **Initialization:** set $k \leftarrow 0, \epsilon > 0, LB \leftarrow -\infty$ and $UB \leftarrow \infty$.

[2] **Get initial solution:** Solve problem (3.1-3.2) using the L-shaped algorithm to get solution $(\hat{x}^0, \hat{y}^0(\omega))$, objective function value $\varphi^0 = \sum_{\omega \in \Omega} p_\omega \Phi^k_{LP}(\rho, \tau, \omega)$, and dual solutions $\hat{\pi}^k(\omega)$ for each $\omega \in \Omega$.

[3] **Check solution integrality:**

if $\hat{y}^k(\omega) \in Y$ **then**

Report $(\hat{x}^k, \hat{y}^k(\omega))$ as optimal.

Stop.

end if

[4] **Calculate and store optimality cuts coefficients for scenarios with integer solution**

for $\omega \in \Omega$ **do**

if $\hat{y}(\omega)^k \in Y$ **then**

Calculate and store optimality cut coefficients $\eta(\omega)^k \leftarrow \hat{\pi}(\omega)^{k\top} T(\omega)$ and $\gamma(\omega)^k \leftarrow \hat{\pi}(\omega)^{k\top} h(\omega)$.

end if

end for

[5] **Fenchel cuts and optimality cuts generation:**

for $\omega \in \Omega$ **do**

if $\hat{y}(\omega)^k \notin Y$ **then**

Compute scenario Fenchel cut coefficients: Run FCG to get $\beta(\omega)^k$ and $g(\omega, \beta(\omega)^k)$.

Add the cut $\beta(\omega)^{k\top} y(\omega) \leq g(\omega, \beta(\omega)^k)$ to subproblem (3.2).

Solve the updated subproblem SP(ω) and get updated subproblem dual solution $\hat{\pi}(\omega)^k$.

Update optimality cut coefficients $\eta^k \leftarrow \eta^k + p_\omega \cdot (\hat{\pi}(\omega)^k)^\top T(\omega)$ and $\gamma^k \leftarrow \gamma^k + p_\omega \cdot (\hat{\pi}(\omega)^k)^\top h(\omega)$.

end if

end for

[6] **Add optimality cut** $\eta^k x + \theta \geq \gamma^k$ to master problem (3.1) and update iterator: set $k \leftarrow k + 1$.

[7] **Solve master problem** (3.1) to get a new first-stage solution $\hat{x}^k$ and objective value z^k.

[8] **Update lower bound:** Set $LB \leftarrow max\{LB, z^k\}$

[9] **ϵ-optimality check:**

if $|UB - LB| \leq \epsilon |LB|$ **then**

Go to step [14].

end if

[10] **Solve subproblems:**

for $\omega \in \Omega$ **do**

Solve subproblem (3.2) to get updated subproblem solution $\hat{y}(\omega)^k$, optimal value $\Phi^k_{LP}(\rho, \tau, \omega)$ and dual solution $\hat{\pi}(\omega)^k$.

if $\hat{y}(\omega)^k \in Y$ **then**

Calculate and store optimality cut coefficients $\eta(\omega)^k \leftarrow \hat{\pi}(\omega)^{k\top} T(\omega)$ and $\gamma(\omega)^k \leftarrow \hat{\pi}(\omega)^{k\top} h(\omega)$.

end if

end for

[11] **Subproblem solutions integrality check:**

for $\omega \in \Omega$ **do**

if $y(\omega)^k \notin Y$ **then**

Go to step [5].

end if

end for

[12] **Update solution and bound information:**

Update incumbent solution: $x^* \leftarrow x^k$.

Update upper bound: $UB \leftarrow min\{UB, c^\top x^k + \sum_{\omega \in \Omega} p_\omega \Phi^k_{LP}(\rho, \tau, \omega)\}$.

[13] *ϵ-optimality check:*

if $|UB - LB| > \epsilon |LB|$ **then**

Go to step [6].

end if

[14] **Declare** x^* ϵ-optimal.

Stop.

For subproblems with a solution that does not satisfy the integrality requirements, Fenchel cut coefficients $\beta^k(\omega)$, and the righthand side $g(\omega, \beta^k(\omega))$ are computed for the iteration k in step [5]. A Fenchel cut is added to subproblem SP(ω). Next, the dual solution obtained by solving the subproblem is used to generate the optimality cut coefficients. Once all subproblems have been solved at a given iteration, the optimality cut is added to the master problem in step [6]. The iteration counter k is increased, and the master problem is solved again in step [7] to get an updated first-stage solution and objective value.

The lower bound LB is updated in step [8]. The gap between the lower bound LB and the upper bound UB is verified in step [9]. If this gap is small enough, then the incumbent solution is declared ϵ-optimal in step [14], and then the algorithm terminates. Otherwise, all the subproblems are solved again, and optimality cut coefficients are updated for subproblems with an integer solution in step [10]. The integrality of subproblem solutions is verified in step [11]: if a subproblem's solution is not integral, the algorithm returns to step [5], to add Fenchel cuts to the subproblems with a non-integer solution, and compute their optimality cut coefficients. Otherwise,

the incumbent solution x^* and the upper bound UB are updated in step [12]. The optimality check is done again in step [13]: if it is satisfied, the incumbent solution is ϵ-optimal, and the algorithm is terminated. Otherwise, the algorithm returns to step [6], and the optimality cut is added to the master problem, and its solved again. The algorithm is continued until the termination condition is satisfied.

3.4 Computational Study

We implemented the ST-FD algorithm, and performed a computational study to demonstrate the performance of the algorithm on randomly generated instances from the literature. The algorithm was implemented in C++ using CPLEX 12.1 Callable Library [40] in Microsoft Visual Studio 2010. Computations were performed on an ACPI x64 computer with an Intel®Xeon®Processor E5620 (2.4GHz) and 12GB RAM. CPLEX MIP and LP solvers were used to optimize the master problem and subproblems. The instances were run to optimality or stopped when a CPU time limit of 3600 seconds (1 hour) was reached. As a benchmark, the deterministic equivalent problem (DEP) for each test instance was created and solved using the CPLEX MIP solver. Computational experiments were conducted on four sets of test instances from stochastic multidimensional knapsack problems with special structure. Next, we describe the formulation and test sets, and then report computational findings.

3.4.1 Stochastic Multidimensional Knapsack Problems Test Sets

General knapsack constrained stochastic programs have received attention in the literature. Knapsack constraints appear in many applications of SIP such as investment planning ([31] and [54]), transportation, scheduling, selling of assets and investment selection ([60] and [61]) and operations strategy [34]. The stochastic multidimensional knapsack problem test instances we consider were first reported in

a dissertation in [15]. This class of SIP can be formulated as follows:

$$\begin{aligned} &\text{Min} \sum_{i=1}^{n_1} c_i^\top x_i + \mathcal{Q}_E(x) \\ &\text{s.t.} \ \sum_{i=1}^{n_1} x_i \leq b \\ &\qquad x_i \in \{0,1\}, \ \forall i = 1 \dots n_1 \end{aligned} \tag{3.11}$$

The function $\mathcal{Q}_E(x)$ is given as,

$$\mathcal{Q}_E(x) = \mathbb{E}_\omega \Psi(\omega, x), \tag{3.12}$$

In problem (3.11), x denotes the first-stage decision vector, $c \in \mathbb{R}^{n_1}$ is the first-stage cost vector, $b \in \mathbb{R}$ is the first-stage righthand side, $\Psi(\omega, x)$ is the recourse function with ω as a realization of a multivariate random variable $\tilde{\omega}$, and $\mathbb{E}_\omega$ denotes the mathematical expectation operator satisfying $\mathbb{E}_\omega[\,|\Psi(\omega, x)|\,] < \infty$. The underlying probability distribution of $\tilde{\omega}$ is discrete with a finite number of realizations (scenarios/subproblems) in set Ω and corresponding probabilities $p_\omega, \omega \in \Omega$. Thus for a given scenario $\omega \in \Omega$, the recourse function $\Psi(\omega, x)$ is given by the following second-stage binary program:

$$\begin{aligned} \Psi(\omega, x) = \text{Min} &\sum_{i=1}^{n_2} q(\omega)^{i\top} y(\omega)^i \\ \text{s.t.} \ &\sum_{i=1}^{n_2} w^{ij} y(\omega)^i \leq h(\omega)^j - \sum_{i=1}^{n_1} x_i, \ \forall j = 1 \dots m_2 \\ &y(\omega)^i \in \{0,1\}, \forall i = 1 \dots n_2. \end{aligned} \tag{3.13}$$

In formulation (3.13), $y(\omega)$ is the recourse decision vector, $q(\omega) \in \mathbb{R}^{n_2}$ is the recourse

cost vector, $w \in \mathbb{R}^{m_2 \times n_2}$ is the recourse matrix, and $h(\omega) \in \mathbb{R}^{m_2}$ is the righthand side.

This formulation has knapsack constraints in both the first- and second-stages, and each subproblem has equal probability of occurrence. Instance data were randomly generated using the uniform distribution ($\mathcal{U}$) with different parameter values. The knapsack weights were generated by sampling from $\mathcal{U}(2, 8)$. Objective function coefficients were generated as done in [105], with the first-stage costs being chosen to be much higher than second-stage costs. Objective function coefficients for first-stage variables were sampled from $\mathcal{U}(400, 650)$ while those for the second-stage were sampled from $\mathcal{U}(6, 16)$. To generate tight knapsack constraints, the righthand side value for each of the constraints was generated by finding the maximum knapsack weight (W_{max}) for the constraint and sampling from $\mathcal{U}(2 + 2W_{max}, 4W_{max})$.

We considered four test sets, each with five randomly generated instances of same size. The problem characteristics are given in Table 3.1. The columns of the table are explained as follows, 'Problem' is the instance name, 'Scens' is the number of scenarios, 'Bvars' is the number of binary variables, 'Constr' is the number of constraints, and 'Nzeros' is the number of non-zero elements for each of the problem instances. The first numeral in the problem name describes the number of first-stage variables, the second describes the number of second-stage variables, and the third describes the number of scenarios.

3.4.2 Computational Results

Due to the large number of test instances, we give summary plots of the results to avoid distraction from the discourse and put the detailed numerical results in tables in the Appendix for the interested reader. The computational results in the Appendix are in tables A.1, A.2, A.3 and A.4, with each table reporting results for one of the four test sets. For each instance, five different replications were executed, with an

hour time limit. The columns of the tables are organized as follows: 'Instance' is the instance name and the following five columns are based on the runs from ST-FD algorithm. 'LB' is the lower bound of the algorithm, and 'UB' is the upper bound of the algorithm. 'FD Cuts' is the number of Fenchel cuts, '%FD' is the percentage of time taken for generating Fenchel cuts, and '%Gap' is the gap between the LB and UB value after the stipulated runtime of one hour. The final row of each table gives the average of the columns.

	Problem	Scens	Bvars	Constr	Nzeros
Set 1	K.10.20.25	25	510	510	15,100
	K.10.20.50	50	1,010	1,010	30,100
	K.10.20.100	100	2,010	2,010	60,100
	K.10.20.150	150	3,010	3,010	90,100
	K.10.20.200	200	4,010	4,010	120,100
Set 2	K.20.30.25	25	770	510	25,200
	K.20.30.50	50	1,520	1,010	50,200
	K.20.30.100	100	3,020	2,010	100,200
	K.20.30.150	150	4,520	3,010	150,200
	K.20.30.200	200	6,020	4,010	200,200
Set 3	K.30.40.25	25	1,030	510	35,300
	K.30.40.50	50	2,030	1,010	70,300
	K.30.40.100	100	4,030	2,010	140,300
	K.30.40.150	150	6,030	3,010	210,300
	K.30.40.200	200	8,030	4,010	280,300
Set 4	K.40.50.25	25	1,290	510	45,400
	K.40.50.50	50	2,540	1,010	90,400
	K.40.50.100	100	5,040	2,010	180,400
	K.40.50.150	150	7,540	3,010	270,400
	K.40.50.200	200	10,040	4,010	360,400

Table 3.1: DEP instance characteristics

Due to the large-scale nature of the instances, an optimality cut suggested in [64] for pure binary first-stage SIP2 was generated and added to the master problem to

completely close the gap between the lower and upper bounds for ST-FD. This was also done in [76] under the D^2 algorithms. Finally, the last column is the CPLEX MIP gap after directly solving the DEP for one hour.

The results from the tables are summarized in Figure 3.2 to show the final percent gap at termination of the algorithms. The instances for each test set (five different size instances each with five replications) are numbered 1 to 25 and are plotted on the horizontal axis of each graph. As can be seen from the plots, the results clearly show that ST-FD algorithm gives the best performance compared to DEP. The results show that the ST-FD algorithm scales well with instance size. Notice that as the size of the instances increase (from 1 to 25) so does the percent gap, an indication of increasing problem difficulty. The direct solver generally performs comparatively well on the smaller size instances in each test set, implying that decomposition may not be necessary for such instances.

As can be seen in the tables in the Appendix, most of the computation time in the ST-FD algorithm is spent in generating Fenchel cuts.

3.4.3 Stochastic Server Location Problem

We also tested the ST-FD algorithm on large-scale SSLP instances introduced in [74], and further reported in [76]. These instances were previously solved using disjunctive decomposition (D2) algorithms. Similar to multidimensional knapsack instances, due to the large-scale nature of the SSLP instances, we generated and added the L2 optimality cut [64] to the master problem to close the gap between the lower and upper bounds when x stabilizes. The results show that CPLEX is unable to solve several instances to optimality within the time limit, an indication of problem difficulty. The usage of decomposition methods is therefore necessary.

The characteristics of the instances set used for our computational tests are given in Table 3.2. This set has a total of 45 instances: 9 problem sizes with 5 replications

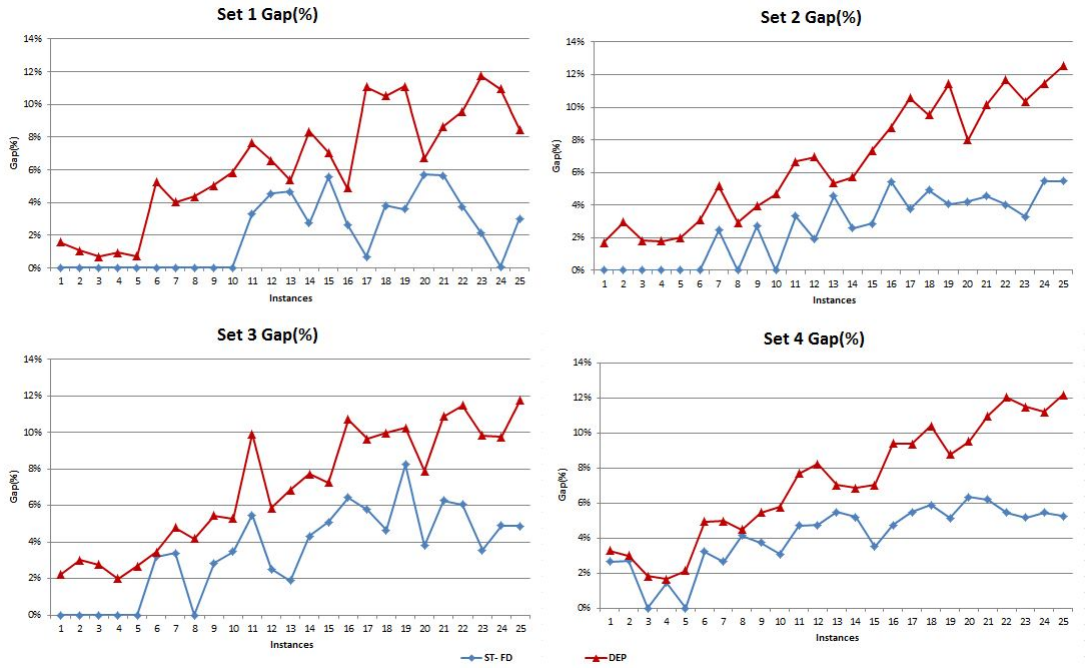

Figure 3.2: Percent gap of each test instance

each. The instances are named 'SSLPm.n.S', where m is the number of potential server locations, n is the number of potential clients, and S= $|\Omega|$ is the number of scenarios. In Table 3.2, 'Cons' is the number of constraints for the instance, 'Bins' is the number of binary variables, 'Cvars' is the number of continuous variables, 'Total Vars' is the total number of the variables, 'SS-Cons' is the number of second-stage constraints, 'SS-Bins' is the number of second-stage binary variables, and 'SS-Cvars' is the number of continuous variables in the second-stage.

3.4.4 Heuristics for Starting Solution

Larger SSLP instances needed a significant amount of time to run L-shaped algorithm, where the convergence between the lower and upper bounds was much slower. The purpose of L-shaped algorithm is to provide the ST-FD algorithm with a 'good' initial feasible solution. The poor performance of the L-shaped method

Instance	Cons	Bins	Cvars	Total Vars	SS-Cons	SS-Bins	SS-Cvars
SSLP5.25.50	1,501	6,255	250	6,505	30	130	5
SSLP5.25.100	3,001	12,505	500	13,005	30	130	5
SSLP10.50.50	3,001	25,010	500	25,510	60	510	10
SSLP10.50.100	6,001	50,010	1,000	51,010	60	510	10
SSLP10.50.500	30,001	250,010	5,000	255,010	60	510	10
SSLP10.50.1000	60,001	500,010	10,000	510,010	60	510	10
SSLP10.50.2000	120,001	1,000,010	20,000	1,020,010	60	510	10
SSLP15.45.5	301	3,390	75	3,465	60	690	15
SSLP15.45.20	1,201	13,515	300	13,815	60	690	15

Table 3.2: SSLP instance characteristics

for larger SSLP instances prompted us to devise a Starting Feasible Point (SFP) algorithm. Instead of using L-shaped algorithm in step [2] for ST-FD, we used SFP algorithm. However, with SFP the master problem of the ST-FD algorithm will not have any optimality cuts, since the L-shaped algorithm was not performed. This slows the convergence for ST-FD algorithm. To compensate for the lack of optimality cuts in the master problem, in step [2] of SFP we solved the relaxed subproblems, and provided optimality cuts to the master problem. Finally, in step [3], a lower bound to the master problem was added based on the objective function values from the relaxed subproblems. Finally, in step [4], a constraint is added to master problem (3.1) to set the lower bound for the number of non-zero binary variables based on the solution from solving DEP in step [1]. This criterion is derived based on the knowledge of the problem.

Algorithm 3 Starting Feasible Point Algorithm

[1] **Solve the instance as a DEP** with binary first-stage variables and continuous second-stage variables to get a binary first-stage solution $\hat{x}$ and relaxed subproblem solutions $\hat{y}$. The runtime is limited to 1800 seconds or a MIP-Gap of 5%.

[2] **Solve the relaxed subproblems** to get the dual solution using the first-stage solution values, and compute optimality cuts for the master problem.

[3] **Bound η in (3.1)** using the subproblem objective values.

[4] **Add a constraint** $\sum_i x_i \geq m_0$ to the master problem, where m_0 is the number of non-zero solution values from step [1].

3.4.5 Computational Results

Tables A.5 and A.6 provide the runtime characteristics using ST-FD algorithm for the SSLP instances. Table 3.3 refers the performance results using the improved ST-FD algorithm. The 'LB' and 'UB' columns refer to the lower and upper bounds obtained using SFP algorithm. The 'Itr' column provides the number of iterations performed by the ST-FD algorithm, 'ST-FD-S Gap (%)' refers the gap between the lower and upper bound after the stipulated runtime using SFP with ST-FD algorithm, 'DEP Gap (%)' is the gap for the DEP solved using CPLEX, and 'STFD Gap (%)' is the gap obtained using the standard ST-FD algorithm with the L-shaped algorithm to get the starting feasible solution. The results indicate that SFP algorithm for the starting feasible solution gives better runtime performance as the improvement on the solution gap is below 5% for most of the instances.

Instance	LB	UB	Itr	STFD-S Gap(%)	DEP Gap(%)	STFD Gap(%)
SSLP.10.50.500a	-343.04	-344.25	24	0.40	6.00	22.00
SSLP.10.50.500b	-349.37	-349.37	23	0	0.50	24.20
SSLP.10.50.500c	-335.53	-335.53	12	0	0.20	21.70
SSLP.10.50.500d	-338.54	-340.58	22	0.60	0.10	25.90
SSLP.10.50.500e	-315.33	-288.14	26	8.60	14.10	25.30
Average				1.92	4.18	23.82
SSLP.10.50.1000a	-358.87	-343.54	13	4.30	6.90	25.30
SSLP.10.50.1000b	-354.90	-331.54	12	6.60	0.50	21.40
SSLP.10.50.1000c	-336.65	-336.65	12	0	13.50	18.70
SSLP.10.50.1000d	-337.39	-340.48	12	0.90	0.60	21.90
SSLP.10.50.1000e	-325.26	-283.10	12	13.00	15.40	47.20
Average				4.96	7.38	26.90
SSLP.10.50.2000a	-360.78	-333.57	4	7.50	23.80	43.70
SSLP.10.50.2000b	-354.03	-345.90	4	2.30	24.30	44.40
SSLP.10.50.2000c	-339.61	-332.71	6	2.00	64.70	46.90
SSLP.10.50.2000d	-339.89	-343.07	4	0.90	15.90	44.80
SSLP.10.50.2000e	-329.39	-296.31	4	10.00	25.20	46.10
Average				4.54	30.78	45.18

Table 3.3: Performance results for larger SSLP instances

3.5 Conclusion

This section presented the ST-FD algorithm for two-stage SIP2s having special structure with binary variables in the second-stage. The algorithm uses Fenchel cuts generated based on the scenario subproblem under each decomposition setting to iteratively recover (at least partially) the convex hull of integer points in the neighborhood of the optimal solution. In ST-FD, the L-shaped algorithm is the method of choice for solving the LP-relaxation of SIP2 problem. Computational results on knapsack and SSLP instances show that the approach can solve large instances in reasonable amount of time in comparison to a direct solver.

4. STOCHASTIC AUTO-CARRIER LOADING PROBLEM

4.1 Introduction

Auto-carrier loading is the process of assigning vehicles to the ramps of an auto-carrier in an optimal manner. The restrictions in auto-carrier loading include government regulations on the height, weight of the axles, and overall weight of the auto-carrier. An auto-carrier contains ramps for loading the vehicles. Furthermore, the safety of the vehicles is an important factor in the loading process. With these restrictions, an auto-carrier logistic company would like to maximize their revenue by delivering the vehicles, and maintaining a higher load efficiency. In the current literature, most of the approaches focus on the integrated problem of routing and loading ([50], [55] and [45]) of auto-carriers, where the loading problem is looked upon for feasibility for a given optimal route. In our setup, we consider a cluster of loads that are to be delivered to a same destination or zip-code, hence loading becomes an important aspect than routing. Good references for algorithms for vehicle routing include [5], [102], [63] and [51]. We consider the tactical planning for the auto-carrier problem, which deals with deciding the types of resources to be used for actual delivery. Such decisions have to be performed four to five days before the actual delivery of vehicles. Based on the nature of the demand, expected revenue, operating costs of the auto-carriers, and loading challenges, the tactical plan will suggest the required resources for the operations. Using operations decisions for tactical planing has been done in the literature before. This has been done in supply chain context in [84], [47], [89] and [6].

4.2 Problem Description

We start with the details of supply chain of auto-carrier transportation, and then give the details of the loading problem.

4.2.1 Distribution Supply Chain

Auto-manufacturers get requests from the dealers for each of the vehicle types. The dealers' demands are aggregated at the auto-manufacturer location, and then vehicles are shipped to the auto-carrier locations in large quantities to take advantage of economy of scale. Auto-carriers logistics companies (ALC) receive the vehicles from the auto-manufacturers in bulk quantities, and deliver the vehicles to the dealers based on their demands. Furthermore, the ALCs have processing centers which process the vehicles based on the customization requests from the dealers. The individual customization requests do not provide an economy of scale for the auto-manufacturers. The dealerships need to carry inventory for customization requests, and this will be an operations overload for the dealers. Hence, ALCs are the preferred locations to complete the customization requests.

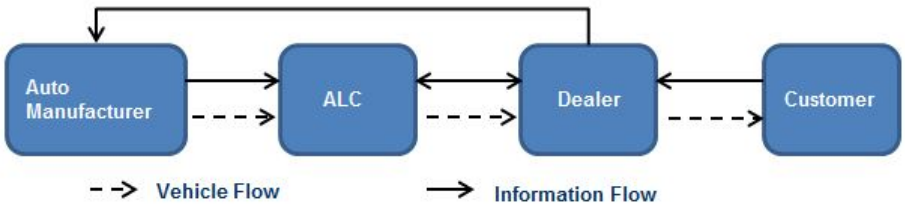

Figure 4.1: Information and vehicle flow in an auto-carrier supply chain

As depicted in Figure 4.1, the customer orders are passed from the dealers to the

auto-manufacturers. Based on the production schedule of the auto-manufacturers, the vehicles are transported (typically by rail) to an ALC in mass quantity. In the US, each vehicle is identified by a sixteen digit alpha numeric code named vehicle identification number (VIN). Each VIN is scheduled to be delivered by ALC to a specified dealer. The dealer and ALC co-ordinate for any customization needs for a particular VIN. The processing center at an ALC location processes the individual customer requests, and then the vehicles are transported to the dealers. There are very limited number of ALCs and auto-manufacturers, while the number of dealers is enormous. The core objective of ALC is to distribute the vehicles requested by the dealers at a minimized cost. The revenue for an ALC is based on the type and location of vehicle delivered.

The processes followed at an ALC are depicted in Figure 4.2. The vehicles are delivered at an ALC's receiving yard from the auto-manufacturers. The vehicles are further processed in the ALC's processing center, where additional customizations are performed based on individual customer requests. Once the processing of the vehicles is completed, the vehicles are transported by auto-carriers to the dealer locations.

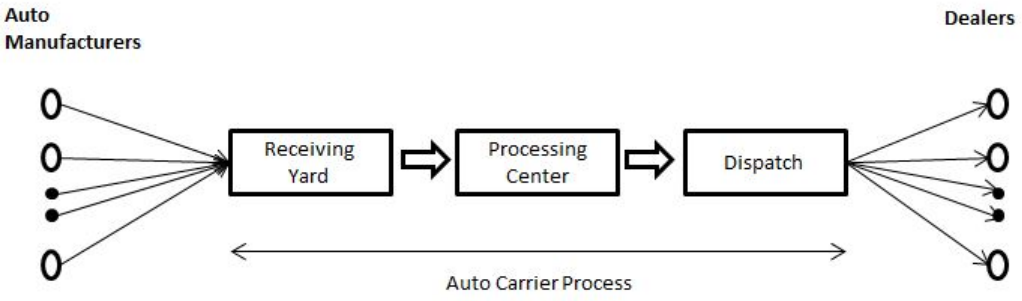

Figure 4.2: Process flow for an ALC

In the US, there are limited ALCs which are delivering vehicles to thousands of dealers. The vehicles vary in size, weight and contour making the loading problem challenging to look at mathematically.

4.2.2 Loading Challenges

ALCs own the auto-carriers used to deliver the vehicles from an ALC's distribution center to dealers. The number and capabilities of ramps depend on an auto-carrier type. The vehicle can be loaded in two different positions, namely front and back. A front position is where a vehicle faces the front side of an auto-carrier, and the back position is where a vehicle faces the rear side of an auto-carrier. All the ramps can be slid in a forward or backward position. The maximum slide for each ramp is limited, and for modeling purposes, we consider a set of discrete slide angles. The vehicle assignments to the ramps may not be one-to-one as a pair of ramps can hold a single large vehicle. We call this a 'split ramp', when more than one ramp is used for holding a vehicle. However, only a specific set of ramps in an auto-carrier can be used as split ramps. Unlike a typical cubing problem, the auto-carrier loading problem is very different with respect to loading flexibility. These flexibilities include sliding the ramps at an angle, changing the vehicle position, and split ramp. For example, vehicles can be loaded at an angle to make adjustments for total height and length of a loaded auto-carrier.

Figure 4.3 shows how the overall length of a loaded auto-carrier changes when some of the ramps are changed from the angular to the flat position. Considering L as the legal length allowed for an auto-carrier, the overall length of auto-carrier L' exceeds the legal limit when all the upper deck ramps are loaded flat. However, when the same vehicles are loaded in sliding positions, the overall length is within the legally allowed limit.

Similarly, loading vehicles on ramps at angles help meeting the legal requirements

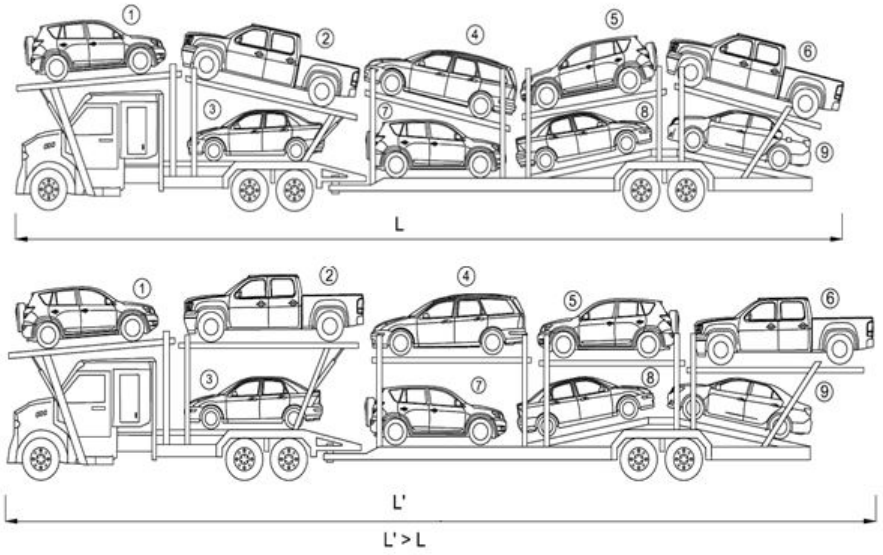

Figure 4.3: Illustration of length advantage (L′-L) due to angular ramps

for total height of an auto-carrier. Another example for benefits in height adjustment due to angular ramp adjustment can be seen in Figure 4.4. If the vehicle on ramp 2 is loaded horizontally, it would exceed the legally allowed height limit for the auto-carrier. With the help of an angular ramp, it is possible to get an advantage in height measurement, and satisfy the legal requirement for the loaded auto-carrier. Similarly, a very long vehicle can be loaded in a split ramp. Although this reduces the overall capacity (number of vehicles loaded in an auto-carrier), it helps to load larger vehicles. A sample case of a split ramp is shown in Figure 4.5.

While the above-mentioned flexibility enhances our ability to build a load with maximum number of vehicles while satisfying legal requirements for height and weight for a loaded auto-carrier, it changes the weight distribution at axles. In the Figure 4.6, let 'W1', 'W2' and 'W3' represent the steer, drive, and tandem axles of an auto-carrier respectively. The axle loads for an auto-carrier are dependent upon the position or distance of a vehicle's center of gravity from the axle of an auto-carrier

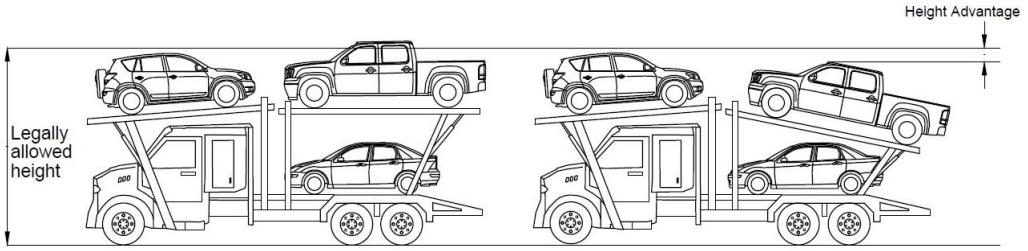

Figure 4.4: Auto-carrier showing height advantage due to backward slide on the ramps.

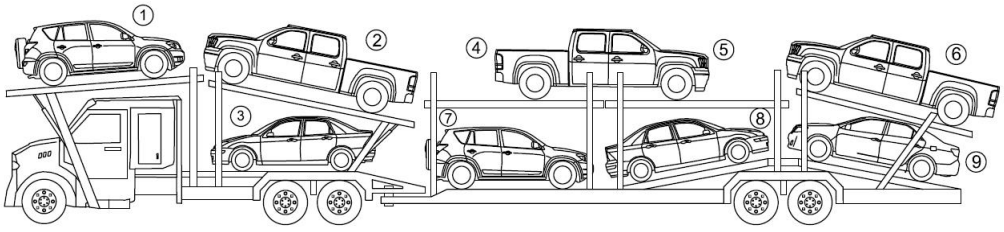

Figure 4.5: A sample case of split ramp

under consideration. Therefore, when we change the vehicle from 'backed in' position to 'driven in' (e.g., see Figure 4.6, ramp 5), it changes the amount of load it exerts on kingpin and tandem axles.

The position of vehicles in the ramps affects the load distribution on the axles. As shown in Figure 4.6, by changing the position of vehicle in ramp 5 from 'driven in' to 'backed in', the load at tandem axle increases by 310 pounds, which exceeds the allowable weight for the axle. On the other hand, the change in ramp 5 position does not make any changes to the weight in the steer axle, but reduces the drive axles load by 347 pounds. In addition to vehicle position, a change in ramp angle can also cause changes to the axle load distribution. Also, the ramp angles and vehicle positions impact the type of vehicle assigned in the adjacent ramp (both vertical

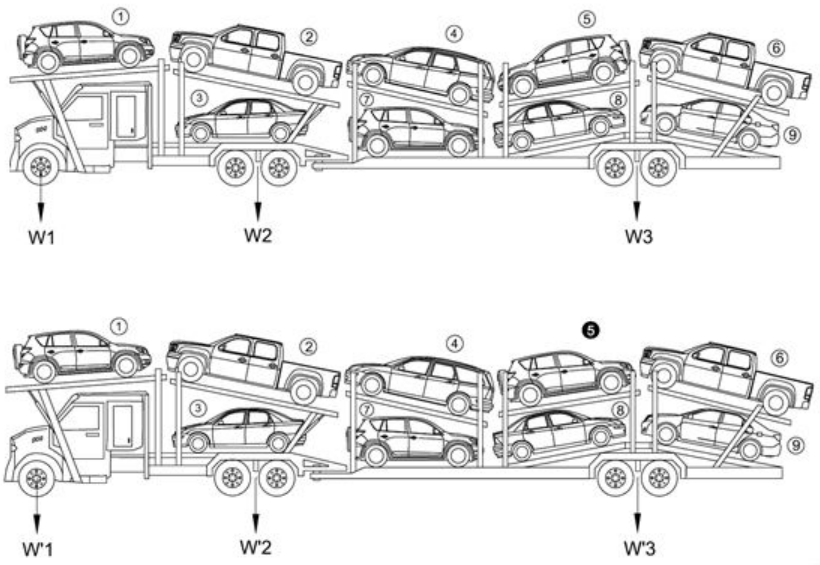

Figure 4.6: Illustration of load variations at various axles due to angular ramps and vehicle positions

and horizontal) due to physical dimensions of a vehicle such as contour, height and length.

ALCs ideally like to reduce the number of auto-carriers used for delivery thereby minimizing operating costs. The vehicles must be loaded in the auto-carriers optimally taking all the above described restrictions into account. The availability of vehicles from the processing center for loading is a random variable. Therefore, we address the tactical problem of deciding the number and types of auto-carriers required under uncertainty in the availability of the vehicles. We call this problem as stochastic auto-carrier problem. In the stochastic auto-carrier model, we choose the type and number of auto-carriers needed while minimizing the operating costs in the first-stage. In the second-stage, we maximize the total expected revenue based on the auto-carriers selected in the first-stage.

4.3 Mathematical Formulation

4.3.1 First-Stage Formulation

Notation

$R \equiv$ set of ramps, indexed by i.

$T \equiv$ set of auto-carriers, indexed by t.

$R_s(t) \equiv$ set of steering wheel ramps for auto-carrier t, $R_s(t) \subset R$.

$R_d(t) \equiv$ set of tandem wheel ramps for auto-carrier t, $R_d(t) \subset R$.

$R_u(t) \equiv$ set of ramps in the upper deck for auto-carrier t.

$R_o(t) \equiv$ set of ramps in the lower deck for auto-carrier t.

$R_i^h(t) \equiv$ set of ramps for height limitation for auto-carrier t, $R_i^h(t) \subset R$ and $\left|R_i^h(t)\right| = 2$.

$R_i^\tau(t) \equiv$ set of ramps for length limitation for auto-carrier t, $R_i^\tau(t) \subset R$.

$R_i^{sp}(t) \equiv$ set of split ramps, $R_i^{sp}(t) \subset R$ for auto-carrier t.

$J \equiv$ set of positions for a vehicle type in a ramp, indexed by j.

$K \equiv$ set of vehicle types, indexed by k.

$L \equiv$ set of allowable slides, indexed by ℓ.

$M \equiv$ set of allowable angles, indexed by m.

$l_{km} \equiv$ length of vehicle type k on a ramp inclined at angle m.

$l_{max}^t(\tau) \equiv$ maximum allowable loaded length for auto-carrier t based on the ramp set τ.

$h_{max}^t \equiv$ maximum allowable height for auto-carrier t.

$m^t \equiv$ maximum number of vehicle types allowed for auto-carrier t.

$h_k \equiv$ height of vehicle type k.

$hadv_{jk\ell m} \equiv$ height advantage for vehicle type k at position j with ramp at an angle m and slide position ℓ.

$ladv_{ijk\ell m} \equiv$ allowable height advantage in the lower deck ramp slide i for vehicle type k that can allow the upper deck vehicle type to slide (due to the contour of vehicle type at lower deck) with position j, ramp angle m and slide position ℓ.

$nsplits_i^t \equiv$ number of ramps within the split $R_i^{sp}(t)$ for auto-carrier t.

$w_k \equiv$ weight of vehicle type k.

$sw_{ijk\ell m} \equiv$ weight (at steering axle of an auto-carrier) of the vehicle type k at ramp i, at position j, ramp angle m and slide position ℓ.

$dw_{ijk\ell m} \equiv$ weight (at driving axle of an auto-carrier) of the vehicle type k at ramp i, at position j, ramp angle m and slide position ℓ.

$sw_{max} \equiv$ maximum allowable weight at the steering axle for any auto-carrier.

$dw_{max} \equiv$ maximum allowable weight at the driving axle for any auto-carrier.

$tw_{max} \equiv$ maximum allowable weight at the tandem axle for any auto-carrier.

$k^t \equiv$ kingpin weight constant of auto-carrier t.

$aw_{max} \equiv$ maximum allowable weight for any auto-carrier.

$c^t \equiv$ operating cost of auto-carrier t.

$tMax \equiv$ maximum number of auto-carriers.

$r_k \equiv$ potential revenue for the vehicle type k.

$\Omega \equiv$ set of scenarios, a scenario ω is defined for the number of available vehicle types.

$lr(i) \equiv$ ramp in the lower deck for the given ramp $i \in R$.

$a_k^\omega \equiv$ number of available vehicle type k for the scenario ω.

$p_\omega \equiv$ probability of occurrence for scenario ω.

Decision variables

In the first-stage, a decision is made whether or not a particular auto-carrier is to be used for the loading in the second-stage. We minimize the total auto-carrier operating costs in the first-stage based on the expected revenue from the second-stage. The total number of auto-carriers to be used is capacitated by the availability of the drivers or required loads.

$$x^t = \begin{cases} 1, & \text{if auto-carrier } t \text{ is used,} \\ 0, & \text{otherwise.} \end{cases}$$

Objective function

The objective function is given as follows:

$$\text{Min} \sum_{t \in T} c^t x^t + \mathbb{E}_\omega \Phi(x, \omega) \tag{4.1}$$

We minimize the total operating costs of auto-carriers. A scenario ω is a realization of random variable $\tilde{\omega}$. Assuming $\tilde{\omega}$ is a discrete random variable with finite realizations, $\mathbb{E}_\omega \Phi(x, \omega)$ can be represented as $\mathbb{E}_\omega \Phi(x, \omega) = \sum_{\omega \in \Omega} p_\omega \Phi(x, \omega)$. A scenario ω describes a realization for the availability of vehicle types based on the random variable $\tilde{\omega}$.

Constraints

The objective function is optimized over the set of feasible solutions described by the following constraints.

Capacity constraints

$$\sum_{t \in T} x^t \leq tMax \tag{4.2}$$

Constraint (4.2) enforces the capacity limitation for the different auto-carriers to be used for loading.

4.3.2 Second-Stage Formulation

Decision variables

In the second-stage, for each scenario ω, a decision is made on how to load the vehicle types on ramps considering the possible positions, slides and angles.

$$y_{ijk\ell m}^{t\omega} = \begin{cases} 1, & \text{if vehicle type } k \text{ is assigned to ramp } i \text{ with position } j, \\ & \text{ramp angle } m, \text{ slide position } \ell \text{ for auto-carrier } t \text{ for} \\ & \text{the scenario } \omega, \\ 0, & \text{otherwise.} \end{cases}$$

$$y^{t\omega}_{sp_i jk\ell m} = \begin{cases} 1, & \text{if vehicle type } k \text{ is assigned to split ramp } sp_i \\ & \text{at position } j, \text{ ramp angle } m, \text{ slide position } \ell \text{ for auto-carrier} \\ & t \text{ for the scenario } \omega, \\ 0, & \text{otherwise.} \end{cases}$$

$$sr^{t\omega}_{sp_i jk\ell m} = \begin{cases} 1, & \text{if split ramp } sp_i \text{ is used for vehicle type } k \text{ at position } j, \\ & \text{ramp angle } m, \text{ slide position } \ell \text{ for auto-carrier } t \\ & \text{for the scenario } \omega, \\ 0, & \text{otherwise.} \end{cases}$$

$ahr^{t\omega}_{i\ell}$ adjusted height of the ramp i at slide position ℓ for auto-carrier t for the scenario ω.

$lr^{t\omega}_{i\ell}$ allowance in the lower ramp i at slide position ℓ for auto-carrier t for the scenario ω.

$kw^{t\omega}$ kingpin weight of the loaded auto-carrier t for the scenario ω.

$tdw^{t\omega}$ auto-carrier drive load weight of the loaded auto-carrier t for the scenario ω.

$tlw^{t\omega}$ auto-carrier trailer load weight of the loaded auto-carrier t for the scenario ω.

$nlw^{t\omega}$ auto-carrier net trailer load weight of the loaded auto-carrier t for the scenario ω.

Objective function

The objective function can now be formulated as follows:

$$\Phi(x,\omega) = \text{Max} \sum_{t\in T}(\sum_{k\in K} r_k(\sum_{ij\ell m} y^{t\omega}_{ij\ell m} + \sum_{sp_i jk\ell m} y^{t\omega}_{sp_i jk\ell m})) \tag{4.3}$$

Constraints

The objective function is optimized over the set of feasible solutions described by the following sets of constraints.

Height constraints

$$\sum_{i\in R^h_i, jk\ell m} h_k y^{t\omega}_{ijk\ell m} + \sum_{sp_i\in R^h_i, jk\ell m} h_k y^{t\omega}_{sp_i jk\ell m} - \sum_{i\in R^h_i,\ell} ahr^{t\omega}_{i\ell} \leq h^t_{max} \qquad \forall\, R^h_i(t),\, t\in T \tag{4.4}$$

$$ahr^{t\omega}_{i\ell} \leq \sum_{jkm} hadv_{jk\ell m} y^{t\omega}_{ijk\ell m} \qquad \forall\, i\in R,\, \ell\in L,\, t\in T \tag{4.5}$$

$$ahr^{t\omega}_{i\ell} \leq lr^{t\omega}_{lr(i)\ell} \qquad \forall\, i\in R_u(t),\, \ell\in L,\, t\in T \tag{4.6}$$

$$lr^{t\omega}_{i\ell} \leq \sum_{jkm} ladv_{jk\ell m} y^{t\omega}_{ijk\ell m} \qquad \forall\, i\in R_o(t),\, \ell\in L,\, t\in T \tag{4.7}$$

Constraints (4.4) enforce maximum height limitation for a pair of vehicle types in the lower and upper deck for each of the auto-carriers. Each element of the set $R^h_i(t)$ represents a ramp in a lower and upper deck which are vertically aligned for an auto-carrier t, and constitutes for maximum height limitation. The variable $ahr^{t\omega}_{i\ell}$

represents maximum allowable slide in vertical position of the vehicle type k assigned at ramp $i \in R_i^h(t)$ or split ramp $sp_i \in R_i^h(t)$. Constraints (4.5) limit the variables $ahr_{i\ell}^{t\omega}$ based on contour of vehicle type at the ramp $i \in R$. Constraints (4.6) limit variables $ahr_{i\ell}^{t\omega}$ based on contour of vehicle type at the ramp in lower deck. The maximum slide for a vehicle type in upper deck on top of the lower deck vehicle type is limited by $lr_{i\ell}^{t\omega}$. Constraints (4.7) limit the variables $lr_{i\ell}^{t\omega}$ based on contour of vehicle type k in the lower deck.

Weight constraints

$$\sum_{i \in R_s(t), jk\ell m} sw_{ijk\ell m} y_{ijk\ell m}^{t\omega} + \sum_{sp_i \in R_s(t), jk\ell m} sw_{sp_i, jk\ell m} y_{sp_i jk\ell m}^{t\omega} + kw^{t\omega} - tdw^{t\omega} \leq sw_{max} \qquad \forall\, t \in T \tag{4.8}$$

$$\sum_{i \in R_s(t), jk\ell m} dw_{ijk\ell m} y_{ijk\ell m}^{t\omega} + \sum_{sp_i \in R_s(t), jk\ell m} dw_{sp_i jk\ell m} y_{sp_i jk\ell m}^{t\omega} + tdw^{t\omega} \leq dw_{max} \qquad \forall\, t \in T \tag{4.9}$$

$$\sum_{i \in R_d(t), jk\ell m} dw_{ijk\ell m} y_{ijk\ell m}^{t\omega} + \sum_{sp_i \in R_d(t), jk\ell m} dw_{sp_i jk\ell m} y_{sp_i jk\ell m}^{t\omega} \leq tw_{max} \qquad \forall\, t \in T \tag{4.10}$$

$$\sum_{ijk\ell m} w_k y_{ijk\ell m}^{t\omega} + \sum_{sp_i jk\ell m} w_k y_{sp_i jk\ell m}^{t\omega} \leq aw_{max} \qquad \forall\ t \in T \tag{4.11}$$

$$tdw^{t\omega} = k^t k w^{t\omega} \qquad \forall\ t \in T \tag{4.12}$$

$$\sum_{i \in R_d(t), jk\ell m} dw_{ijk\ell m} y^{t\omega}_{ijk\ell m} + \sum_{sp_i \in R_d(t), jk\ell m} dw_{sp_i jk\ell m} y^{t\omega}_{sp_i jk\ell m} = tlw^{t\omega} \qquad \forall\, t \in T \tag{4.13}$$

$$\sum_{i \in R_d(t), jk\ell m} w_k y^{t\omega}_{ijk\ell m} + \sum_{sp_i \in R_d(t), jk\ell m} w_k y^{t\omega}_{sp_i jk\ell m} = nlw^{t\omega} \qquad \forall\, t \in T \tag{4.14}$$

$$kw^{t\omega} = nlw^{t\omega} - tdw^{t\omega} \qquad \forall\ t \in T \tag{4.15}$$

The total weight for each axle of auto-carrier t is restricted in the weight constraints. The three axle weights, namely steer axle weight, drive axle weight, and tandem axle weight are restricted in the constraints (4.8), (4.9), and (4.10), respectively. The set of ramps for steer and drive axle weights are represented in the set $R_s(t)$ for an auto-carrier t. Similarly, the set of ramps for tandem axle weight are represented in the set $R_d(t)$ for auto-carrier t. Constraints (4.11) restrict the total weight for an auto-carrier t. Constraints (4.12) provide the relationship between trailer drive load and kingpin weight for an auto-carrier t. Constraints (4.14) and (4.15) determine the value for trailer load and net load for the trailer ramps. Finally, constraints (4.16) give the relationship between kingpin, net trailer load, and trailer load.

Length constraints

$$\sum_{km} l_{km} \cdot \Big(\sum_{i \in R^\tau_i(t), j\ell} y^{t\omega}_{ijk\ell m} + \sum_{sp_i \in R^\tau_i(t), j\ell} y^{t\omega}_{sp_i jk\ell m} \Big) \leq l^t_{max}(\tau) \qquad \forall\, R^\tau_i(t),\, t \in T \tag{4.16}$$

Constraints (4.17) restrict the total length for each of the ramp sets in $R^\tau_i(t)$ for auto-carrier t.

Split Ramp constraints

$$\sum_{jk\ell m} y^{t\omega}_{sp_i jk\ell m} \leq sr^{t\omega}_{sp_i} \qquad \forall\, sp_i \in R^{sp}_i(t),\, t \in T \tag{4.17}$$

$$\sum_{ijk\ell m} y^{t\omega}_{ijk\ell m} \leq nsplits^t_i \cdot (1 - sr^{t\omega}_{sp_i}) \qquad \forall\, sp_i \in R^{sp}_i(t),\, t \in T \tag{4.18}$$

Constraints (4.18) ensure whether a ramp is used as a split ramp or not. The variable $sr^{t\omega}_{sp_i}$ is used as an indicator variable for split ramps usage. Constraints (4.19) ensure that assignment of vehicle types to the individual ramps is based on whether the ramps are used as split ramps or not.

Assignment constraints

$$\sum_{jk\ell m} y^{t\omega}_{ijk\ell m} + \sum_{i \subset sp_i \in R^{sp}_i(t), jk\ell m} y^{t\omega}_{sp_i jk\ell m} \leq 1 \qquad \forall\, i \in R,\, t \in T \tag{4.19}$$

$$\sum_{t} (\sum_{ij\ell m} y^{t\omega}_{ijk\ell m} + \sum_{sp_i \in R^{sp}_i(t), jk\ell m} y^{t\omega}_{sp_i jk\ell m}) \leq a^{\omega}_k \qquad \forall\, k \in K \tag{4.20}$$

Constraints (4.20) ensure that each ramp is assigned a maximum of only one vehicle type. Constraints (4.21) restrict the availability of each vehicle type k for scenario ω.

First/second-stage linking constraints

$$\sum_{ijk\ell m} y^{t\omega}_{ijk\ell m} + \sum_{sp_i \in R^{sp}_i(t), jk\ell m} y^{t\omega}_{sp_i jk\ell m} \leq m^t x^t \qquad \forall\, t \in T \tag{4.21}$$

Constraints (4.22) ensure that availability of a particular auto-carrier t for the second-stage is based on the first-stage decision x^t.

4.4 Solution Scheme and Instance Generation

For our analysis, the vehicle types are classified into three categories: Truck, Sedan, and Hatchback. For each of these categories, the processing times are assumed

from uniform distribution $\mathcal{U}(2, 10)$. In the future, for the generation of scenarios, a distribution fitting needs to be done using real data. Based on the batch of VINs to be delivered, and the processing times for each of the vehicle types, the availability for each vehicle type is determined. Based on the realization of the processing time for the vehicle types, the availability parameter a_k^ω is determined.

We constructed three types of action plans for processing the vehicle types. They are 1) Optimistic: the vehicle types are processed 20% earlier than the realized completion time, 2) Realistic : the vehicle types are processed as realized completion time, and 3) Pessimistic : the vehicle types are processed 20% later than the realized completion time. The action plans are kept at 20% apart from each other. However, the percentage should be based on the discretion of the planner in the processing center. As each vehicle type can take any of the three different action plans, there are 27 possible scenarios. The possible scenarios at an aggregated level is depicted in Figure 4.7. In Figure 4.7, Truck(O), Truck(R), and Truck(P) represent that all the vehicles of type 'Truck' are processed 'Optimistically', 'Realistically', and 'Pessimistically' respectively. Similarly, action plans for 'Sedan' and 'Hatchback' can be constructed.

In Figure 4.8, nine possible scenarios for Truck(O) are shown. Similarly, possible scenarios for Truck(R) and Truck(P) sets can be constructed. For an illustration, in Figure 4.8, the first scenario (Truck(O), Sedan(O), and Hatchback(O)) represent optimistic completion time for Truck, Sedan, and Hatchback. Similarly other eight scenarios can be constructed for vehicle type 'Truck'. Though there are twenty seven scenarios, there are two scenarios which could be unrealistic in practice. The scenario 'Truck(O)', 'Sedan(O)', and 'Hatchback(O)' may not be practical. This indicates that all the vehicles are processed 20% before the completion time which may not be possible without additional resources. Similarly, the scenario 'Truck(P)',

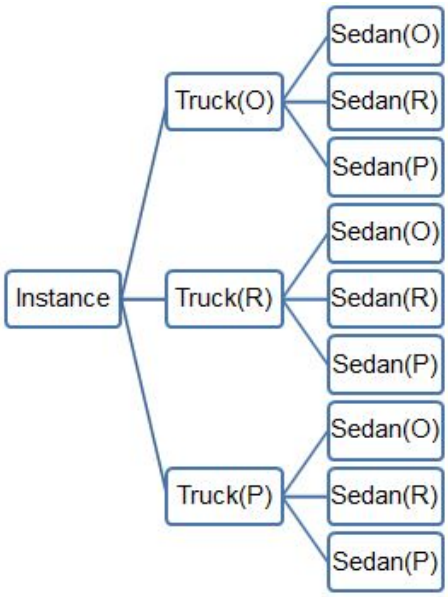

Figure 4.7: Illustration of instance generation based on action plans

'Sedan(P)', and 'Hatchback(P)' leaves too much slack for the resources. Hence, these scenarios are not included in the study. The expected profit from the load will drive the schedule of vehicles at the vehicle processing center, and the profit is a function of both the vehicle type and destination. The planner prefers to get an indication of expected profit from different possible action plans for the vehicle processing center. For example, if the planner likes to provide preference to 'Trucks' for loading, then the planner gets an indication of expected revenue from an instance with the scenarios with 'Truck(O)' getting a preference over other scenarios. Similarly, other instances can be constructed based on planner's preference.

Let F be the set of preferences of the planner. Let us consider the set F with three realizations namely 'Truck(O)', 'Sedan(O)', and 'Hatchback(O)', where 'Truck(O)' represents planner's choice to estimate the expected revenue with an optimistic action plan for the trucks at the vehicle processing center, and similarly for 'Sedan(O)' and 'Hatchback(O)'. The scenarios with 'T(O)' as the only optimistic setting are provided

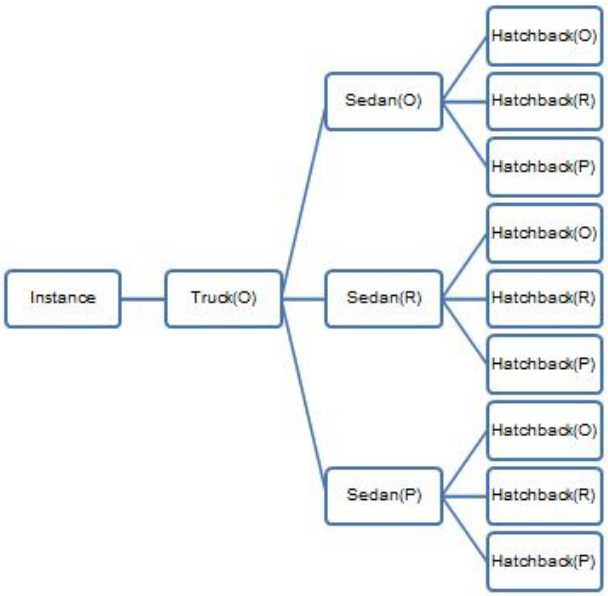

Figure 4.8: Scenario generation

with a higher probability (60%), followed by scenarios with two optimistic setting as 5%, which are rare to occur, and the rest of the probability are split among other scenarios. The probability of occurrence for each scenario is represented as,

$$p_\omega(f) = \begin{cases} 0.15, & \text{if } |f| = 1 \text{ and } |O| = 1; \\ 0.0083, & \text{if } |O| = 2; \\ 0.023, & \text{otherwise} \end{cases}$$

where f is an element from the set F. For illustration, when f is 'Truck(O)', then the scenarios with 'Truck(O)', and only one 'optimistic' setting will get probability of occurrence as 0.15. The set $|f| = 1$ represents that the condition is true, and $|O|$ denotes the number of optimistic setting in the scenario. Denoting 'T' for 'Truck', 'S' for 'Sedan', and 'H' for 'Hatchback', the scenarios with probability of occurrence as 0.15 (4 scenarios splitting 60% equally), i.e, f is 'Truck(O)' and $|O| = 1$ are

(T(O),S(R),H(R)), (T(O),S(R),H(P)), (T(O),S(P),H(R)), and (T(O),S(P),H(P)). These scenarios are more realistic as the planner needs to handle optimistic setting only for one vehicle type considering all the vehicle types have equal amount of load. Similarly, for the scenarios with two optimistic setting will get 0.0083 (6 scenarios splitting 5% equally) as probability of occurrence, and there will be six such scenarios. All the remaining scenarios, 15 of them will get 0.023 as probability of occurrence.

4.5 Computational Study

The ST-FD algorithm was used for computational study. The algorithm was implemented in C++ using CPLEX 12.1 Callable Library [40] in Microsoft Visual Studio 2010. Computations were performed on an ACPI x64 computer with an Intel®Xeon®Processor E5620 (2.4GHz) and 12GB RAM. CPLEX MIP and LP solvers were used to optimize the master problem and subproblems. The instances were run to optimality or stopped when a CPU time limit of 10,800 seconds (3 hours) was reached. As a benchmark, DEP for each test instance was created and solved using CPLEX MIP solver. We also analyzed the value of stochastic solution (VSS), which represents the advantage of using stochastic information for decision making. Without considering stochastic information, the planner could have used the expected value of the stochastic data by substituting $\tilde{\omega}$ with $E[\tilde{\omega}]$ in the recourse function. This problem is called expected value (EV) problem. The first-stage decision made using EV problem can then be evaluated in a two-stage setting. The objective is the expected value of EV problem, denoted by EEV. The difference between the EEV and the objective from the stochastic program (4.1) - (4.3) is VSS. The larger the VSS, the higher the benefit from using a stochastic model. Table 4.1 depicts the characteristics of the instance used for computational study. The instance has 20 variables in the master problem, and 1,24,440 variables in the subproblem, and the number of non-zero elements for 25 scenarios is more than 41 million.

Instance	Scenarios	MP-Vars	SP-Vars	DEP - NZs
20-	25	20	1,24,440	41,510,020

Table 4.1: Instance characteristics

The large number of non-zero elements in DEP indicate the necessity for decomposition algorithm. The runtime results are presented in Table 4.2. The columns are as follows, 'Instances' is the type of the instance, 'Auto-Carriers' is the number of auto-carriers available for loading, 'Loads' is the number of loads required, 'ST-FD gap (%)' is the percentage gap at the end of stipulated time by using ST-FD algorithm, 'DEP (%)' is the MIP gap for solving DEP using CPLEX, and 'VSS(%)' is the percentage of value of stochastic solution for the instance.

Instance	Auto-Carriers	Loads	ST-FD Gap(%)	DEP(%)	VSS(%)
Truck(O)	20	2	3.02	2.86	2.81
Sedan(O)	20	2	3.87	1.81	3.66
Hatchback(O)	20	2	4.18	1.68	4.12
Truck(O)	20	4	5.11	7.93	5.62
Sedan(O)	20	4	4.85	6.56	4.95
Hatchback(O)	20	4	4.93	4.22	4.92
Truck(O)	20	6	6.85	36.68	5.51
Sedan(O)	20	6	0.41	52.11	4.21
Hatchback(O)	20	6	2.08	49.71	3.83
Truck(O)	20	8	1.71	34.57	8.66
Sedan(O)	20	8	0.88	25.37	8.13
Hatchback(O)	20	8	1.57	44.05	6.38

Table 4.2: Runtime characteristics (2 hours)

The results clearly indicate the advantage of using ST-FD algorithm over DEP for SACP instances. The VSS % increases as the number of requested loads increases which indicates the importance of stochastic information when demand and supply

gets tighter. For e.g., the objective value for the instance Sedan(O) is around $115,000, and hence, a 8.13% VSS can be valued at $9,350. Each instance constructed for an action plan will provide an expected profit for the planner based on the uncertainty of vehicles available for loading. The planner is expected to run few action plans to decide the further course of action for the vehicle processing center.

4.6 Conclusion

In this section, we introduced stochastic auto-carrier loading problem. Then we provided two-stage SIP formulation for tactical planning on the number and types of auto-carriers needed for loading based on the uncertainty in the availability of vehicle types. Instances were generated based on the preferences of the planner's action plan for the vehicle processing center. We used ST-FD algorithm presented in the section 3 to solve the instances. Expected value problems were solved, and the results justify the use of stochastic information for decision making.

5. FENCHEL DECOMPOSITION FOR MIXED INTEGER PROGRAMS WITH SPECIAL STRUCTURE

5.1 Introduction

In the literature, Fenchel cuts have been derived for the IPs with binary variables. In this section, we derive Fenchel cuts for IPs with general integer variables and special structure. We develop theory to characterize the properties of the convex hull of integer points needed for the generation of a Fenchel cut. Then we derive an algorithm for obtaining a reduced set of integer points needed for generating a Fenchel cut based on the relaxed solution of the IP. In general, it is a challenging task to get the smallest set of integer points required for the generation of a Fenchel cut. We then extend the methodology for SIP2 with general integer variables in the second stage. We perform computational experiments with randomly generated and MIPLIB instances to evaluate the new methodology. The results from our preliminary computational experiments show that FCG procedure is able to generate additional Fenchel cuts by using the reduced set of integer points.

5.2 Fenchel Cut Generation Procedure for General Integer Programs

Let us start with the definitions needed for the derivation of Fenchel cuts for IPs with general integer variables. Based on the definition of the set F^{IP} in (3.5), let $C(F^{IP})$ denote the convex hull of feasible integer points for the set F^{IP}. Let F_v^{IP} be the subset of integer points within F^{IP}, i.e, $F_v^{IP} \subseteq F^{IP}$ and let $C(F_v^{IP})$ denote the convex hull of feasible integer points in F_v^{IP}. In this section, we consider that the set Y defined in (1.3) imposes integer restrictions on all or some of its components.

Based on the definition for F^{LP} in (3.4), let $\hat{y} \in F^{LP}$ be given, such that $\hat{y} \notin C(F^{IP})$. Then the separation problem (SP) is to find a valid inequality $\pi^\top y \leq \pi_0$ for

problem (1.3) such that $\pi^\top \hat{y} > \pi_0$, where π and π_0 are the vectors with appropriate dimensions.

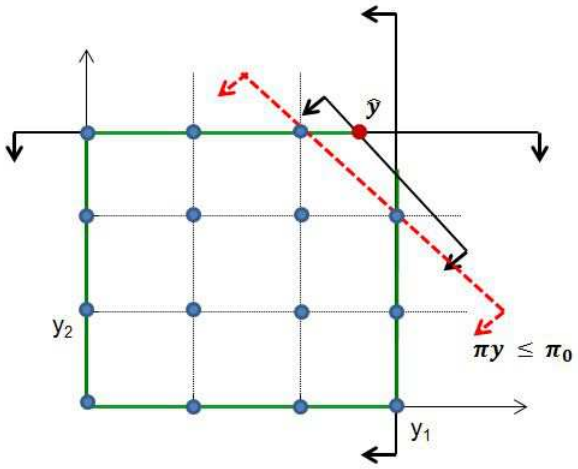

Figure 5.1: Separation problem

For the problem depicted in Figure 5.1, let $\hat{y} \in F^{LP}$ be the solution to LP-relaxation $\Phi^{LP}(\rho, \tau)$ as defined in (3.3). By solving a separation problem, we can generate a Fenchel cut $\pi^\top y \leq \pi_0$, such that $\pi^\top \hat{y} > \pi_0$.

We would like to restrict our derivation of Fenchel cuts to the subset of integer points F_v^{IP}, so that a generated cut valid for $C(F_v^{IP})$ is also valid for $C(F^{IP})$. The Fenchel cut generated using a subset of integer points is depicted in Figure 5.2.

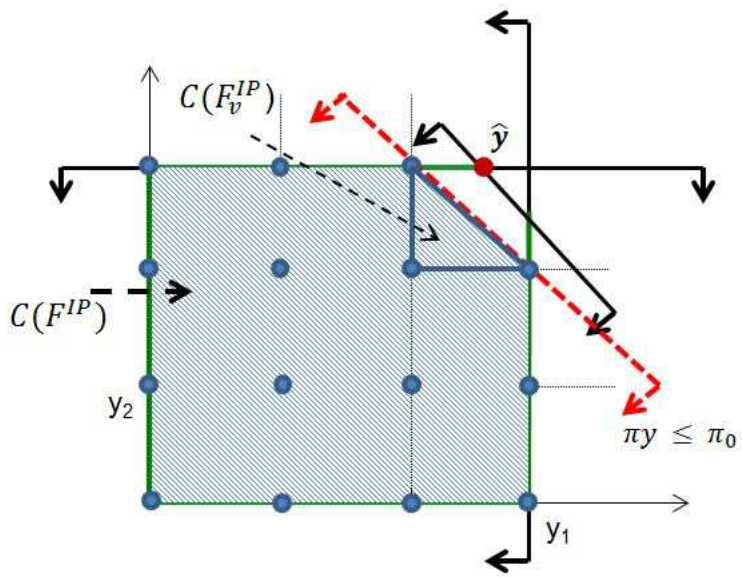

Figure 5.2: Separation problem with reduced integer feasible set

5.2.1 Integer Set Generation for Fenchel Cut Generation

We devise a methodology to obtain the subset F_v^{IP}, and subsequently use it to generate Fenchel cuts. Also, since $F_v^{IP} \subseteq F^{IP}$, we need to form the set F_v^{IP} such that the generated Fenchel cut does not cut off any integer point in the set F^{IP}.

As stated in the section 3, we need to solve problem (3.7) for generating a Fenchel cut. However, this is not a trivial task due to integrality restrictions on F^{IP}. To generate a Fenchel cut based on the LP-relaxation solution, the FCG procedure generally needs a subset of integer points of the LP-relaxation feasible set. In this section, we derive an algorithm to obtain such a set $F_v^{IP} \subseteq F^{IP}$ for generating a Fenchel cut. The smallest possible subset F_v^{IP} is desirable but we need to make sure that $C(F_v^{IP})$ does not cut off any integer solution. In this section, since our focus is to generate a subset F_v^{IP} for solving (3.7), we restate the subproblem (1.3) can be

restated as follows:

$$\begin{aligned} \text{IP}(\rho,\tau)\text{: Min } & \rho^\top y \\ \text{s.t. } & Wy \leq \tau \\ & 0 \leq y \leq u \\ & y \in \mathbb{Z}^{n_2} \end{aligned} \tag{5.1}$$

The dimensions of the parameters are as stated in section 1. The general integer variables y are bounded by the vector u. Let I be the set of variable indices, and K be the set of constraint indices for (5.1). Also, let the elements of the matrix W be denoted by w_{kt}, where $k \in K$ is the constraint index, and $t \in I$ is the variable index. We consider (5.1) under the following assumptions:

(A1) The polyhedron $\mathcal{P}$ defined by (5.1) is full dimensional with dimension n_2.

(A2) The dimension of $C(F^{IP})$ is n_2.

(A3) F^{IP} contains the origin.

(A4) All the elements of W and τ are non-negative. The formulation (5.1) has only knapsack constraints with positive co-efficients and righthand side.

Let p be an index for an integer point y^p such that $y^p \in F^{IP}$, and P is the set of indices for the integer points in $C(F^{IP})$. Let y_i^p be the i^{th} component of y. Referring to Figure 5.2, considering the integer point $y^p = (3, 2)$ then $y_1^p = 3$, and $y_2^p = 2$. Let y^{LP} be the solution to the LP-relaxation to (5.1), y^{IP} be the optimal solution to (5.1), and $\bar{y}$ be the lower bound on y. Let $\bar{y}$ be initialized as $\lfloor y^{LP} \rfloor$.

Let d_{ij}^p be the projected distance in the (i,j) space. The value of d_{ij}^p is calculated as follows:

$$d_{ij}^p = \min\left\{ \left(\left[\tau_k - \sum_{t \in I: t \neq i, t \neq j} w_{kt} y_t^{LP} - w_{ki} \bar{y}_i \right] / w_{kj} \right)_{\forall k \in K} - y_j^p,\ u_j - y_j^p \right\}, \quad (5.2)$$

where τ_k is the righthand side for the constraint $k \in K$, and u_j is the upper bound for the variable $j \in I$ Also, let $f_i(y^p)$ be the shortest projected distance along the axis j for the point y^p. The value of $f_i(y^p)$ is calculated as follows:

$$f_i(y^p) = \min\left\{ d_{ij}^p \right\}_{\forall j \in I | i \neq j} \quad (5.3)$$

Let $f_i'(y^p)$ be the shortest projected distance along the axis i for the point y^p. The value of $f_i'(y^p)$ is calculated as follows:

$$f_i'(y^p) = \min\left\{ d_{ji}^p \right\}_{\forall j \in I | i \neq j} \quad (5.4)$$

Next, we state that there exists a set $F_v^{IP} \subseteq F^{IP}$, which is sufficient to generate the Fenchel cutting plane.

LEMMA 5.1. *Given F^{IP}, there exists a set $F_v^{IP} \subseteq F^{IP}$ such that F_v^{IP} is sufficient for generating a Fenchel cut.*

Proof. Given F^{IP}, then either $F_v^{IP} = F^{IP}$ or $F_v^{IP} \subset F^{IP}$. For $F_v^{IP} = F^{IP}$, it is obvious that entire set F^{IP} can be used for generating a Fenchel cut to cut off y^{LP}. Now consider the case $F_v^{IP} \subset F^{IP}$. Since the origin belongs to F^{IP}, then $|F^{IP}| \geq n_2 + 1$ as $C(F^{IP})$ is full dimensional. Then F_v^{IP} can be constructed such that there exists an integer point $y^{p'} \notin F_v^{IP}$ and $y^{p'} \in F^{IP}$, as $|F_v^{IP}| \geq n_2$. This

is due to the fact that only n_2 affinely independent integer points are needed to construct a facet for $C(F^{IP})$. Hence, $|F^{IP}| > |F_v^{IP}|$ which gives $F_v^{IP} \subset F^{IP}$. $\square$

We state the required properties for an integer point $y^p \in F_v^{IP}$ in the following corollary.

COROLLARY 5.2. *Let $y^p \in F_v^{IP}$. Then either of the following must be true:*

(i) $d_{ij}^p < 1$, $\forall i, j \in I | j \neq i$ *or*

(ii) $d_{ij}^p \geq 1$ *and any integer point* $y^{p'} \in F^{IP}$ *such that* $y_i^{p'} - y_i^p \geq 1, \forall i \in I$ *for at least one index* $i \in I$ *also belongs to* F_v^{IP}.

When $d_{ij}^p < 1$, then there does not exist an integer point $y_i^p \in F^{IP}$. Alternatively, when $d_{ij}^p \geq 1$, then there exists an integer point $y_i^{p'} \in F^{IP}$ such that $d_{ij}^p > d_{ij}^{p'}$. This means that there is an integer point $y_i^{p'}$ between y_i^p and the binding constraint. Since, $y_i^p < y_i^{p'}$, and $y_i^p, y_i^{p'} > 0 \Rightarrow d_{ij}^p - y_i^p > d_{ij}^p - y_i^{p'} \Rightarrow y_i^{p'} - y_i^p > 0$, which implies that $y_i^{p'} - y_i^p \geq 1$, since $y_i^{p'}, y_i^p \in F^{IP}$. However by *(ii)*, if $d_{ij}^p \geq 1$ and $y_i^{p'} - y_i^p \geq 1, \forall i \in I$, then the point $y^{p'} \in F_v^{IP}$.

In corollary 5.2, we define the properties for the integer points in the set F_v^{IP}. It may be desirable to get the smallest possible set F_v^{IP}. However, by construction the Fenchel cut generated based on the set F_v^{IP} should not cut off any optimal solution in the set F^{IP}. We evaluate each of the components of y and add an integer point y^p to the set F_v^{IP} if it is the closest integer point to y^{LP} for the component or if all other integer points between y^p and constraints are already in the set F_v^{IP}.

In the following lemma, we state the requirements for the minimum cardinality for the set F_v^{IP}. Ideally, we would like the set F_v^{IP} to have lesser number of integer points, as evaluating (3.7) is expensive.

LEMMA 5.3. *Given F_v^{IP}, there exists an integer point $y' \in F_v^{IP}$. Let Υ_i be a set of integer points such that*

$$\Upsilon_i = \underset{y^p}{argmin}\ \{f_i(y^p) \mid y^p \in F^{IP}, y_i^p > 0\}\ then\ y' = \underset{y^k}{argmin}\ \{f_i'(y^k) \mid y^k \in \Upsilon_i\} \quad (5.5)$$

for all $i \in I$.

Proof. When $C(F^{IP})$ is not full dimensional, then it is obvious that any point $y^p \in \{F_v^{IP} \cup F^{IP}\}$ will satisfy the lemma. However for assumption (A2), we prove this by contradiction. Let $y^{p'}$ be the origin, and also let us assume that there exists an index $i \in I$ for which $y^p \notin F_v^{IP}$ with $y_i^p > 0$ for any $p \in P$. But this would mean that $y^{p'} \in F_v^{IP}$, so a Fenchel cut could potentially pass through $y^{p'}$. This implies that $C(F^{IP})$ is not full dimensional, which is a contradiction. □

For any two points $y^{p'}, y^{p''} \in F^{IP}$, if $d_{ij}^{p'} > d_{ij}^{p''}$ for a given $j \in I | j \neq i$, then if $y^{p'} \in F_v^{IP} \Rightarrow y^{p''} \in F_v^{IP}$ by corollary 5.2. However, for a *smallest* set F_v^{IP}, $y^{p''} \in F_v^{IP}$.

PROPOSITION 5.4. *The minimum cardinality of the set F_v^{IP} is n_2.*

Proof. By lemma 5.3, there should be at least one integer point for every component $i \in I$. This implies that $|I| = n_2$. Also, since Fenchel cutting planes are facets, then we need at least n_2 affinely independent points in the set F_v^{IP} for generating the facet. □

Based on the properties stated, we would like to form a smallest set F_v^{IP} for evaluating (3.7). Based on the properties of lemma 5.3 and proposition 5.4, $\left|F_v^{IP}\right| = n_2$, assuming (5.1) is full dimensional. However, getting the smallest set is not trivial unless we have an oracle providing an ideal interior point $y^p \in F^{IP}$, on which the smallest set F_v^{IP} can be constructed. In the next section, we present an algorithm

to obtain the set F_v^{IP} using the corollary 5.2, lemma 5.3, and proposition 5.4. In the algorithm, we start with an initial point $y^p \in F_v^{IP}$, where y^p is constructed based on y^{LP}. The corollary 5.2 and lemma 5.3 are used to check whether the set of points in F_v^{IP} are enough and valid, if not then the set F_v^{IP} is expanded further by adding adequate integer points from the set F^{IP}.

5.2.2 Integer Set Generation Algorithm

In Algorithm 4, we initially use the lower bounds for the components of y^{LP} to obtain the set F_v^{IP}. In each iteration, we evaluate two components of y along i and j, and the lower bounds for the components are decreased based on the projected distance of the components from the binding constraints. Each pair of components, y_i and y_j are evaluated in two dimensional space by projecting all other variables into the two (i, j) dimensional space. The pair of components (y_i, y_j) are evaluated in the two dimensional space with the criterion that $\bar{y_i}$ and $\bar{y_j}$ provide at least one integer point for the generation of the Fenchel cut in i^{th} direction. We also make sure that $\bar{y_i}$ and $\bar{y_j}$ do not remove any integer point in $C(F^{IP})$, so that the generated Fenchel cut does not cut off any optimal solution. The algorithm for obtaining the set F_v^{IP} is stated as follows:

Algorithm 4 Integer Set Generation (ISG) Procedure

[1] **Initialize:** Let y^{LP} be the LP-relaxation solution for the program $\Phi^{LP}(\rho, \tau)$. Let $\bar{y}$ be the lower bound of the variables in (5.1), and initialized as $\bar{y}_i = \lfloor y_i^{LP} \rfloor$. Let $K' \subseteq K$ be the subset of indices for the binding constraints at current solution y^{LP}.

[2] **Compute Distance** d_{ij}**:**

for $i \in I$ **do**

  for $j \in I \setminus i$ **do**

    $d_{ij} = 0, f_k = 0$

    for $k \in K'$ **do**

      (a) *Assign RHS for constraint index k:*

      $f_k \leftarrow \tau_k$

      (b) *Projections for all other variables except indices i and j:*

      for $t \in I \setminus \{i, j\}$ **do**

        $f_k \leftarrow f_k - w_{kt} y_t^{LP}$

      end for

      (c) *Compute Distance:*

      $r_k \leftarrow f_k - w_{ki} \bar{y}_i$

      $d_{ij} \leftarrow \min \{r_k / w_{kj}, u_j\}$

    end for

  end for

end for

[3] **Evaluate of Bounds:**

for $i \in I$ **do**

  for $j \in I \setminus i$ **do**

    for $k \in K'$ **do**

      (d) *Evaluate* d_{ij}*:*

      if $(i < j)$ & $(d_{ij} < 1)$ **then**

        if $\bar{y}_i \geq 1$ **then**

          $\bar{y}_i \leftarrow \bar{y}_i - 1$; $k \leftarrow k - 1$;

        end if

      else if $d_{ij} - \bar{y}_j < 1$ **then**

        if $\bar{y}_i \geq 1$ **then**

          $\bar{y}_i \leftarrow \bar{y}_i - 1$; $k \leftarrow k - 1$;

        else if $\bar{y}_j \geq 1$ **then**

          $\bar{y}_j \leftarrow \bar{y}_j - 1$; $k \leftarrow k - 1$;

        end if

      end if

      (e) *Check for Integer Points:*

      $b \leftarrow 1$

      while $\bar{y}_i - b > 0$ **do**

        $r_k^{(1)} \leftarrow f_k - w_{ki}(\bar{y}_i - b)$; $d_{ij}^{(1)} \leftarrow r_k^{(1)}/w_{kj}$; $b \leftarrow b + 1$;

        if $(\lfloor d_{ij}^{(1)} \rfloor - \lfloor d_{ij} \rfloor \geq 1)$ & $(\lfloor d_{ij}^{(1)} \rfloor \leq u_i)$ **then**

          $\bar{y}_i \leftarrow \bar{y}_i - b$

        end if

      end while

    end for

  end for

end for

[4] **Use the computed lower bound** $\bar{y}_i$ for the variable index $i \in I$ in FCG.

In step [1], we initialize the parameters using the solution to the LP-relaxation for (5.1) represented as y_i^{LP} for $i \in I$, where I is the set of variable indices. Furthermore, $\bar{y}_i$ is the parameter of the algorithm which we intend to use as lower bound for the i^{th} component in the formulation (5.1). We would like to *maximize* the value of $\bar{y}_i$ without cutting off any integer solution for the original problem. Initially $\bar{y}_i$ is initialized to $\lfloor y_i^{LP} \rfloor$. In step [2a], the righthand side of binding constraint $k \in K'$ is assigned to parameter f_k. In step [2b], for any $i, j \in I$, such that $i \neq j$, we calculate the projected distance d_{ij} in (i, j) space as given in (5.4). We then evaluate d_{ij} in step [2c]. The distance d_{ij} is the measure of distance from the other component's axis to the binding constraint. If d_{ij} indicates the absence of an integer point along i^th axis, then both $\bar{y}_i$ and constraint index are reduced by one, so that $\bar{y}$ will be re-evaluated again for the binding constraint k using the expanded set F_υ^{IP}. Hence, we start with a smallest set of integers based on $\lfloor y_i^{LP} \rfloor$, and as the algorithm progresses, the set F_υ^{IP} is expanded by decreasing $\bar{y}$ based on corollary 5.2 and lemma 5.3.

In step [3d], we check the property *(ii)* of corollary 5.2, and if there are any integer points along the component i, then $\bar{y}_i$ is reduced further to accommodate additional integer points into the set F_υ^{IP}. A set obtained using ISG procedure is illustrated in Figure 5.3.

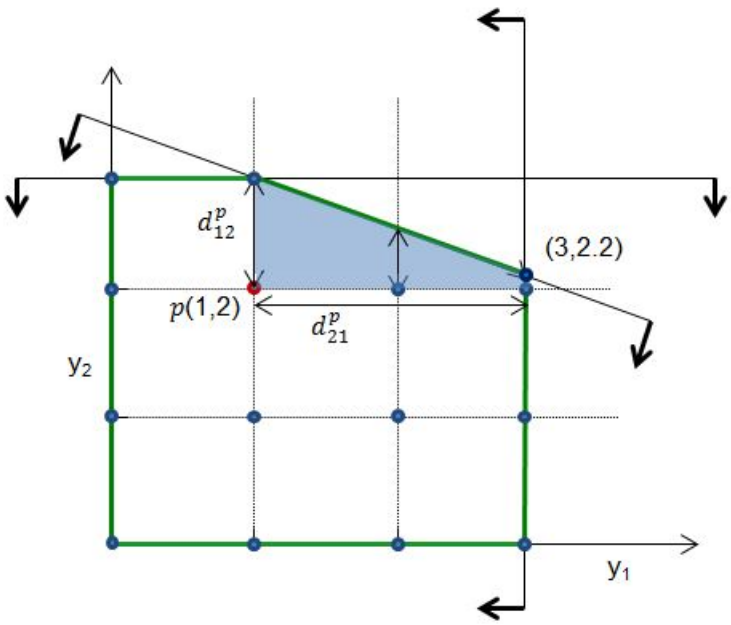

Figure 5.3: Illustration of ISG procedure

We need to make sure that the reduced set $C(F_v^{IP})$ is sufficient to get the required Fenchel cut. We evaluate only the set of binding constraints indices $K' \subseteq K$ based on y^{LP}, since we are only interested in the set F_v^{IP} based on y^{LP} for generating the Fenchel cut. In step [2] and [3], we evaluate each pair $(i, j) \in I$ for the binding set of constraints indices $K' \subseteq K$. For a given pair (i, j), we calculate the distance between the new lower bound $\bar{y}_i$ of the component and the binding constraint. If the calculated distance d_{ij} is less than one, then $\bar{y}_i$ has to be reduced further as there is no integer point available in the given axis for the generation of Fenchel cut. If $\bar{y}_i$ can not be lowered further due to the lower bound of the variable, then $\bar{y}_j$ is reduced to accommodate an integer point for $C(F_v^{IP})$. The complexity of the algorithm is $\mathrm{O}(n^3 m)$, where n is the number of variables, and m is the number of constraints. The two components projection is similar to 'Fourier-Motzkin elimination method'.

The details of Fourier-Motzkin elimination method can be found at [43], [107], and the details of its implementation in [17], [67] and [85].

Next, we describe a procedure for integer set generation (ISG) procedure within FCG for solving 5.1.

Algorithm 5 Integer Set Generation (ISG) Procedure for Integer Programs

[1] **Initialize:** set $t \leftarrow 0$.

[2] **Solve LP-Relaxation:** Solve the LP-relaxation of problem (5.1), and let y^{LP} be the relaxed solution to (5.1).

[3] **Check solution integrality:**

if $y^{LP} \in \mathbb{Z}$ **then**

Report y^{LP} as optimal.

Stop.

end if

[4] **Use ISG and FCG to generate Fenchel cuts:**

Run FCG with ISG to get $\beta^{(t)}$ and $g(\beta^{(t)})$.

Add the cut $\beta^{(t)}y \leq g(\beta^{(t)})$ to problem (5.1).

Increment the counter $t \leftarrow t + 1$.

Go to step [2].

We use Algorithm 5 for testing instances of formulation (5.1). In the algorithm, we first initialize the parameters in step [1]. In step [2], we solve the LP-relaxation of the problem , and the solution is stored as y^{LP}. We check of integrality for y^{LP} in step [3]. If y^{LP} is integral, we report y^{LP} as optimal. Otherwise, we generate a Fenchel cut using the ISG procedure, and add to the LP-relaxation of (5.1), and continue to step [2].

5.2.3 Numerical Example

Let problem (5.1) be given as follows:

$$\begin{aligned} \text{IP1: } & \text{Min} - y_1 - y_2 \\ & \text{s.t. } 0.4y_1 + y_2 \leq 3.4 \\ & \quad 0 \leq y_1, y_2 \leq 3 \\ & \quad y_1, y_2 \in \mathbb{Z}, \end{aligned} \tag{5.6}$$

The LP-relaxation to the problem (5.6) has the optimal solution (3, 2.2). Thus, a Fenchel cut has to cut off the fractional point. The steps are as follows:

1. $y^p = (\lfloor 3 \rfloor, \lfloor 2.2 \rfloor)(3, 2)$. Therefore, $y_1^p = 3$, $y_2^p = 2$, $\bar{y}_1 = 3$, $\bar{y}_2 = 2$, based on $y_1^{LP} = 3$ and $y_2^{LP} = 2.2$.

2. $i = 1, j = 2, k = 1$, $r_1 \leftarrow 3.4 - 0.4(3) = 2.2$.
   $d_{12}^p = \min\{2.2/1 - y_2^p, 3 - y_2^p\} = 0.2$.
   Since $d_{12}^p < 1 \Rightarrow \bar{y}_1 \leftarrow \bar{y}_1 - 1 = 3 - 1 = 2$, update y^p as $(2, 2)$, and re-evaluate the constraint. In the next iteration, $\bar{y}_1 = 1$, and the algorithm passes to the next component $\bar{y}_2$.
   Similarly, for $\bar{y}_2 = 2$, $r_1 \leftarrow 3.4 - 2 = 1.4$.
   $d_{21}^p = \min\{1.4/0.4 - y_1^p, 3\} = \min\{3.5 - y_1^p, 3 - y_1^p\} = 2$.
   Since $d_{21}^p \geq 1$, we do not make any changes to $\bar{y}_2$. Hence, the new lower bounds for the components are $y_1 = 1$ and $y_2 = 2$, respectively.

The value $d_{12}^p = 0.2$ represent the distance between the point $y^p(3, 2)$ and the point $y^p(3, 2.2)$ for a binding constraint in y_2 component's direction. Since there is no integer point in the direction, the algorithm iterates to reach the point $y^{p'}(2, 2)$,

where the distance $d_{12}^{p'} = \min\left\{2.6/1 - y_2^{p'}, 3 - y_2^{p'}\right\} = 0.6$. The distance 0.6 is the distance between the point $y^{p'}(2,2)$ and the binding constraint in $y_2^{p'}$ direction. Since $d_{12}^{p'} < 1$, the algorithm is continued to next iteration. The distance $d_{12}^{p''} = \min\left\{3/1 - y_2^{p''}, 3 - y_2^{p''}\right\} = 1$ is the distance between the point $y^{p''}(1,2)$ and the binding constraint at the point $y^{p''}(1,2)$.

The feasible set based on the new origin $(1,2)$ is depicted in Figure 5.4-(a). The feasible set is used for the generation of Fenchel cuts using FCG procedure, and the generated Fenchel cut is depicted in Figure 5.4-(b). The procedure to shift the origin is performed using step (d) in ISG procedure. Based on the ISG procedure, the new origin for the FCG procedure is shifted to (1, 2).

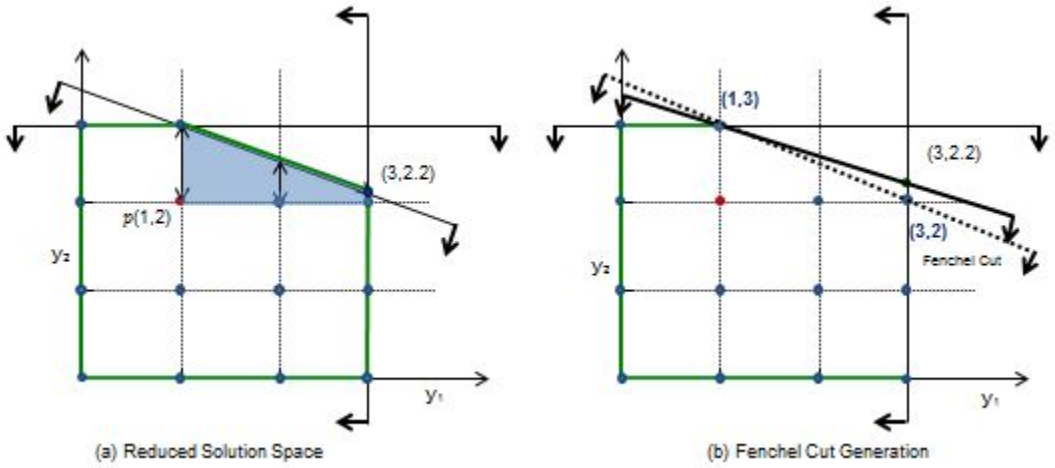

Figure 5.4: ISG procedure - step (d)

Let us consider another problem for (5.1) and depicted in Figure 5.5 as follows:

$$\begin{aligned} \text{IP1: } & \text{Min} - 1.2y_1 - 3.4y_2 \\ & \text{s.t. } 6y_1 + 5y_2 \leq 37.4 \\ & 0 \leq y_1, y_2 \leq 5 \\ & y_1, y_2 \in \mathbb{Z}, \end{aligned} \tag{5.7}$$

The LP-relaxation for the problem (5.7) gives the solution (5,1.48). Thus, the objective for FCG procedure is to cut off the relaxed solution from the solution space. Using step [d] of the ISG procedure, the new origin for FCG procedure is shifted to (4,0). However, based on $C(F_v^{IP})$, the generated Fenchel cut removes an integer point (2,5) from the convex hull of the solution space. Hence, we use step [e] to deduct any possibility of removing off integer points from the solution space based on the current reference obtained using step [d]. Thus step [e] gives the new reference (2,0) accounting for the possible integer points in $C(F^{IP})$. The reduced solution space based on the new origin (4,0) without step (e) is shown in Figure 5.5-(a). The reduced solution space based on step [e] for the generation of Fenchel cuts using the FCG procedure is shown in Figure 5.5-(b).

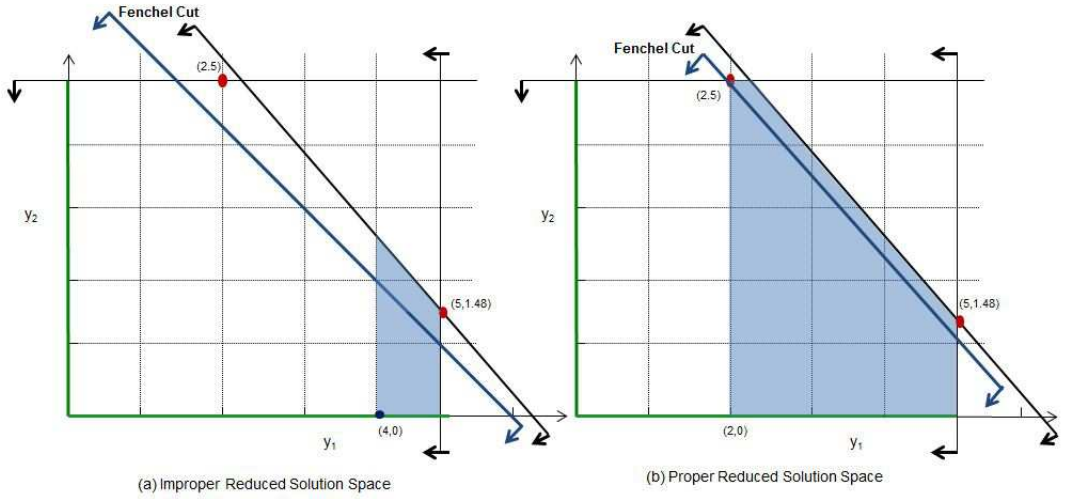

Figure 5.5: ISG procedure - step (e)

The newly computed lower bounds from ISG procedure is used for (3.10) in step [2] of FCG. The objective of ISG procedure is to obtain a smaller reduced set $F_v^{IP} \subseteq F^{IP}$ which further gives $C(F_v^{IP}) \subseteq C(F^{IP})$. A reduced set F_v^{IP} for solving (3.10) is expected to provide better runtime for the generation of Fenchel cut.

5.3 Computational Study

We performed a computational study of the ISG procedure using bounded knapsack problems. We used randomly generated instances to validate the procedure, and instances from the literature to quantify the performance. The procedure is embedded in the ST-FD algorithm, and computational experiments were performed for SIP2s with general integer variables in the second-stage.

5.3.1 Integer Programs Test Set

We performed computational study of Algorithm 5. The focus of the study for this section is to validate ISG procedure rather than measuring its performance.

The objective values for each instance are compared with the corresponding optimal value of DEP. We constructed instances for the formulation (5.1) in 12 different sizes. The sets one to six are 'small', while the sets seven to twelve are 'large'. The characteristics of the test sets are given in Table 5.1,

Instances	Instance Characteristics - Small		
	# Variables	# Constraints	Avg # NZs
Set1	4	4	17
Set2	4	4	17
Set3	6	6	37
Set4	6	6	33
Set5	8	8	58
Set6	8	8	59

Table 5.1: Instance characteristics - small

Each of the sets has 100 randomly generated instances. In the tables 5.1 and 5.3, '# Variables' is the number of variables in the model, '# Constraints' is the number of constraints in the model, and 'Avg # NZs' is the average number of non-zeros in the model.

Instances	Runtime Characteristics - Small			
	Avg # R-MIPs	Avg # R-FC-Cuts	Avg # MIPs	Avg # FC-Cuts
Set1	11.71	0.84	11.71	0.84
Set2	10.45	0.73	10.48	0.73
Set3	33.59	1.30	33.35	1.29
Set4	34.62	1.29	34.61	1.29
Set5	80.61	1.93	80.55	1.93
Set6	65.28	1.56	65.38	1.56

Table 5.2: Runtime characteristics - small (100 instances for each set)

In Table 5.2, 'Avg # MIPs' denotes the average number of MIPs solved for FCG procedure, 'Avg # FC-Cuts' denotes the average number of Fenchel cutting planes generated by FCG, 'Avg # R-MIPs' denotes the average number of MIPs solved for FCG procedure using ISG procedure, 'Avg # R-FC-Cuts' denotes the average number of Fenchel cutting planes generated by FCG using ISG procedure. The results for larger instances are given in Table 5.3 and 5.4.

Instances	Instance Characteristics - Large		
	# Variables	# Constraints	Avg # NZs
Set7	10	10	92
Set8	10	10	92
Set9	15	15	204
Set10	15	15	204
Set11	20	20	361
Set12	20	20	361

Table 5.3: Instance characteristics - large

Instances	Runtime Characteristics - Large			
	Avg # B-MIPs	Avg # B-FC-Cuts	Avg # MIPs	Avg # FC-Cuts
Set7	147.46	2.38	148.12	2.39
Set8	138.25	2.27	138.32	2.27
Set9	506.16	4.01	504.65	4.01
Set10	544.45	4.33	546.34	4.32
Set11	1523.26	7.24	1502.06	7.17
Set12	1007.36	4.54	985.18	4.46

Table 5.4: Runtime characteristics - large (100 instances for each set)

In each instance, the upper bound of the variable was set to five. Random

parameter '$rMin$' is the product of '# Variables', upper bound for the variables, and 2. Another random parameter '$rMax$' is the product of '# Variables', upper bound for the variables, and 5. Then the right hand side for each of the constraints is set as $rMin + (\mathcal{U}(0,1) * (rMax - rMin))$. Similarly, let '$vMin$' be the lower bound for the variable, and '$vMax$' be the upper bound of the variable. The co-efficient for the variable in a constraint is generated as $vMin + (\mathcal{U}(0,1) * (vMax - vMin))$. The co-efficients of the variables in the objective function are taken from uniform distribution $\mathcal{U}(0,10)$.

Tables 5.2 and 5.4 represent the average number of Fenchel cuts and MIPs solved for each of the sets. E.g., 3,335 MIPs are solved for 100 instances of Set3 without ISG to generate 129 Fenchel cuts. Similarly, 3,359 MIPs are solved for 100 instances with ISG procedure to generate 130 Fenchel cuts. Hence, 24 additional MIPs are solved for ISG for 100 randomly generated instances.

5.3.2 MIPLIB Instances

We tested the implementation of the algorithm with appropriate MIPLIB [1] instances. We chose MIPLIB instances with constraint type legend as 'IKN' (Integer knapsack), and the characteristics of the instances are given in the Table 5.5.

Instance	Rows	Cols	NZs	Int	Bin	Con	Obj
noswot	182	128	735	25	75	28	-41
lectsched-4-obj	14,163	7,901	82,428	236	7,665	-	4
lectsched-2	30,738	17,656	186,520	369	17,287	-	0
neos16	1,018	377	2,801	41	336	-	446

Table 5.5: Problem characteristics - MIPLIB

The columns of Table 5.5 are explained as follows : 'Instance' is the name of

the MIPLIB instance, 'Rows' is the number of constraints for the instance, 'Cols' is the number of variables for the instance, 'NZs' is the number of non-zero elements for the instance, 'Int' is the number of general integer variables, 'Bin' is the number of binary variables, 'Con' is the number of continuous variables, and 'Obj' is the objective value for the instance.

Tables 5.6 and 5.7 depict the runtime characteristics for MIPLIB instances with and without using ISG procedure, respectively. The columns are explained as follows : 'Instance' is the name of the instance, '# F Cuts' is the number of Fenchel cuts generated, 'LB' is the lower bound of the algorithm, 'UB' is the upper bound of the algorithm, 'MIPs Solved' is the number of MIPs solved for generating the Fenchel cuts for the instances, 'Time(secs)' is the time taken for each of the instances, and 'MIPs/F Cut' is the average number of MIPs solved for generating a Fenchel cut.

Instance	# F Cuts	LB	UB	MIPs Solved	Time(secs)	MIPs/F Cut
noswot	130	-41	-41	5,609	3,626	43.1
neos16	1	425	446	36	10,800	36.0

Table 5.6: Runtime characteristics - FCG-ISG

A 3-hour time limit is used, and there is only one instance that could reach optimality within three hours using ISG procedure. Two out of four instances are able to generate Fenchel cuts thus producing lower and upper bounds for the algorithm using ISG procedure. As stated earlier, the ISG procedure has a complexity of $O(n^3m)$. As shown in Table 5.5 for runtime characteristics for the instances, ISG procedure is able to scale only for two instances with lesser number of 'Rows' and 'Cols'. However, for the *scalable* instances, FCG procedure with ISG uses fewer number of MIPs compared to FCG procedure without ISG as noted in the column

'MIPs/F Cut' in Tables 5.6 and 5.7. This indicates that ISG has let FCG procedure to evaluate lesser number of integer points for the function (3.10) compared to FCG procedure without using it.

Instance	# F Cuts	LB	UB	MIPs Solved	Time(secs)	MIPs/F Cut
noswot	160	-41	-41	23,179	3,634	144.9
neos16	1	425	446	38	10,800	38.0
lectsched-2	3	0	0	74	10,600	24.7
lectsched-4-obj	4	0	4	434	10,800	108.5

Table 5.7: Runtime characteristics - FCG

FCG procedure without ISG is able to obtain lower and upper bounds as it only needs to evaluate (3.10) rather than obtaining the reduced integer set for the instances.

5.4 Extension to Stochastic Integer Programs

We embedded ISG procedure with ST-FD algorithm, and computational experiments are performed based on randomly generated instances for two-stage SIP2s with general integer variables in the second-stage.

5.4.1 Multidimensional Knapsack Instances for SIP

General knapsack constrained stochastic programs have received attention in the literature. This class of SIP can be formulated as follows:

$$\begin{aligned} &\text{Min} \sum_{i=1}^{n_1} c_i^\top x_i + \mathcal{Q}_E(x) \\ &\text{s.t.} \sum_{i=1}^{n_1} x_i \leq b \\ &\quad x_i \in \{0,1\},\ \forall i = 1 \dots n_1 \end{aligned} \tag{5.8}$$

In the above formulation, $\mathcal{Q}_E(x)$ denotes the expected second-stage cost based on the first-stage decision x, and we assume randomness in the objective costs, technology matrix and righthand side in the second-stage. The function $\mathcal{Q}_E(x)$ is given as,

$$\mathcal{Q}_E(x) = \mathbb{E}_\omega \Phi(q(\omega), h(\omega), T(\omega)x), \tag{5.9}$$

In problem (5.8), x denotes the first-stage decision vector, $c \in \Re^{n_1}$ is the first-stage cost vector, $b \in \Re$ is the first-stage righthand side, $\mathcal{Q}_E(x)$ is the expected recourse function with $\bar{\omega}$ being a multivariate random variable, and $\mathcal{Q}_E(.)$ denotes the mathematical expectation operator satisfying $\mathbb{E}_\omega \left[\mid \Phi(q(\omega), h(\omega), T(\omega)x) \mid \right] < \infty$. The underlying probability distribution of $\bar{\omega}$ is discrete with a finite number of realizations (scenarios/subproblems) in set Ω, and corresponding probabilities $p_\omega, \omega \in \Omega$. Thus for a given scenario $\omega \in \Omega$, the recourse function $\Phi(q(\omega), h(\omega), T(\omega)x)$ is given by the following second-stage IP:

$$\begin{aligned} \Phi(q(\omega), h(\omega), T(\omega)x) = \text{Min} & \sum_{i=1}^{n_2} q(\omega)^{i\top} y(\omega)^i \\ \text{s.t.} & \sum_{i=1}^{n_2} w^{ij} y(\omega)^i \leq \sum_{i=1}^{n_1} m \cdot t^i x_i, \ \forall j = 1 \ldots m_2 \\ & \sum_{i=1}^{n_2} v^i y(\omega)^i \leq h(\omega)^j, \ \forall j = 1 \ldots m_3 \\ & 0 \leq y(\omega)^i \leq u^i, y(\omega)^i \in \mathbb{Z}^+, \forall i = 1 \ldots n_2. \end{aligned} \tag{5.10}$$

In formulation (5.10), $y(\omega)$ is the recourse decision vector, $q(\omega) \in \Re^{n_2}$ is the recourse cost vector, $w \in \Re^{m_2 \times n_2}$ is the fixed recourse parameter, and $t \in \Re^{n_1}, v \in \Re^{n_2}$ are parameter vectors taking values 0 or 1, m is a constant, and $h(\omega) \in \Re^{m_3}$ is the

righthand side. The decision vector $y(\omega)$ is bounded by vector u. Finally, $\mathbb{Z}^+$ is the set of nonnegative integers. In a supply chain context, the first-stage decision vector x can be a decision on selection of facilities or mode of transport or resources. For a realization ω, the recourse decision vector $y(\omega)$ in the second-stage could be an amount of products produced or transported based on strategic decision from the first-stage solution x. Additionally, we have included knapsack type constraints for the second-stage decision vectors. The problem setup is similar to a stochastic lot-sizing problem ([21], [52], [100] and [101]).

The formulation in (5.8)-(5.9) has knapsack constraints in both the first- and second-stages, and each subproblem has equal probability of occurrence. Instance data were randomly generated using uniform distribution ($\mathcal{U}$) with different parameter values. The knapsack weights were generated by sampling from $\mathcal{U}(2, 8)$. Objective function coefficients were generated with the first-stage costs being chosen to be much higher than second-stage costs. The coefficients for the decision vector in the objective function for the first-stage were sampled from $\mathcal{U}(0, 400)$ and for the second-stage were sampled from $\mathcal{U}(10, 20)$. To generate tighter knapsack constraints, the righthand side value of each constraint was generated by finding the maximum knapsack weight (W_{max}) for the constraint, and then sampling from $\mathcal{U}(2+2W_{max}, 4W_{max})$.

5.4.2 Larger Test Instances - I

We constructed five test sets, each with 15 randomly generated instances. The problem characteristics are given in Table 5.8. The columns of the table are as follows: 'Instance' is the name of the instance, 'Scens' is the number of scenarios, 'Bvars' is the number of binary variables, 'Ivars' is the number of general integer variables, 'Constr' is the number of constraints, 'Nzeros' is the number of non-zero elements, and 'Avg LP Gap (%)' is the average gap between LP-relaxation and final integer objective values for the problem sets. The first numeral in the problem name

describes the number of first-stage variables, the second describes the number of second-stage variables, and the third describes the number of scenarios. For e.g, 'KI.10.20.25' represents the test instance with 10 first-stage variables, 20 second-stage variables, and 25 scenarios. The runtime performance of ST-FD with ISG is compared with DEP. The DEPs were run using direct solver CPLEX 12.5. We used 12 randomly generated instances for Set1.10, and 15 randomly generated instances for each of the other sets.

	Instance	Scens	Bvars	Ivars	Constr	Nzeros	Avg LP Gap (%)
Set1.10	KI.10.20.25	25	10	500	510	13,780	24.40
Set2.10	KI.10.20.50	50	10	1,000	1,010	27,530	9.40
Set3.10	KI.10.20.100	100	10	2,000	2,010	55,030	12.90
Set4.10	KI.10.20.150	150	10	3,000	3,010	127,530	17.40
Set5.10	KI.10.20.200	200	10	4,000	4,010	174,030	17.00

Table 5.8: DEP instance characteristics

In the following figures and tables, let 'ST-FD' denotes the ST-FD algorithm using L^1-norm, and 'ST-FD-R' denotes the ST-FD using L^1-norm and ISG procedure. Figures 5.6, 5.7 and 5.8 depict the gap in percentage for each instance in Set1.10 and Set2.10, Set3.10 and Set4.10, and Set5.10, respectively. For Set1.10, the gaps for the instances using ST-FD algorithm for 'ST-FD and' 'ST-FD-R', and MIP gap from DEP are almost identical. However, for all other sets, runtime performance for ST-FD and ST-FD-R is better than DEP.

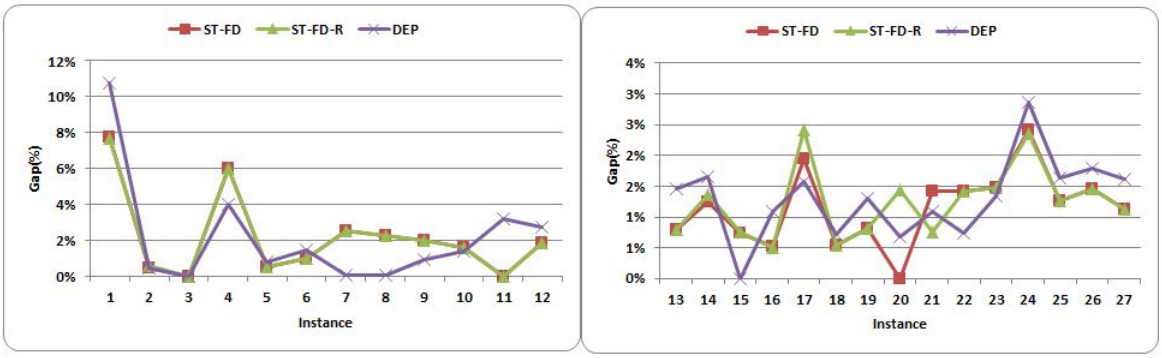

Figure 5.6: Gap percentage for Set1.10 and Set2.10

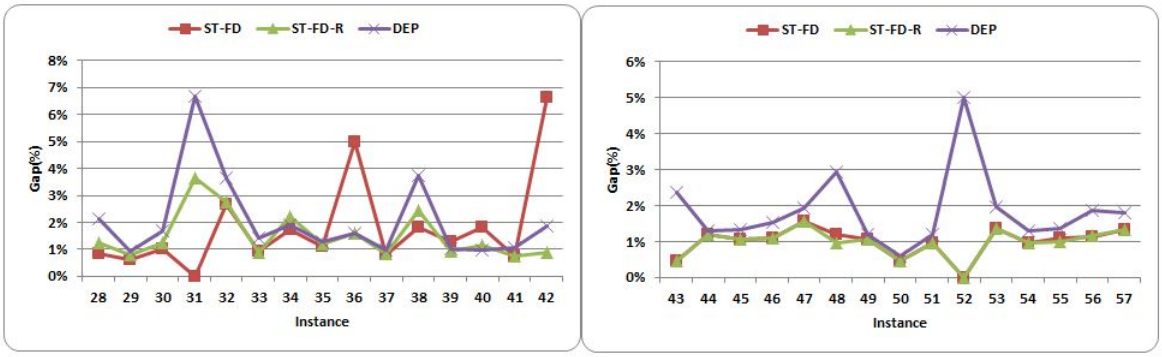

Figure 5.7: Gap percentage for Set3.10 and Set4.10

For larger test instances like Set5.10, runtime performance of ST-FD and ST-FD-R is consistently better than DEP. Also, the runtime performance of ST-FD and ST-FD-R is almost identical, and ST-FD-R performs slightly better than ST-FD for larger test instances Set4.10 and Set5.10.

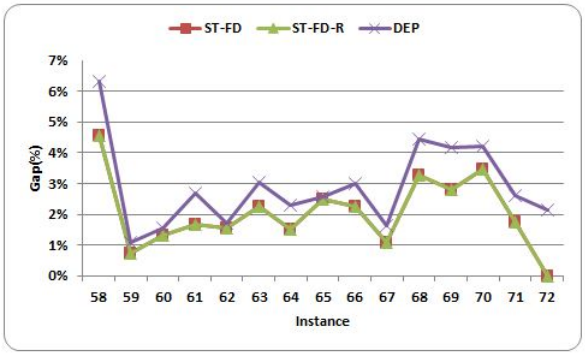

Figure 5.8: Gap percentage for test Set5.10

Figure 5.9 depicts the percentage of improvement in terms of total number of MIPs solved in ST-FD-R and ST-FD. The improvement in the percentage is given by (ST-FD-R$_{MIPs}$ - ST-FD$_{MIPs}$)/ST-FD$_{MIPs}$, where ST-FD-R$_{MIPs}$ is the total number of MIPs solved for an instance with ST-FD-R, and ST-FD$_{MIPs}$ is the total number of MIPs solved for an instance with ST-FD. The results show that ST-FD-R consistently solves a greater number of MIPs for FCG. The improvement in the percentage for the larger sets is around 3-4%.

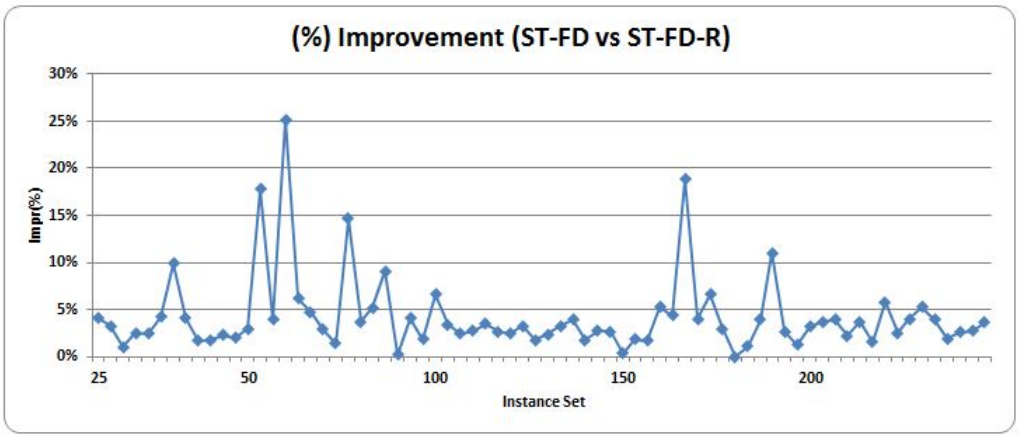

Figure 5.9: Percentage improvement for MIPs solve using ST-FD-R

Table 5.9 details the performance characteristics. The columns are explained as follows, 'Sets' is the set of instances, 'Avg Impr (%)' is the average for the improvement in the percentage of total number of MIPs solved by ST-FD-R over ST-FD, 'Avg FC Cuts (ST-FD)' is the average number of Fenchel cuts generated by FCG using ST-FD, 'Avg FC Cuts (ST-FD-R)' is the average number of Fenchel cuts generated by FCG using ST-FD-R, 'Avg Gap(%) - ST-FD' is the average gap using ST-FD, 'Avg Gap(%) ST-DF-R' is the average gap using ST-FD-R, and finally, 'Avg Gap DEP(%)' is the average MIP gap for DEP instances using direct solver CPLEX.

The results show that the average number of Fenchel cuts generated by ST-FD and ST-FD-R is almost identical, and slightly more Fenchel cuts are generated by ST-FD-R in Set3.10 and Set4.10. The improvement in the percentage for the total number of MIPs solved by ST-FD-R is around 3-4%. This indicates that ST-FD-R algorithm is able to solve more MIPs compared to ST-FD algorithm. However, both ST-FD and ST-FD-R have better gaps compared to MIP gap from DEP instances.

Sets	Avg Impr (%)	Avg FC Cuts (ST-FD)	Avg FC Cuts (ST-FD-R)	Avg Gap(%) - ST-FD	Avg Gap(%) ST-DF-R	Avg Gap DEP(%)
Set1.10	3.28	146	143	2.16	2.17	2.15
Set2.10	6.92	228	261	1.09	1.24	1.30
Set3.10	3.05	293	300	1.81	1.50	2.06
Set4.10	4.40	400	420	1.02	0.98	3.34
Set5.10	3.38	360	360	2.05	2.05	2.90

Table 5.9: Performance characteristics

5.4.3 Larger Test Instances - II

In this section, we present the computational results for KI.30.40 instances. As per the nomenclature, these instances have 30 first-stage variables and 40 second-stage variables. The DEP instance characteristics are given in table 5.10.

	Instance	Scens	Bvars	Ivars	Constr	Nzeros
Set1.30	KI.30.40.25	25	20	1,000	510	15,100
Set2.30	KI.30.40.50	50	20	2,000	1,010	30,100
Set3.30	KI.30.40.100	100	20	4,000	2,010	60,100
Set4.30	KI.30.40.150	150	20	6,000	3,010	90,100
Set5.30	KI.30.40.200	200	20	8,000	4,010	120,100

Table 5.10: DEP instance characteristics

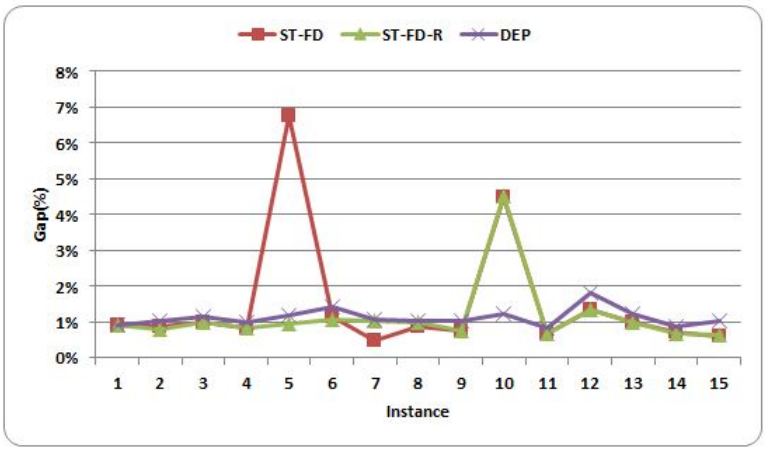

Figure 5.10: Gap percentage for Set1.30, Set2.30, and Set3.30

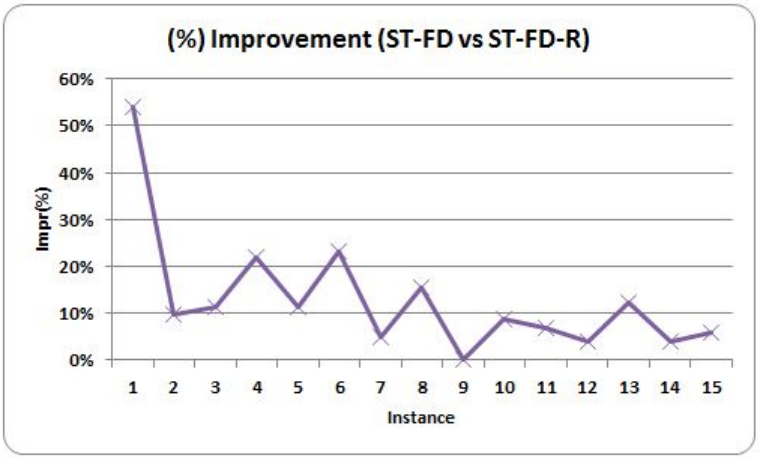

Figure 5.11: Improvement for MIPs solve using ST-FD-R (Set1.30, Set2.30, Set3.30)

Five instances are randomly generated for each set. Figure 5.10 shows the gap from ST-FD algorithm with and without using ISG procedure, and MIP gap from DEP. Set1.30 instances are numbered from one to five, followed by Set2.30 instances

from six to ten, and so on. Figure 5.11 depicts the percentage of improvement in the total number of MIPs solved by ST-FD-R and ST-FD. The results indicate that ST-FD-R is consistently able to solve more MIPs in FCG procedure. Figure 5.12 depicts the total number of Fenchel cuts generated by ST-FD and ST-FD-R. The total number of Fenchel cuts generated by ST-FD-R and ST-FD-R is almost identical for sets Set1.30, Set2.30 and Set3.30.

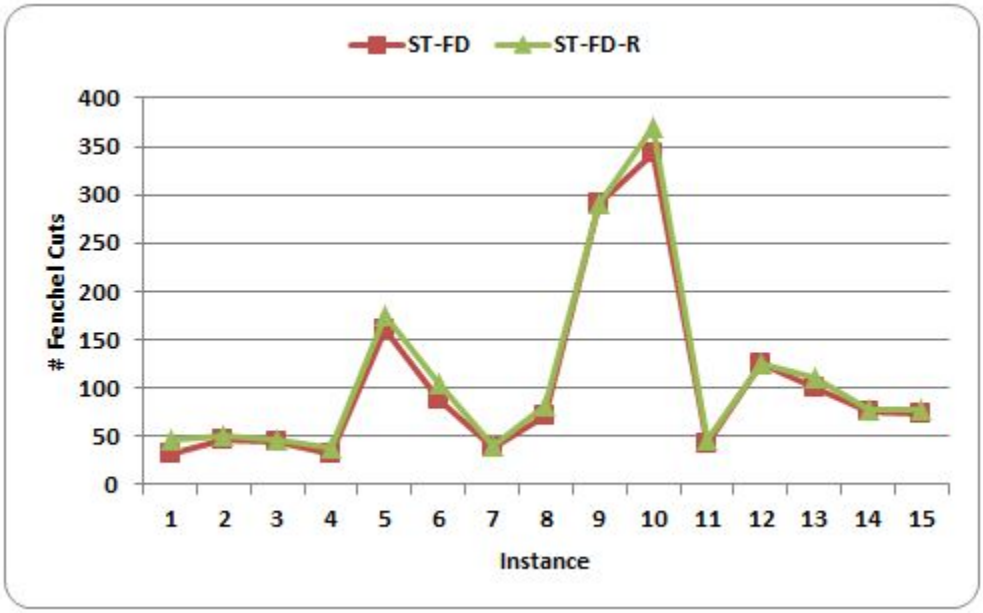

Figure 5.12: Number of Fenchel cuts for Set1.30, Set2.30, and Set3.30

5.4.4 L^1 vs L^2 Normalization

In this section, we report the computational results for larger instances Set4.30 and Set5.30 using L^1 and L^2 normalizations. We solve a LP and a quadratic program in FCG procedure with L^1 and L^2-normalizations, respectively. An L^1-normalization, $\Pi^\beta = \{\beta \in R^{n2}_+ : 0 \leq \beta \leq 1, \sum \beta \leq 1\}$ is expected to provide a better choice in

terms of solution time. However, an L^2 unit sphere,

$\Pi^{\beta} = \{\beta \in R_{+}^{n2} : 0 \leq \beta \leq 1, \sum_i \beta_i^2 \leq 1\}$ is expected to provide faster convergence in FCG procedure.

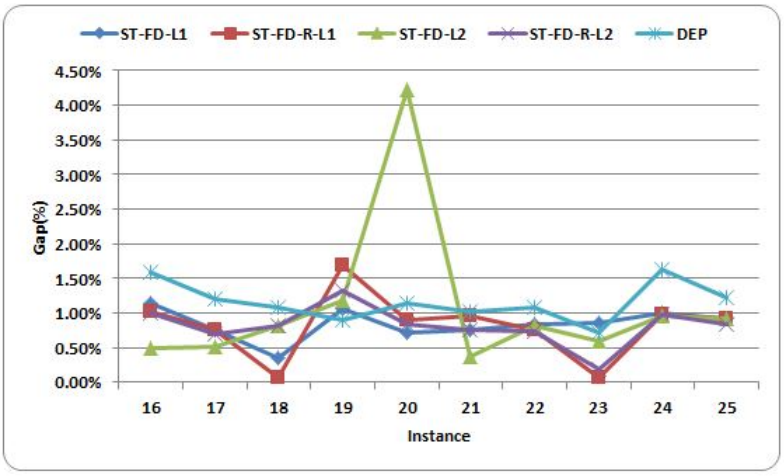

Figure 5.13: Gap percentage for Set4.30 and Set5.30

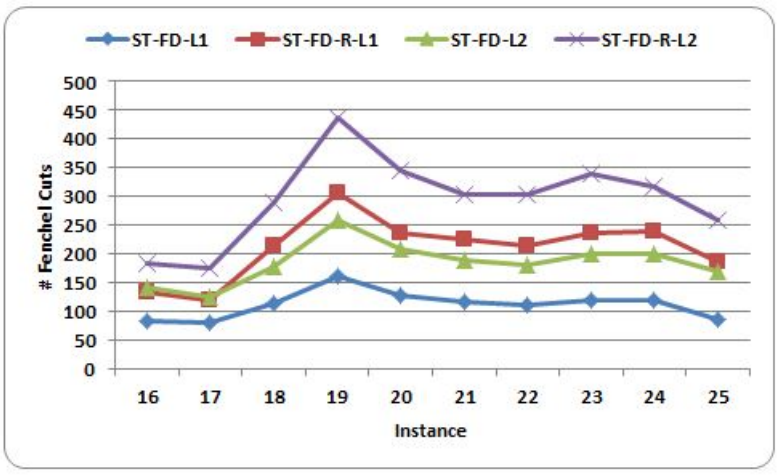

Figure 5.14: Number of Fenchel cuts for Set4.30 and Set5.30

Figure 5.13 depicts the gap for ST-FD and ST-FD-R algorithms using L^1 and L^2 normalizations. Except for few instances, both ST-FD and ST-FD-R perform better than DEP. Figure 5.14 depicts the total number of Fenchel cuts generated by ST-FD and ST-FD-R algorithms using L^1 and L^2 normalizations. ST-FD-R under L^2 normalization generates maximum number of Fenchel cuts, and ST-FD using L^1 normalization generates minimum number of Fenchel cuts. As expected, L^2 normalization provides better convergence in FCG.

Figure 5.15 depicts the ratio for the number of MIPs solved in ST-FD and ST-FD-R using L^1 and L^2 normalizations. The ST-FD algorithm with L^2 normalization solves lesser number of MIPs for the generation of Fenchel cuts, hence it is used as a benchmark. The ratio is the number of MIPs generated by each algorithm to the number of MIPs generated by ST-FD using L^2 normalization. ST-FD-R algorithm using L^1 has the highest ratio, and this indicates that more MIPs are solved by ST-FD-R during the stipulated runtime.

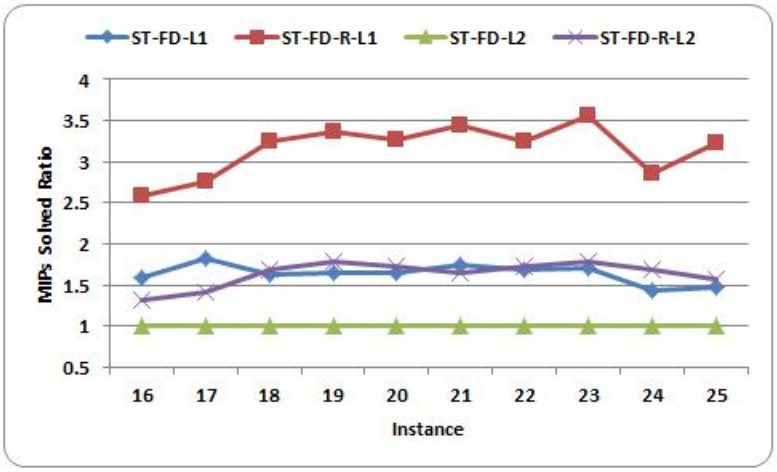

Figure 5.15: Ratio of MIPs solved for Set4.30 and Set5.30

Tables 5.11 and 5.12 represent the average performance characteristics of ST-FD, ST-FD-R and DEP using L^1 and L^2 normalizations, respectively. The columns are explained as follows, 'Avg M/F' is the average number of MIPs solved for generating a Fenchel cut using ST-FD, 'Avg M/F-R' is the average number of MIPs solved for generating a Fenchel cut using ST-FD-R, and other columns are as explained for Table 5.9.

Scenarios	Avg Impr (%)	Avg FC Cuts (ST-FD)	Avg FC Cuts (ST-FD-R)	Avg Gap(%) - ST-FD	Avg Gap(%) ST-FD-R	Avg Gap DEP(%)	Avg M/F	Avg M/F-R
Set1.30	21.64	63.00	71.40	2.07	0.90	1.06	117.68	122.57
Set2.30	10.46	166.40	177.20	1.54	1.67	1.17	93.06	94.54
Set3.30	6.53	83.60	87.40	0.87	0.87	1.17	148.69	151.53
Set4.30	83.37	113.00	202.00	0.81	0.89	1.18	134.63	140.33
Set5.30	103.27	110.20	220.20	0.88	0.74	1.13	133.29	135.46

Table 5.11: Performance characteristics - using L^1 norm

ST-FD and ST-FD-R have better average gap compared to the gap of DEP instances for most of the sets. The average number of MIPs solved for generating a Fenchel cut is slightly higher for ST-FD-R procedure compared to ST-FD. This indicates that ST-FD-R uses a greater number of iterations for convergence in FCG. However, the average gap % indicates that ST-FD-R has a better gap compared to ST-FD.

Table 5.12 indicates that the average number of MIPs solved for generating Fenchel cuts using ST-FD and ST-FD-R algorithms is almost identical. However, the results for Set4.30 and Set5.30 in Table 5.11 indicate that a Fenchel cut is generated

Scenarios	Avg Impr (%)	Avg FC Cuts (ST-FD)	Avg FC Cuts (ST-FD-R)	Avg Gap(%) - ST-FD	Avg Gap(%) ST-FD-R	Avg Gap DEP(%)	Avg M/F	Avg M/F-R
Set4.30	58.14	182.60	285.60	1.45	0.94	1.18	50.11	51.62
Set5.30	68.05	188.00	304.40	0.74	0.70	1.13	48.52	50.43

Table 5.12: Performance characteristics - using L^2 norm

with lesser number of MIPs using L^2 normalization.

5.5 Conclusion

We presented a new procedure to obtain a reduced set of integer points for FCG for generating a Fenchel cutting plane. We illustrated the validity of the procedure using IPs, and the limitations of the procedure by MIPLIB instances. Furthermore, the methodology is extended to two-stage SIP2s with general integer variables in the second-stage. The results exhibit the computational advantages by the new methodology. Even though the new methodology takes more iterations for convergence in FCG procedure, it is able to solve MIPs quickly. Finally, we compared L^1 and L^2 normalizations for FCG procedure. Despite of solving quadratic programs for L^2 normalization, it provides a faster convergence in FCG. In the future, computational instances for second-stage formulation can be generated as described in [78], [77] and [66]. This will be useful to observe the ability of ISG to reduce the gap between lower and upper bounds of ST-FD algorithm.

6. CONCLUSIONS AND FUTURE WORK

6.1 Summary

Stochastic programming is an important field within mathematical programming and deals with finding optimal solutions to mathematical programs with uncertain data. In the last decade, due to the advent of computing power, the field has been getting a lot of attention from the researchers. This dissertation develops theory and a methodology based on Fenchel cutting planes for SIPs with binary or general integer variables in the second stage. Computational experiments demonstrate the scalability of the proposed methods.

In SIP, Fenchel cutting planes are used for recovering (at least partially) the convex hull of integer points for the subproblems. Fenchel cutting planes are developed for SIP with binary variables in second-stage based on the theory developed in [73]. A computational study is performed based on randomly generated instances. The ST-FD algorithm is consistently able to outperform DEP runs. Also, the performance of ST-FD algorithm is tested using instances from the literature. SFP algorithm is introduced and used instead of L-shaped algorithm to obtain initial solution, and this new approach for ST-FD algorithm outperforms the DEP runs. This shows the importance of using a good starting point algorithm in Benders' framework.

One important application we study is the stochastic auto-carrier loading problem (SACP). This may be the first time that an SIP model has been considered for the auto-carrier loading problem. SACP involves the supply chain of vehicle delivery from the auto-manufacturers to the end customers. Our SIP model for SACP focuses on tactical planning regarding the number and type of auto-carriers required based on the random availability of vehicle types. We use the actual dimensions of vehicle

types and auto-carriers in our mathematical model. A preliminary computational study for SACP problem exhibits the scalability of ST-FD algorithm for real world problems. Similarly, higher VSS values justifies the importance of using stochastic information for real world auto-carrier loading problems.

We explore Fenchel cuts for IPs with general integer variables and develop theory and a methodology for generating a reduced set of integer points for computing a Fenchel cutting plane. The methodology is validated using computational experiments and is within the ST-FD framework to solve SIP2 with general integer variables in the second-stage. Preliminary computational experiments show that the new methodology is able to solve more MIPs during the stipulated runtime. The new methodology also generates more Fenchel cutting planes compared to ST-FD. In terms of normalizations, the L^2 norm is able to generate the most number of Fenchel cutting planes using FCG. Though quadratic programming models are to be solved for the L^2 norm, it provides better convergence as compared to the L^1 normalization.

6.2 Future Work

For two-stage SIP2 with binary variables in the second-stage with special structure, the ST-FD algorithm obtained better percentage gaps compared to the DEP. As future work, algorithm can be extended to problems with general integer variables in the first-stage. Even though the Fenchel cuts are computationally expensive to generate, ST-FD algorithm exhibited better performance for problems with special structure. Combining the idea of disjunctive cuts in disjunctive decomposition (D^2) algorithm introduced in [92] with the Fenchel cuts is a potential extension of the new method. Fenchel cuts provide deeper cuts but are computationally expensive. Hence, in the future, the D^2 algorithm can be embedded within the ST-FD algorithm to allow for generating a Fenchel cut for one scenario, and then extrapolating it to

other scenarios.

The delivery of vehicles to the dealers is a costly affair, and considering stochastic information in tactical planning leverages higher monetary benefits. Extension to SACP is to include the capacity planning of auto-carriers for auto-logistic companies (ALCs). Auto-carriers are huge investments for ALCs, and they are available in various capabilities and capacities. It is an important asset and strategic decision for ALCs to invest in auto-carriers. Based on the stochastic information on vehicle types and future demand, SIP is a suitable tool for such a decision making process. In the current SACP computational study, IPs for FCG are quickly solved, but FCG took many iterations to generate a Fenchel cut. Hence, in the future any additional effort to speed up the convergence may provide improvement in runtime performance.

In this dissertation, theory and methodology are derived for SIP2 with general integer variables in the second-stage, and then the methodology is verified by computational experiments. For larger instances, ST-FD with ISG is able to generate more Fenchel cuts compared to ST-FD without ISG procedure. There is still room to improve the theory and computational study. Another possible extension to the theory is to derive a methodology to enumerate the integer points in the convex hull of subproblem for FCG based on the reduced set generated by ISG. Finally, another potential extension is to use BAB scheme along with Fenchel cutting planes for SIP2 with general integer variables.

REFERENCES

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Miplib 2003. *Operations Research Letters*, 34(4):361–372, 2006.

[2] Gerald Y Agbegha. *An optimization approach to the auto-carrier problem*. PhD dissertation, Case Western Reserve University, Cleveland, 1992.

[3] Gerald Y Agbegha, Ronald H Ballou, and Kamlesh Mathur. Optimizing auto-carrier loading. *Transportation Science*, 32(2):174–188, 1998.

[4] Shabbir Ahmed, Mohit Tawarmalani, and Nikolaos V Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377, 2004.

[5] Manuel A Alba Martínez, Jean-Francois Cordeau, Mauro Dell'Amico, and Manuel Iori. A branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. *INFORMS Journal on Computing*, 25(1):41–55, 2013.

[6] Antonio Alonso-Ayuso, Laureano F Escudero, Araceli Garìn, Maria T Ortuño, and Gloria Pérez. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26(1):97–124, 2003.

[7] Kent Andersen, Gérard Cornuéjols, and Yanjun Li. Split closure and intersection cuts. *Mathematical Programming*, 102(3):457–493, 2005.

[8] Egon Balas. Intersection cuts - a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

[9] Egon Balas. Intersection cuts from disjunctive constraints. Technical report, DTIC Document, 1974.

[10] Egon Balas and Pierre Bonami. Generating lift-and-project cuts from the

lp simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computation*, 1(2-3):165–199, 2009.

[11] Egon Balas and Michael Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*, 94(2-3):221–245, 2003.

[12] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1-3): 295–324, 1993.

[13] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42 (9):1229–1246, 1996.

[14] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, New Jersey, 2011.

[15] Eric B Beier. *Subgradient-based Decomposition Methods for Stochastic Mixed-integer Programs with Special Structures*. PhD dissertation, Texas A&M University, College Station, 2011.

[16] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[17] Aart JC Bik and Harry AG Wijshoff. Implementation of fourier-motzkin elimination. In *Proceedings of the first annual Conference of the ASCI*, pages 377–386. Citeseer, 1994.

[18] John R Birge and Francois Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 2011.

[19] Charles E Blair and Robert G Jeroslow. The value function of an integer program. *Mathematical Programming*, 23(1):237–273, 1982.

[20] Maurizio Boccia, Antonio Sforza, Claudio Sterle, and Igor Vasilyev. A cut

and branch approach for the capacitated p-median problem based on fenchel cutting planes. *Journal of Mathematical Modelling and Algorithms*, 7(1):43–58, 2008.

[21] James H Bookbinder and Jin-Yan Tan. Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, 34(9):1096–1108, 1988.

[22] Andrew E Boyd. Generating fenchel cutting planes for knapsack polyhedra. *Siam Journal on Optimization*, 3(4):734–750, 1993.

[23] Andrew E Boyd. Solving integer programs with fenchel cutting planes and preprocessing. In *IPCO*, pages 209–220, 1993.

[24] Andrew E Boyd. Fenchel cutting planes for integer programs. *Operations Research*, 42(1):53–64, 1994.

[25] Andrew E Boyd. Solving 0/1 integer programs with enumeration cutting planes. *Annals of Operations Research*, 50(1):61–72, 1994.

[26] Andrew E Boyd. On the convergence of fenchel cutting planes in mixed-integer programming. *SIAM Journal on Optimization*, 5(2):421–435, 1995.

[27] Claus C Carøe. *Decomposition in Stochastic Integer Programming*. University of Copenhagen. Institute of Mathematical Sciences. Department of Operations Research, København, Denmark, 1998.

[28] Claus C Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1):37–45, 1999.

[29] Claus C Carøe and Jørgen Tind. A cutting-plane approach to mixed 0–1 stochastic integer programs. *European Journal of Operational Research*, 101 (2):306–316, 1997.

[30] Claus C Carøe and Jørgen Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1-3):451–

464, 1998.

[31] Robert L Carraway, Robert L Schmidt, and Lawrence R Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics (NRL)*, 40(2):161–173, 1993.

[32] Lin Chih-Hung. An exact solving approach to the auto-carrier loading problem. *Journal of Society for Transportation and Traffic Studies*, 1(1):93–107, 2013.

[33] Vasek Chvatal. *Linear Programming*. Macmillan, New York, 1983.

[34] João Claro and Jorge P deSousa. A multiobjective metaheuristic for a mean-risk static stochastic knapsack problem. *Computational Optimization and Applications*, 46(3):427–450, 2010.

[35] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Corner polyhedron and intersection cuts. *Surveys in Operations Research and Management Science*, 16(2):105–120, 2011.

[36] William Cook, Ravindran Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1-3): 155–174, 1990.

[37] Gérard Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming*, 112(1):3–44, 2008.

[38] Gérard Cornuéjols and Claude Lemaréchal. A convex-analysis perspective on disjunctive cuts. *Mathematical Programming*, 106(3):567–586, 2006.

[39] Gerard Cornuéjols and Giacomo Nannicini. Practical strategies for generating rank-1 split cuts in mixed-integer linear programming. *Mathematical Programming Computation*, 3(4):281–318, 2011.

[40] IBM ILOG CPLEX. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[41] George B Dantzig. Linear programming under uncertainty. *Management*

Science, 1(3-4):197–206, 1955.

[42] George B Dantzig. *Linear Programming and Extensions*. Princeton university press, Princeton, New Jersey, 1965.

[43] George B Dantzig and Curtis B Eaves. Fourier-motzkin elimination and its dual. *Journal of Combinatorial Theory, Series A*, 14(3):288–297, 1973.

[44] Mauro Dell'Amico, Simone Falavigna, and Manuel Iori. Optimization of a real-world auto-carrier transportation problem. *Transportation Science*, 0(0):null, 0. doi: 10.1287/trsc.2013.0492. URL `http://pubsonline.informs.org/doi/abs/10.1287/trsc.2013.0492`.

[45] Karl F Doerner, Guenther Fuellerer, Richard F Hartl, Manfred Gronalt, and Manuel Iori. Metaheuristics for the vehicle routing problem with loading constraints. *Networks*, 49(4):294–307, 2007.

[46] Marcelo LL dosSantos, Edson L daSilva, Erlon C Finardi, and Raphael EC Goncalves. Practical aspects in solving the medium-term operation planning problem of hydrothermal power systems by using the progressive hedging method. *International Journal of Electrical Power & Energy Systems*, 31(9): 546–552, 2009.

[47] Laureano F Escudero, E Galindo, G Garcıa, E Gomez, and V Sabau. Schumann, a modeling framework for supply chain management under uncertainty. *European Journal of Operational Research*, 119(1):14–34, 1999.

[48] Guenther Fuellerer, Karl F Doerner, Richard F Hartl, and Manuel Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 36(3):655–673, 2009.

[49] Dinakar Gade, Simge Kücükyavuz, and Suvrajeet Sen. Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64, 2014.

[50] Michel Gendreau, Manuel Iori, Gilbert Laporte, and Silvano Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2006.

[51] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43. Springer, New York, 2008.

[52] Yongpei Guan, Shabbir Ahmed, George L Nemhauser, and Andrew J Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105(1):55–84, 2006.

[53] Kjetil K Haugen, Arne Løkketangen, and David L Woodruff. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research*, 132(1):116–122, 2001.

[54] Mordechai I Henig. Risk criteria in a stochastic knapsack problem. *Operations Research*, 38(5):820–825, 1990.

[55] Manuel Iori and Silvano Martello. Routing problems with loading constraints. *Top*, 18(1):4–27, 2010.

[56] Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41(2):253–264, 2007.

[57] Henry J. U.S. Auto Sales Are Looking Good For December, For 2013, And For 2014. `http://www.forbes.com/sites/jimhenry/2013/12/30/u-s-auto-sales-are-looking-good-for-december-for-2013-and-for-2014/`, 2014. [Online; accessed 20-Feb-2014].

[58] Peter Kall and Stein W Wallace. *Stochastic Programming*. Hoboken, New Jersey, 1994.

[59] Willem K Klein Haneveld and Maarten H van der Vlerk. Stochastic integer

programming: General models and algorithms. *Annals of Operations Research*, 85:39–57, 1999.

[60] Anton J Kleywegt and Jason D Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[61] Anton J Kleywegt and Jason D Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[62] Nan Kong, Andrew J Schaefer, and Brady Hunsaker. Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. *Mathematical Programming*, 108(2-3):275–296, 2006.

[63] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3): 345–358, 1992.

[64] Gilbert Laporte and Francois V Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[65] Francois V Louveaux and Rüdiger Schultz. Stochastic integer programming. *Handbooks in Operations Research and Management Science*, 10:213–266, 2003.

[66] Silvano Martello and Paolo Toth. *Knapsack Problems*. Wiley, New York, 1990.

[67] Richard K Martin. *Large Scale Linear and Integer Optimization: A Unified Approach: A Unified Approach*. Springer, New York, 1999.

[68] Ernst W Mayr. Some complexity results for polynomial ideals. *Journal of Complexity*, 13(3):303–325, 1997.

[69] Brian M Miller. Auto carrier transporter loading and unloading improvement. Technical report, DTIC Document, 2003.

[70] John M Mulvey and Hercules Vladimirou. Applying the progressive hedging

algorithm to stochastic generalized networks. *Annals of Operations Research*, 31(1):399–424, 1991.

[71] George L Nemhauser and Laurence A Wolsey. *Integer and Combinatorial Optimization*, volume 18. Wiley, New York, 1988.

[72] Lewis Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research*, 58(1):229–243, 2010.

[73] Lewis Ntaimo. Fenchel decomposition for stochastic mixed-integer programming. *Journal of Global Optimization*, 55(1):141–163, 2013.

[74] Lewis Ntaimo and Suvrajeet Sen. The million-variable "march" for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400, 2005.

[75] Lewis Ntaimo and Suvrajeet Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40(3):299–319, 2008.

[76] Lewis Ntaimo and Matthew W Tanner. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *Journal of Global Optimization*, 41(3):365–384, 2008.

[77] David Pisinger. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 33(1):1–14, 1999.

[78] David Pisinger. A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, 12(1):75–82, 2000.

[79] András Prékopa. *Stochastic Programming*. Springer, New York, 1995.

[80] MT Ramos and Jesús Sáez. Solving capacitated facility location problems by fenchel cutting planes. *Journal of the Operational Research Society*, 56(3): 297–306, 2005.

[81] Ralph T Rockafellar. *Convex Analysis*, volume 28. Princeton university press, Princeton, New Jersey, 1997.

[82] Ralph T Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1): 119–147, 1991.

[83] Jesús Sáez. Solving linear programming relaxations associated with lagrangean relaxations by fenchel cutting planes. *European Journal of Operational Research*, 121(3):609–626, 2000.

[84] Tjendera Santoso, Shabbir Ahmed, Marc Goetschalckx, and Alexander Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115, 2005.

[85] Murray Schechter. Integration over a polyhedron: an application of the fourier-motzkin elimination method. *American Mathematical Monthly*, pages 246–251, 1998.

[86] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1998.

[87] Rüdiger Schultz. Continuity properties of expectation functions in stochastic integer programming. *Mathematics of Operations Research*, 18(3):578–589, 1993.

[88] Rüdiger Schultz, Leen Stougie, and Maarten H Van Der Vlerk. Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis. *Mathematical Programming*, 83(1-3):229–252, 1998.

[89] Peter Schütz, Asgeir Tomasgard, and Shabbir Ahmed. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, 199(2):409–419, 2009.

[90] Suvrajeet Sen. Algorithms for stochastic mixed-integer programming models. *Handbooks in Operations Research and Management Science*, 12:515–558, 2005.

[91] Suvrajeet Sen and Julia L Higle. An introductory tutorial on stochastic linear programming models. *Interfaces*, 29(2):33–61, 1999.

[92] Suvrajeet Sen and Julia L Higle. The c 3 theorem and a d 2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Mathematical Programming*, 104(1):1–20, 2005.

[93] Suvrajeet Sen and Hanif D Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

[94] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*, volume 9. SIAM, Philadelphia, 2009.

[95] Hanif D Sherali and Warren P Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, volume 31. Springer, New York, 1998.

[96] Hanif D Sherali and Barbara MP Fraticelli. A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1-4):319–342, 2002.

[97] Hanif D Sherali and Xiaomei Zhu. On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108(2-3):597–616, 2006.

[98] Leen Stougie, Maarten H Van Der Vlerk, et al. *Stochastic Integer Programming*. Institute of Actuarial Sciences & Econometrics, University of Amsterdam, Amsterdam, Netherlands, 1996.

[99] Roberto Tadei, Guido Perboli, and Federico D Croce. A heuristic algorithm for the auto-carrier transportation problem. *Transportation Science*, 36(1):55–62, 2002.

[100] Armagan S Tarim and Brian G Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88(1):105–119, 2004.

[101] Horst Tempelmeier. Stochastic lot sizing problems. In *Handbook of Stochastic Models and Analysis of Manufacturing System Operations*, pages 313–344. Springer, 2013.

[102] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Siam, Philadelphia, 2001.

[103] Federal Highway Administration U.S. Department of Transportation. Federal Size Regulations for Commercial Motor Vehicles. `http://ops.fhwa.dot.gov/freight/publications/size_regs_final_rpt/`, 2014. [Online; accessed 20-Feb-2014].

[104] Richard M Van Slyke and Roger Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[105] Jean-Paul Watson and David L Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370, 2011.

[106] Jean-Paul Watson, Roger JB Wets, and David L Woodruff. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554, 2010.

[107] Paul H Williams. Fourier-motzkin elimination extension to integer programming problems. *Journal of Combinatorial Theory, Series A*, 21(1):118–123,

1976.

[108] Richard D Wollmer. Two stage linear programming under uncertainty with 0–1 integer first stage variables. *Mathematical Programming*, 19(1):279–288, 1980.

[109] Laurence A Wolsey. *Integer Programming*, volume 42. Wiley, New York, 1998.

[110] Yang Yuan and Suvrajeet Sen. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, 21(3):480–487, 2009.

APPENDIX A

COMPUTATIONAL RESULTS - ST-FD

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
1	K.10.20.25a	-131.70	-131.70	350	73.93	0	1.56
2	K.10.20.25b	-126.13	-126.13	889	80.17	0	1.06
3	K.10.20.25c	-128.14	-128.14	740	75.01	0	0.70
4	K.10.20.25d	-129.82	-129.82	625	80.01	0	0.92
5	K.10.20.25e	-130.65	-130.65	650	80.75	0	0.71
	Average			651	77.97	0	0.99
6	K.10.20.50a	-88.83	-88.83	4,195	76.07	0	5.26
7	K.10.20.50b	-90.38	-90.38	5,550	66.04	0	4.03
8	K.10.20.50c	-81.75	-81.75	6,720	66.44	0	4.35
9	K.10.20.50d	-81.02	-81.02	2,944	66.49	0	5.05
10	K.10.20.50e	-87.31	-87.31	5,166	62.65	0	5.82
	Average			4,915	67.54	0	4.90
11	K.10.20.100a	-60.66	-58.71	10,469	66.34	3.32	7.64
12	K.10.20.100b	-64.77	-61.95	12,097	64.66	4.55	6.58
13	K.10.20.100c	-59.27	-56.63	12,785	58.88	4.66	5.36
14	K.10.20.100d	-57.67	-56.14	10,242	74.56	2.73	8.34
15	K.10.20.100e	-59.33	-56.21	10,635	62.55	5.55	7.05
	Average			11,246	65.40	4.16	6.99
16	K.10.20.150a	-54.94	-53.52	13,200	64.46	2.65	4.87
17	K.10.20.150b	-51.77	-51.41	11,935	61.55	0.70	11.07
18	K.10.20.150c	-55.03	-53.01	11,111	82.52	3.81	10.53
19	K.10.20.150d	-48.31	-46.63	11,264	79.07	3.60	11.09
20	K.10.20.150e	-53.17	-50.30	12,674	67.96	5.71	6.71
	Average			12,037	71.11	3.29	8.85
21	K.10.20.200a	-51.29	-48.55	13,183	65.47	5.64	8.66
22	K.10.20.200b	-50.21	-48.39	12,727	86.69	3.76	9.58
23	K.10.20.200c	-50.36	-49.31	10,600	86.90	2.14	11.74
24	K.10.20.200d	-51.21	-51.17	12,892	70.59	0.08	10.94
25	K.10.20.200e	-49.49	-48.04	13,181	85.16	3.02	8.47
	Average			12,517	78.96	2.93	9.88

Table A.1: Set 1 Computational results

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
1	K.20.30.25a	-142.13	-142.13	1,156	76.82	0	1.70
2	K.20.30.25b	-141.23	-141.23	1,150	64.85	0	2.95
3	K.20.30.25c	-138.81	-138.81	850	65.75	0	1.81
4	K.20.30.25d	-136.65	-136.65	1,775	69.90	0	1.79
5	K.20.30.25e	-141.71	-141.71	2678	65.24	0	1.99
	Average			1,522	68.51	0	2.05
6	K.20.30.50a	-92.43	-92.43	5,953	61.98	0	3.09
7	K.20.30.50b	-90.57	-88.39	7,195	68.81	2.47	5.16
8	K.20.30.50c	-89.35	-89.35	5,850	62.88	0	2.93
9	K.20.30.50d	-93.30	-90.84	7,700	82.63	2.71	3.95
10	K.20.30.50e	-90.45	-90.45	5,700	61.63	0	4.68
	Average			6,480	67.59	1.03	3.96
11	K.20.30.100a	-69.40	-67.17	8,000	80.55	3.32	6.66
12	K.20.30.100b	-59.88	-58.76	7,585	84.60	1.91	6.94
13	K.20.30.100c	-65.66	-62.81	10,734	69.14	4.54	5.34
14	K.20.30.100d	-64.34	-62.72	7,982	85.43	2.58	5.72
15	K.20.30.100e	-60.28	-58.60	7,535	82.81	2.87	7.34
	Average			8,367	80.51	3.04	6.40
16	K.20.30.150a	-59.78	-56.70	8,700	69.50	5.43	8.76
17	K.20.30.150b	-58.46	-56.34	8,231	88.67	3.76	10.57
18	K.20.30.150c	-57.58	-54.88	7,800	87.87	4.92	9.53
19	K.20.30.150d	-55.42	-53.25	8,247	78.83	4.08	11.42
20	K.20.30.150e	-58.82	-56.44	8,366	73.38	4.22	8.00
	Average			8,269	79.65	4.48	9.66
21	K.20.30.200a	-53.83	-51.48	8,570	79.11	4.56	10.18
22	K.20.30.200b	-51.70	-49.70	9,000	83.68	4.02	11.68
23	K.20.30.200c	-51.54	-49.90	9,393	82.65	3.29	10.34
24	K.20.30.200d	-55.33	-52.46	8,600	85.31	5.47	11.47
25	K.20.30.200e	-55.16	-52.29	8,365	82.87	5.49	12.53
	Average			8,786	82.72	4.57	11.24

Table A.2: Set 2 Computational results

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
1	K.30.40.25a	-142.16	-142.16	2,325	78.60	0	2.21
2	K.30.40.25b	-144.10	-144.10	3,250	60.21	0	3.02
3	K.30.40.25c	-141.16	-141.16	3,200	70.03	0	2.77
4	K.30.40.25d	-141.71	-141.71	1,400	59.18	0	1.99
5	K.30.40.25e	-146.17	-146.17	5,700	59.99	0	2.68
	Average			3,175	65.60	0	2.53
6	K.30.40.50a	-96.25	-93.25	7,650	66.15	3.22	3.45
7	K.30.40.50b	-94.07	-90.98	6,550	73.22	3.40	4.80
8	K.30.40.50c	-93.38	-93.38	5,800	60.78	0	4.18
9	K.30.40.50d	-95.30	-92.65	6,800	72.66	2.86	5.44
10	K.30.40.50e	-98.20	-94.89	6,800	57.80	3.49	5.28
	Average			6,720	66.12	2.59	4.63
11	K.30.40.100a	-68.39	-64.83	6,600	63.98	5.49	9.91
12	K.30.40.100b	-69.29	-67.59	6,600	88.68	2.51	5.87
13	K.30.40.100c	-64.26	-63.08	7,500	70.43	1.88	6.85
14	K.30.40.100d	-67.31	-64.52	6,800	86.12	4.32	7.73
15	K.30.40.100e	-67.19	-63.93	6,900	73.55	5.10	7.25
	Average			6,880	76.55	3.86	7.52
16	K.30.40.150a	-63.63	-59.78	8,530	84.77	6.44	10.73
17	K.30.40.150b	-63.01	-59.56	7,200	82.14	5.79	9.66
18	K.30.40.150c	-61.13	-58.41	7,200	89.53	4.66	9.97
19	K.30.40.150d	-62.88	-58.09	7,950	67.74	8.25	10.26
20	K.30.40.150e	-60.41	-58.20	6,750	78.86	3.80	7.90
	Average			7,526	80.61	5.79	9.70
21	K.30.40.200a	-59.19	-55.69	6,800	89.03	6.28	10.89
22	K.30.40.200b	-59.40	-56.01	7,600	72.00	6.05	11.48
23	K.30.40.200c	-53.33	-51.49	7,600	81.32	3.57	9.85
24	K.30.40.200d	-56.46	-53.82	6,991	84.42	4.91	9.76
25	K.30.40.200e	-57.13	-54.48	6,600	86.82	4.86	11.80
	Average			7,118	82.72	5.14	10.76

Table A.3: Set 3 Computational results

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
1	K.40.50.25a	-147.33	-143.50	5,175	57.68	2.67	3.29
2	K.40.50.25b	-149.68	-145.69	5,750	83.57	2.74	3.01
3	K.40.50.25c	-146.17	-146.17	5,050	55.90	0	1.85
4	K.40.50.25d	-146.70	-144.56	5,925	65.91	1.48	1.67
5	K.40.50.25e	-147.34	-147.34	4,975	58.44	0	2.16
	Average			5,375	64.30	1.38	2.40
6	K.40.50.50a	-97.30	-94.23	5,950	81.21	3.26	4.96
7	K.40.50.50b	-95.79	-93.29	5,550	84.62	2.68	4.98
8	K.40.50.50c	-99.68	-95.70	6,350	59.94	4.16	4.50
9	K.40.50.50d	-99.68	-96.07	5,900	82.22	3.76	5.49
10	K.40.50.50e	-96.50	-93.60	5,400	85.65	3.10	5.79
	Average			5,830	78.73	3.39	5.14
11	K.40.50.100a	-70.01	-66.84	6,600	75.51	4.74	7.72
12	K.40.50.100b	-69.48	-66.32	5,900	73.53	4.76	8.26
13	K.40.50.100c	-71.21	-67.50	6,100	89.03	5.50	7.05
14	K.40.50.100d	-70.39	-66.89	6,300	74.76	5.23	6.89
15	K.40.50.100e	-68.91	-66.56	6,000	91.19	3.53	7.04
	Average			6,180	80.81	4.75	7.39
16	K.40.50.150a	-60.61	-57.86	6,000	88.33	4.75	9.42
17	K.40.50.150b	-62.93	-59.65	5,850	81.39	5.50	9.41
18	K.40.50.150c	-63.87	-60.30	6,300	77.91	5.92	10.44
19	K.40.50.150d	-61.17	-58.17	5,700	82.50	5.16	8.81
20	K.40.50.150e	-65.37	-61.45	6,750	88.47	6.38	9.54
	Average			6,120	83.72	5.54	9.52
21	K.40.50.200a	-61.04	-57.46	6,800	87.20	6.23	11.01
22	K.40.50.200b	-60.11	-56.99	6,600	88.89	5.47	12.07
23	K.40.50.200c	-59.45	-56.52	6,400	88.56	5.18	11.52
24	K.40.50.200d	-59.21	-56.14	6,400	88.99	5.47	11.22
25	K.40.50.200e	-58.48	-55.55	6,200	89.43	5.27	12.19
	Average			6,480	88.62	5.53	11.60

Table A.4: Set 4 Computational results

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
1	SSLP.5.25.50a	-79.86	-79.86	5	5.47	0	0
2	SSLP.5.25.50b	-134.24	-134.24	13	8.84	0	0
3	SSLP.5.25.50c	-117.08	-117.08	0	0.41	0	0
4	SSLP.5.25.50d	-112.98	-112.98	3	0.44	0	0
5	SSLP.5.25.50e	-110.42	-110.42	10	0.11	0	0
	Average			6	3.05	0	0
6	SSLP.5.25.100a	-90.75	-90.75	65	10.67	0	0
7	SSLP.5.25.100b	-131.03	-131.03	26	12.31	0	0
8	SSLP.5.25.100c	-117.37	-117.37	1	0.03	0	0
9	SSLP.5.25.100d	-122.48	-122.48	6	5.30	0	0
10	SSLP.5.25.100e	-109.65	-109.65	10	7.30	0	0
	Average			22	7.12	0	0
11	SSLP.10.50.50a	-344.54	-344.54	247	11.52	0	0
12	SSLP.10.50.50b	-330.88	-330.88	161	7.37	0	0
13	SSLP.10.50.50c	-343.22	-343.22	300	19.15	0	0
14	SSLP.10.50.50d	-327.80	-327.80	229	15.70	0	0
15	SSLP.10.50.50e	-313.54	-313.54	264	12.03	0	0
	Average			240	13.15	0	0
16	SSLP.10.50.100a	-338.56	-338.56	465	12.20	0	0
17	SSLP.10.50.100b	-347.88	-347.88	385	9.12	0	0
18	SSLP.10.50.100c	-333.32	-333.32	492	14.34	0	0
19	SSLP.10.50.100d	-338.59	-338.59	486	15.68	0	0
20	SSLP.10.50.100e	-307.98	-307.98	523	14.05	0	0
	Average			470	13.08	0	0
21	SSLP.10.50.500a	-410.38	-322.19	4,758	39.65	27.37	6.57
22	SSLP.10.50.500b	-410.06	-318.43	4,862	34.14	28.78	0
23	SSLP.10.50.500c	-378.10	-296.15	3,814	29.65	27.67	0
24	SSLP.10.50.500d	-389.47	-288.62	3,319	24.10	34.94	0
25	SSLP.10.50.500e	-370.69	-285.63	4,134	41.80	29.78	14.02
	Average			4,177	33.87	29.71	4.12

Table A.5: Computational results SSLP instance - I

No.	Instance	**ST-FD Algorithm**					CPLEX
		LB	UB	FD Cuts	%FD	%Gap	C %Gap
26	SSLP.10.50.1000a	-447.11	-343.54	5,512	38.41	30.15	6.93
27	SSLP.10.50.1000b	-434.68	-344.30	5,852	38.06	26.25	0.54
28	SSLP.10.50.1000c	-409.12	-332.69	6,205	39.60	22.97	13.53
29	SSLP.10.50.1000d	-418.33	-328.13	5,585	39.94	27.49	0.60
30	SSLP.10.50.1000e	-392.08	-292.47	5,644	37.66	34.06	15.39
	Average			5,760	38.74	28.18	7.40
31	SSLP.10.50.2000a	-475.58	-267.68	5,236	36.65	77.67	23.83
32	SSLP.10.50.2000b	-471.24	-267.03	5,822	37.18	76.47	24.29
33	SSLP.10.50.2000c	-453.75	-251.25	7,290	41.98	80.60	64.66
34	SSLP.10.50.2000d	-460.25	-253.99	5,506	36.19	81.21	15.94
35	SSLP.10.50.2000e	-442.52	-238.37	6,999	40.99	85.64	25.18
	Average			6,171	38.60	80.32	30.78
36	SSLP.15.45.5a	-236.38	-231.20	4,060	64.02	2.24	0
37	SSLP.15.45.5b	-230.60	-230.60	3,660	62.58	0	0
38	SSLP.15.45.5c	-215.63	-208.80	3,765	68.67	3.27	0
39	SSLP.15.45.5d	-202.60	-196.40	4,014	71.78	3.16	0
40	SSLP.15.45.5e	-219.62	-212.40	4,995	78.81	3.40	0
	Average			4,099	69.17	2.41	0
41	SSLP.15.45.20a	-206.15	-206.15	1,416	26.08	0	0
42	SSLP.15.45.20b	-256.70	-256.70	190	6.73	0	0
43	SSLP.15.45.20c	-245.33	-239.80	1,156	44.32	2.31	0
44	SSLP.15.45.20d	-263.05	-263.05	147	5.62	0	0.16
45	SSLP.15.45.20e	-267.05	-262.55	354	18.67	1.71	0
	Average			653	20.28	0.80	0.03

Table A.6: Computational results SSLP instance - II

No.	Instance	ST-FD Algorithm					ST-FD-R Algorithm					CPLEX
		LB	UB	FD Cuts	%Gap	# MIPs	LB	UB	FD Cuts	%Gap	# MIPs	C %Gap
1	KI.10.20.25a	0	0	0	0	0	0	0	0	0	0	0
2	KI.10.20.25b	0	0	50	0	1,988	0	0	50	0	2,034	3.23
3	KI.10.20.25c	-286.17	-284.63	296	0.54	16,510	-286.22	-284.63	296	0.55	16,913	0.77
4	KI.10.20.25d	-234.09	-230.33	150	1.61	16,055	-234.09	-230.33	150	1.61	16,338	1.37
5	KI.10.20.25e	-305.39	-302.44	125	0.97	11,511	-305.39	-302.44	125	0.97	11,996	1.48
	Average			124	0.62	9,212			124	0.63	9,456	1.37
6	KI.10.20.50a	-415.52	-413.38	300	0.52	21,505	-415.32	-413.21	350	0.51	26,911	1.08
7	KI.10.20.50b	-486.00	-483.33	200	0.55	13,727	-486.00	-483.33	200	0.55	14,376	0.71
8	KI.10.20.50c	-254.44	-250.81	200	1.43	19,534	-488.20	-484.54	300	0.75	22,402	1.08
9	KI.10.20.50d	-392.05	-389.14	150	0.74	12,074	-392.08	-389.14	150	0.75	12,547	0
10	KI.10.20.50e	-318.00	-315.46	200	0.80	18,007	-318.01	-315.46	200	0.80	18,546	1.46
	Average			210	0.81	16,969			240	0.67	18,956	0.87
11	KI.10.20.100a	-488.20	-484.39	300	0.78	31,150	-488.20	-484.54	300	0.75	32,004	1.08
12	KI.10.20.100b	-388.95	-385.79	300	0.81	27,464	-388.95	-385.79	300	0.81	28,409	0.92
13	KI.10.20.100c	-415.31	-411.76	300	0.86	29,949	-415.31	-411.76	300	0.86	30,652	0.99
14	KI.10.20.100d	-440.42	-436.61	400	0.86	27,159	-440.42	-436.61	400	0.86	27,886	1.86
15	KI.10.20.100e	-288.39	-285.84	300	0.89	25,779	-288.39	-285.84	300	0.88	26,461	1.43
	Average			320	0.83	28,300			320	0.84	29,082	1.26
16	KI.10.20.150a	-5.56	-5.56	300	0	4,251	-5.56	-5.56	300	0	4,284	27.40
17	KI.10.20.150b	-202.44	-203.39	748	0.47	31,150	-202.44	-203.39	748	0.47	31,266	2.36
18	KI.10.20.150c	-347.46	-345.82	600	0.47	35,900	-347.46	-345.82	600	0.47	38,287	0.60
19	KI.10.20.150d	-162.02	-160.07	600	1.21	27,166	-161.62	-160.07	750	0.96	32,283	2.94
20	KI.10.20.150e	-364.47	-360.92	300	0.97	27,848	-364.47	-360.92	300	0.97	28,934	1.30
	Average			510	0.62	25,263				0.57	27,000	6.92
21	KI.10.20.200a	-354.10	-354.10	400	0	17,876	-354.10	-354.10	400	0	18,525	2.12
22	KI.10.20.200b	-380.50	-377.72	400	0.73	31,515	-380.50	-377.72	400	0.73	32,674	1.10
23	KI.10.20.200c	-451.11	-447.08	400	0.89	40,852	-451.11	-447.08	400	0.89	41,925	0.96
24	KI.10.20.200d	-247.10	-244.45	400	1.07	35,371	-247.10	-244.45	400	1.07	37,235	1.62
25	KI.10.20.200e	-405.47	-400.14	400	1.31	35,720	-405.47	-400.14	400	1.31	37,115	1.56
	Average			400	0.80	32,266				0.80	33,494	1.47

Table A.7: Set10.20 Computational results (using L^1 norm)

No.	Instance	ST-FD Algorithm					ST-FD-R Algorithm					CPLEX
		LB	UB	FD Cuts	%Gap	# MIPs	LB	UB	FD Cuts	%Gap	# MIPs	C %Gap
1	KI.30.40.25a	-584.11	-578.87	32	0.90	4,545	-584.11	-578.87	47	0.90	7,002	0.92
2	KI.30.40.25b	-563.92	-558.99	47	0.87	6,688	-563.57	-558.99	50	0.81	7,329	1.04
3	KI.30.40.25c	-501.02	-496.05	44	0.99	6,477	-501.02	-496.05	47	0.99	7,215	1.15
4	KI.30.40.25d	-606.43	-601.25	32	0.85	6,287	-606.43	-601.25	38	0.85	7,662	1.00
5	KI.30.40.25e	-2566.43	-2393.42	160	6.74	13,073	-2,591.59	-2,566.77	175	0.96	14,551	1.18
	Average			63	2.07	7,414			71	0.90	8,751	1.06
6	KI.30.40.50a	-501.14	-495.42	87	1.14	11,338	-500.88	-495.42	104	1.09	13,957	1.43
7	KI.30.40.50b	-537.59	-535.06	39	0.47	7,404	-540.59	-535.06	41	1.02	7,758	1.08
8	KI.30.40.50c	-714.96	-708.57	72	0.89	11,115	-712.35	-705.36	82	0.98	12,839	1.05
9	KI.30.40.50d	-2,390.51	-2,372.8	291	0.74	28,228	-2,390.52	-2372.80	290	0.74	28,575	1.03
10	KI.30.40.50e	-2,143.38	-2,047.66	343	4.47	19,345	-2,143.38	-2,046.44	369	4.52	21,036	1.24
	Average			166	1.54	15,486			177	1.67	16,753	1.10
11	KI.30.40.100a	-687.99	-683.20	43	0.70	8,251	-687.99	-683.20	46	0.70	8,814	0.84
12	KI.30.40.100b	-431.61	-425.76	125	1.35	13,278	-431.61	-425.76	125	1.35	13,801	1.82
13	KI.30.40.100c	-539.86	-534.59	101	0.98	13,541	-539.86	-534.58	110	0.98	15,187	1.24
14	KI.30.40.100d	-787.87	-782.05	75	0.74	12,704	-786.67	-781.43	78	0.67	13,211	0.89
15	KI.30.40.100e	-575.38	-571.90	74	0.60	14,378	-575.62	-571.90	78	0.65	15,204	1.05
	Average			83	0.87	12,430			87	0.87	13,243	1.17
16	KI.30.40.150a	-549.46	-543.14	84	1.15	11,700	-554.09	-548.49	133	1.01	18,986	1.60
17	KI.30.40.150b	-712.58	-707.13	80	0.76	13,767	-712.58	-707.13	119	0.76	20,865	1.20
18	KI.30.40.150c	-615.62	-617.81	114	0.36	15,011	-617.41	-617.82	214	0.06	29,736	1.09
19	KI.30.40.150d	-482.34	-477.27	160	1.05	19,480	-481.93	-473.83	307	1.68	39,988	0.89
20	KI.30.40.150e	-555.72	-551.73	127	0.72	16,110	-556.75	-551.70	237	0.91	32,162	1.14
	Average			113	0.81	15,213			202	0.89	28,347	1.18
21	KI.30.40.200a	-613.86	-609.18	116	0.76	15,618	-618.41	-612.47	226	0.96	31,011	1.01
22	KI.30.40.200b	-633.17	-627.92	110	0.83	15,096	-632.70	-627.92	214	0.76	29,097	1.08
23	KI.30.40.200c	-701.63	-695.56	118	0.87	16,952	-705.40	-705.94	237	0.08	35,227	0.72
24	KI.30.40.200d	-527.71	-522.44	120	1.00	12,733	-527.64	-522.44	238	0.99	25,461	1.62
25	KI.30.40.200e	-745.98	-739.09	87	0.92	13,046	-745.98	-739.09	186	0.92	28,347	1.22
	Average			110	0.88	14,689			220	0.74	29,828	1.13

Table A.8: Set30.40 Computational results (using L^1 norm)

No.	Instance	ST-FD Algorithm					ST-FD-R Algorithm					CPLEX
		LB	UB	FD Cuts	%Gap	# MIPs	LB	UB	FD Cuts	%Gap	# MIPs	C %Gap
1	KI.30.40.150a	-552.44	-549.70	141	0.50	7,356	-555.25	-549.70	183	1.00	9,739	1.60
2	KI.30.40.150b	-710.71	-707.13	126	0.50	7,543	-712.10	-707.09	176	0.70	10,611	1.20
3	KI.30.40.150c	-622.91	-617.81	177	0.82	9,175	-622.91	-617.81	288	0.82	15,397	1.09
4	KI.30.40.150d	-482.29	-476.61	260	1.18	11,852	-481.90	-475.55	437	1.32	21,082	0.89
5	KI.30.40.150e	-555.47	-531.97	209	4.23	9,820	-556.35	-551.69	344	0.84	16,886	1.14
	Average			182	1.45	9,149			285	0.94	14,743	1.18
6	KI.30.40.150a	-615.44	-613.16	189	0.37	8,995	-613.86	-609.22	302	0.75	14,904	1.01
7	KI.30.40.150b	-633.17	-627.93	182	0.83	8,977	-632.66	-627.93	303	0.75	15,516	1.08
8	KI.30.40.150c	-711.42	-707.12	200	0.60	9,893	-694.84	-696.17	339	0.19	17,558	0.72
9	KI.30.40.150d	-527.48	-522.44	199	0.95	8,938	-527.63	-522.44	318	0.98	15,013	1.62
10	KI.30.40.150e	-745.98	-739.10	170	0.92	8,808	-745.26	-739.09	260	0.83	13,762	1.22
	Average			188	0.74	9,122			304	0.70	15,350	1.13

Table A.9: Set30.40 Computational results (using L^2 norm)