TEAM ORIENTEERING PROBLEM WITH
STOCHASTIC TIME-DEPENDENT TRAVEL
TIME

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Şifanur Çelik
July 2021
Team Orienteering Problem with Stochastic Time-dependent Travel Time
By Şifanur Çelik
July 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Özlem Çavuş İyigün (Advisor)

Bahar Yetiş Kara

Serhat Gül

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School
ABSTRACT

TEAM ORIENTEERING PROBLEM WITH STOCHASTIC TIME-DEPENDENT TRAVEL TIME

Şifanur Çelik
M.S. in Industrial Engineering
Advisor: Özlem Çavuş İyigün
July 2021

According to United Nations, human population living in urban areas is expected to increase in the coming years. This increase will have an effect on the traffic density in the urban areas. This motivates employees whose job is to visit customers during the day, such as logistics company employees, to consider the impact of traffic density on the travel times while visiting customers. This study aims to find prior optimal tours for more than one agent to visit customers and to maximize total expected profit within a given time limit while taking the uncertainties in travel times caused by traffic congestion into account. Agents are not required to visit every customer and the tour of each agent starts and ends at a certain depot node. It is assumed that the travel time to go from a customer to another customer is random and depends on the departure. We use a time-dependent travel time model that has first-in-first-out property while calculating the travel times. We propose a two-stage stochastic mixed-integer program to formulate the problem and suggest Integer L-shaped method in order to solve large-scale problem instances. In our computational study, we analyze the benefit of using stochastic solutions, and observe that Integer L-shaped method is superior to CPLEX in terms of computational time.

Keywords: Two-stage Stochastic Programming, Orienteering Problem, Integer L-Shaped Method, Time-dependent Stochastic Travel Time.

Anahtar sözcükler: İki Aşamalı Stokastik Model, Oryantiring Problemi, Tam Sayılı L-şekilli Yöntem, Zamana Bağlı Rassal Yolculuk Süresi.
Acknowledgement

I extend my gratitude to all the people I had a chance to have in my life, especially to the ones who provided the essential support during this two-year path.

First and foremost, I am definitely indebted to my advisor Asst. Prof. Özlem Çavuş for her advice, indescribable support, and enormous patience throughout my graduate study at Bilkent University. Without her guidance, I could not have come as far as I am right now. It was both a great privilege and an honor for me to have her as my advisor. I cannot thank her enough for the time that she dedicated to me.

Being a member of Bilkent IE family has changed me in a way that my vocabulary is way too insufficient to describe. It was an honor to have the opportunity to interact with each member of the department.

I am grateful to my family Mithat Çelik, Büşra Kaçan, Esra Akal, Rasime Kirbıyık, and Tuba Mavibaş for their support, patience, and eternal love that I have always felt throughout this journey. Their contribution was present in each and every step that I have taken. They provided the strength and resilience that I so much needed.

To have the friends I have is an unthinkable fortune, and the memories we have shared are a treasure no king has ever possessed. Whenever I feel down, they give me the wings with which the sky itself is no limit. They give me the inspiration that is enough to empower an exhausted outnumbered army for a decisive victory. Special thanks to Hande Özge Aydoğan, İdil Su Kaya, Özge Öztürk, Efe Sertkaya, Nilsu Uzunlar, Emre Düzoylum, Utku Çallak, İsmail Burak Taş and Eda Nur Acar. I could not have done this without you. I love you so much.

The last on this list, but surely not the least importance, would be 'Bluejay
Coffee House' - the provider of the best coffee in Bilkent and even better place to work. This thesis has been written there. Thanks to Mehmet, Takashi, Alihan, Eren, Ruslan, and Can for making this place special.
I would like to dedicate my thesis to my beloved mother who is always with me.
Contents

1 Introduction 1

2 Literature Review 6

2.1 Two-stage Stochastic Linear Programming with Recourse 6

2.1.1 Two-stage Stochastic (Mixed) Integer Programming with Recourse 7

2.2 Orienteering Problems 11

2.3 Time-dependent Travel Time in Optimization Models 18

3 Problem Definition and Formulation 19

3.1 Preliminary: Time-Dependent Travel Time Model 21

3.2 A Two-Stage Stochastic Model: Team Orienteering Problem with Stochastic Time-dependent Travel Times 26

4 Integer L-shaped Method 32

4.1 L-shaped Method 32
4.2 Integer L-shaped Method ........................................ 35
4.3 Integer L-shaped Method Applied to Our Problem .......... 37

5 Computational Study .............................................. 44

5.1 Experimental Design .......................................... 44
5.2 Experimental Results and Analysis ............................ 46
    5.2.1 Value of Stochastic Solution ............................. 48
    5.2.2 Stochastic and Deterministic Tours ..................... 52
    5.2.3 Integer L-shaped Algorithm .............................. 54

6 Conclusion and Future Work .................................... 62

A Results ........................................................... 72
List of Figures

1.1 Road condition on Ishinomaki ................................. 2
1.2 Average vehicle speed on main streets .......................... 3

3.1 Step Speed Function ........................................... 22
3.2 Continuous Piecewise Linear Function .......................... 25

5.1 Nodes and Associated Rewards ................................. 45
5.2 A sample tour containing 7 nodes, 1 agent having a time limit of 7.5 hours ........................................ 47
5.3 Value of Stochastic Solution for 1 agent ....................... 49
5.4 Value of Stochastic Solution for 2 agents ..................... 51
5.5 A priori tour for 10 nodes and 1 agent having 10 hours to complete the tour ........................................ 53
5.6 A priori tour for 9 nodes and 1 agent having 8 hours to complete the tour ........................................ 54
# List of Tables

5.1 Congestion Status of the Roads ........................................ 46
5.2 Computational Results of CPLEX and Integer L-shaped Method . 55
5.3 Analysis of Integer L-shaped Algorithm with 20 Scenarios, 2
    Agents and Time Limit of 5 Hours ................................. 57
5.4 Analysis of Integer L-shaped Algorithm with 20 Scenarios, 2
    Agents and Time Limit of 6 Hours ................................. 58
5.5 Analysis of Integer L-shaped Algorithm with 20 scenarios, 2 Agents
    and Time Limit of 7 Hours ........................................ 58
5.6 Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3
    Agents and Time Limit of 7 Hours ................................. 59
5.7 Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3
    Agents and Time Limit of 8 Hours ................................. 59
5.8 Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3
    Agents and Time Limit of 9 Hours ................................. 60
5.9 Analysis of Integer L-shaped Algorithm with 20 scenarios and 30
    customer nodes ...................................................... 61
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Value of Stochastic Solution-1</td>
<td>73</td>
</tr>
<tr>
<td>A.2 Value of Stochastic Solution-2</td>
<td>74</td>
</tr>
<tr>
<td>A.3 Value of Stochastic Solution-3</td>
<td>75</td>
</tr>
<tr>
<td>A.4 Value of Stochastic Solution-4</td>
<td>76</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Traffic congestion is a widespread problem in crowded cities due to rapidly growing urbanization and an increase in the need for transportation. According to the 2018 Revision of World Urbanization Prospects prepared by the Population Division of the UN Department of Economics and Social Affairs (UN DESA), human population living in urban areas is expected to rise to 68% from 55% by 2050 [1]. As people living in crowded cities face transportation problems, the possibility of an increase in traffic congestion alerts people to take action on this issue or consider congestion on their day-to-day decisions in the interest of long-term sustainability.

Traffic congestion increases people’s ineffective use of time, and for the companies, especially logistic companies, it is one of the main problems they face. Hence, comprehensive consideration of the traffic congestion helps them increase their profit and customer satisfaction by being on time. For example, one can consider the organization CARE, international relief and humanitarian organization. CARE works with several logistic companies in different areas. With this collaboration, they aim to maximize their emergency-response capacity and send needed help for the places that faced a natural disaster. Their goal is an effective and timely distribution of the help since the time is crucial, and they pursue reaching as many deprived people as possible [2]. In order to achieve this goal,
some decisions must be made. Those decisions can be related to the orienteering problem (OP). In OP, within the specified time limit, one or more vehicles start the tour from a starting point and try to visit places to collect rewards so that the total collected reward is maximum. The logistic companies who collaborate with CARE organization try to meet people’s needs and aim to maximize their visit to places where deprived people are present. The number of people in an affected area is considered as a reward to be collected. Since time is crucial for them, they try to find a path between places within the determined time limit. The condition of the road changes according to the nature and the location of the disaster. Figure 1.1 shows the road conditions during the Great East Japan Earthquake on March 11, 2011 [3].

Figure 1.1: Road condition on Ishinomaki

(a) Traffic jam before the flood

(b) Road condition after the flood

According to the study on people’s behaviour and traffic congestion after the Great East Japan Earthquake on March 11, 2011 [3], human evacuation behavior and road network conditions play an important role for the designing effective support plans and operations. Below, one can see that average travel speed changes depending on the departure time.
Figure 1.2: Average vehicle speed on main streets

Hence, consideration of time-dependent travel time in modeling can be more realistic and may provide better decisions making in the pursuit of achieving their goal.

OPs can be seen in different areas of life. Its application on the home fuel delivery problem is explained in Golden et al. [4]. The home fuel delivery problem is about a certain number of vehicles making a delivery to many customers throughout the day. The customers’ fuel stock has be kept at a certain level in any time of the day. The predicted stock level is taken as an indicator of urgency, which is considered the customers’ rewards, and the model is constructed as an OP. The ultimate goal is to decide upon the customers’ set to visit who require the fuel urgently. OP can also be implemented in Mobile Tourist Guide [5]. Those kinds of problems are called Tourist Trip Design Problems. In that problem, a tourist wants to travel between places that are worth seeing. The tourist has different preference levels for each location. The duration of the trip is limited. Hence, the decision maker has to form a path of different locations to maximize his/her utility. This problem can also be modeled considering the time windows for each location. This extension is strengthened by the idea that some locations are worth seeing at specific time periods; for instance, a beach
is aesthetic at sunset. The time-dependency nature of this problem is given by Gavalas et al. [6] with integrating public transportation to travel between places. Ilhan et al. [7] give an example of applications of OP by giving reference to Original Equipment Manufacturer (OEM). This company undergoes a change that makes the inventory at suppliers obsolete by eliminating a product line. They want to decide which suppliers to go to maximize the retrieved claims, which are the difference between the value said by the supplier and the audited stock level value. If the company consists of one auditor, the problem is formulated as an OP. In the mentioned applications, as the travel time changes according to the traffic congestion on the roads, considering time-dependency on departure time in the calculation of travel time makes the problem more applicable to real life. Keeping in mind that the road condition can be influenced by unpredictable events such as accidents, road constructions, etc., defining the speed as a random variable is reasonable to this extent. According to Malandraki and Daskin [8], variation in the travel times comes from two elements. The first one is based on the hourly or seasonal changes in the average traffic congestion level. This part of the variation could be estimated. The second element is random events such as instant weather changes, road constructions, or accidents. Focusing on the deterministic travel times may not be helpful in applications as travel time changes according to departure time and random events that influence the traffic level.

In this study, we present a two-stage stochastic linear mixed integer programming model that gives an optimal tour for each identical agent in the presence of travel time uncertainty. By considering the uncertainty in speed conditions, we aim to model a feasible tour in accordance with a time limit. The objective of our model is to maximize the expected total collected rewards of customer nodes in a graph by homogeneous agents. The randomness of the problem emerges from a real-life problem: traffic congestion that causes a change in travel time to determined places. In this thesis, we consider an exact solution method for a two-stage mixed integer stochastic optimization model. We propose an Integer L-shaped, which is an exact solution method, to tackle large-size problem instances. Our computational experiments analyze the efficiency of the CPLEX and Integer
L-shaped method and the benefit of solving a two-stage stochastic optimization model. We differ from the existing literature in terms of the following aspects:

- We propose a novel model to find a feasible tour that maximizes expected total collected reward considering time-dependency in stochastic travel time.
- We use two-stage stochastic programming to take uncertainty in the travel times into account. This approach is critical as we reflect the uncertainty in real life through scenarios each of which has different variations in the travel time.
- We propose an exact solution method named as the Integer L-shaped method to find optimal tours for large-scale instances.
- Our computational experiments provide valuable insights on the advantages of using an exact solution method and the benefit of modeling the problem as a two-stage stochastic problem.

The rest of this study is organized as follows. In Chapter 2, we review the relevant studies in the literature. In Chapter 3, we describe the problem, time-dependent travel time model and present our model. In Chapter 4, we give solution methodology for solving two-stage stochastic model. Chapter 5 provides the computational study about the model and the solution approach. Finally, in Chapter 6, we give our concluding remarks.
Chapter 2

Literature Review

In this chapter, we first introduce two-stage stochastic programming and continue with the two-stage stochastic mixed-integer programming formulation along with solution approaches present in the literature in Section 2.1. Then, we introduce Orienteering Problems and present a brief survey of existing variants of Orienteering Problems mainly based on deterministic and stochastic parameters. We also present existing solution methods utilized for OP, which can be categorized as near-optimal and exact solution methods. Lastly, we give a brief review of the use of time-dependent travel times in optimization models.

2.1 Two-stage Stochastic Linear Programming with Recourse

In this section, we provide a general formulation of two-stage stochastic programming with recourse. The notations used here are in parallel with Birge and Louveaux [9]. The pillar stone of the two-stage stochastic linear programming with recourse is created by Dantizg and Madansky [10], and Beale [11]. The problem has two main attributes, which are deterministic and stochastic. One is free of certainty, and the latter is subject to certainty. There are two sets of decision
variables which are referred to as first-stage and second-stage decision variables. First-stage decision variables are taken before the uncertainty in the problem is realized, while the second-stage decision variables are the ones taken after the uncertainty is observed. The general formulation of the problem presented in the literature is given below.

\[
\begin{align*}
\min & \quad z = c^T x + \mathbb{E}_\xi[\min\{q(\omega)^T y(\omega)\}] \\
\text{s.t.} & \quad Ax = b, \\
& \quad T(\omega)x + Wy(\omega) = h(\omega), \\
& \quad x \geq 0, y(\omega) \geq 0
\end{align*}
\]

where \( x \in \mathbb{R}^{n_1} \) is first-stage decision vector, \( c \in \mathbb{R}^{n_1}, b \in \mathbb{R}^{m_1} \) and \( A \in \mathbb{R}^{m_1 \times n_1} \) are the first-stage decision vectors and matrices related to \( x \). Let \( \Omega \) be the set of all scenarios and \( \omega \in \Omega \) be a realized random scenario, \( q(\omega) \in \mathbb{R}^{n_2}, h(\omega) \in \mathbb{R}^{m_2} \) and \( T(\omega) \in \mathbb{R}^{m_2 \times n_1} \) are random data realized for random scenario \( \omega \in \Omega \). \( W \in \mathbb{R}^{m_2 \times n_2} \) is a matrix related to second-stage decision vector \( y(\omega) \in \mathbb{R}^{n_2} \). We define random vector \( \xi(\omega) = (q(\omega)^T, h(\omega)^T, T_1(\omega), \ldots, T_{m_2}(\omega)) \) for \( \omega \in \Omega \). In the objective function, the first part containing the decision variable \( x \) is deterministic and the latter part is the expectation taken over all elementary events \( \omega \). The stochastic attributes could be from a continuous or discrete distribution. In this study, we assume that \( \xi \) is a discrete random vector. This choice is more preferred in purpose due to efficient solution methods.

### 2.1.1 Two-stage Stochastic (Mixed) Integer Programming with Recourse

In the literature, different extensions of two-stage stochastic programming have been studied. One of them is to have integer decision variables. The model (2.1)-(2.4) can be modified with the constraint below instead of the constraint (2.4).

\[
x \in \mathbb{R}^{n_1-s_1}_+ \times \mathbb{Z}^{s_1}_+,
\quad y(\omega) \in \mathbb{R}^{n_2-s_2}_+ \times \mathbb{Z}^{s_2}_+
\]

where \( 0 \leq s_1 \leq n_1 \) and \( 0 \leq s_2 \leq n_2 \). Moreover, some or all of the integer variables can be set to be binary. In this study, our formulation consists of integer variables
which are restricted to be binary variables.

In literature, it is stated that the models with integer variables and stochastic components are difficult to solve. Hence, a two-stage stochastic program with mixed-integer variables is not easily tackled. Ahmed [12] points out that three aspects make the two-stage stochastic program with integer variables challenging to solve.

1. Computing the recourse function for fixed first-stage decisions and specific realization of the random vector: This computation may involve solving second-stage problems that could be NP-hard problems. Therefore, it is computationally challenging.

2. Computing the recourse function for a fixed first-stage decisions: Consider the case where the random vector is generated from a continuous distribution. The assessment of the recourse function requires integration which is generally not possible for an integer program. If the random variables are generated from a discrete distribution, this leads to solving many similar integer programs. Hence, it is computationally challenging.

3. Computing the expected second-stage cost: If there are integer second-stage variables, the recourse function is non-convex and not continuous in general. Hence, this leads to some difficulties in computation.

One should note that a significant amount of theory and solution methods have been suggested for two-stage stochastic linear programs with continuous decision variables. The solution methods existing in the literature rely on decomposition and can be classified as primal (stage-wise) and dual (scenario-wise) decomposition algorithms. In general, primal decomposition algorithms are modified versions of Benders’ decomposition [13] and the L-shaped method [14]. The main difference relies on the approximation of the second-stage cost function. In dual decomposition algorithms, Lagrangian dual program is obtained by decomposing the problem into subproblems for each scenario \( \omega \in \Omega \). Since Lagrangian dual is a convex continuous model, the problem can be tackled by subgradient methods for fixed first-stage decision \( x \) [15].
The solution methods proposed in the literature are modified according to having integer first and second-stage decision variables. Wollmer [16] and Laporte and Louveaux [14] work on two-stage mixed-integer programming that has binary first-stage and continuous second-stage variables. Wollmer [16] proposes an implicit enumeration scheme that backtracks and searches through feasible solutions. Laporte and Louveaux. [14] present the integer L-shaped method, which benefits from the branch and bound method and L-shaped algorithm.

Carøe and Schultz [15] consider a maximization problem with mixed-integer variables not only at the first stage but also at the second stage. They apply dual decomposition with the branch and bound algorithm. The node found by branch and bound is solved by Lagrangian dual, in which the best solution gives an upper bound for the currently tackled problem. As they relax the non-anticipativity constraints, at bounding, the scenario solutions may differ. In that case, the average of the solution is taken, and it is rounded to reach a feasible solution with rounding heuristics they suggested in [15]. A new objective value is found using these new solutions, and the current best objective value is updated. The found value of the first-stage decision variable $x$ is used to create cuts for $x$.

Escudero et al. [17] work on a two-stage stochastic mixed 0-1 problem. The decision variables are restricted to be binary or continuous. They propose a new decomposition algorithm, a modified version of Lagrangian decomposition and referred to as cluster-based Lagrangian decomposition. They manage to give strong lower bounds for best solution of the problem. Their method decomposes the model into a set of scenario clusters by taking the non-anticipativity constraints into account implicitly.

The two-stage stochastic program with binary first-stage and mixed integer second-stage variables is studied by Ntaimo [18]. They introduce Fenchel decomposition, that is a variation of the cutting plane algorithm. The method considers the Fenchel cuts that are used in integer programming. The relaxed second-stage problem is solved by the L-shaped method. Afterward, cuts obtained by Fenchel decomposition are added to the subproblems, and subproblems are solved again. If the solutions of the subproblem are integers, subgradient cuts are formed and
added to the master problem. The variation of the model where the second stage has only integer variable is studied by Gade et al. [19]. A solution method that combines the Benders decomposition and Gomory cuts is introduced. In literature, a two-stage stochastic program with mixed-integer first-stage and integer second-stage variables are first studied by Carøe and Tind [20] and Ahmed et al. [21]. The fixed recourse is considered in [20]. They propose a solution method that combines the L-shaped algorithm and duality theory. They prove that the proposed algorithm converges to optimal value in finitely many iterations if the subproblems are solved using the branch and bound algorithm or Gomory’s fractional cutting plane algorithm. In the latter work [21], the proposed method reformulates the problem by variable transformation and exploits the branch and bound method. Variable transformation helps to eliminate discontinuity of the second-stage cost function at branching such that the exact value of the second-stage cost function is obtained at bounding.

Hemmecke and Schultz [22] introduce a new solution method that is applicable for the problems having integer variables at both stages. Their method is not based on any of the decomposition techniques mentioned above. They differ from the literature by decomposing the test instances of the problem. The same program is later considered by Kong et al. [23]. The random component of the problem is limited to the parameter on the right-hand side of the constraint (2.3) and the coefficient matrix is taken with integer values. An integer programming-based and dynamic programming-based algorithms are introduced.

In the literature, the two-stage stochastic program with binary first-stage variables and 0-1 mixed integer second-stage variables is also studied. Sen and Higle [24] employ primal decomposition on the same problem with fixed recourse. They relax the subproblems with disjunctive programming theory. They observe that scenarios have a common coefficient for second-stage decision variables in the generated cuts with this relaxation. Hence, a cut is valid for other scenarios as well. This notion is referred to as Common Cut Coefficient Theorem that is used to apply the disjunctive decomposition algorithm. This work is extended in Sen and Sherali [25]. They combine branch and bound method and disjunctive decomposition algorithm. Subproblems are solved using the branch and bound
method, and cuts are generated according to the disjunctive decomposition algorithm. Moreover, in [25], disjunctive programming with lift and project cuts is studied on the problem with continuous first-stage decision variables and 0-1 mixed second-stage variables. Sherali and Fraticelli [26] propose a variation of Benders decomposition, in which subproblems are tackled with reformulation linearization technique.

2.2 Orienteering Problems

The sports game called orienteering is the origin of the Orienteering Problem. Each player starts the game at a specified point and is obliged to return that point by visiting each checkpoint within a time limit. Each checkpoint has different scores, and players try to collect the score points as much as possible to win the game. OP can be considered as a combination of Knapsack Problem and Travelling Salesperson Problem.

Orienteering Problem is first introduced by Tsiligirides [27] in the literature. It can also be referred to as Selective Traveling Salesman Problem [28] and Maximum Collection Problem [29]. In OP, each customer (place) has a specific reward waiting to be collected, and the agent decides which places to visit in what order. The problem aims to maximize total collected reward within a time limit by providing a tour starting from and ending in a specific place. Each place can be visited at most once. However, the agent is not obliged to visit all the places. This is one of the parts where the Orienteering Problem differentiates from the traveling salesman problem. The general formulation of the problem is as follows. Consider a graph $G(N^+, A)$ where $N^+$ is the set of nodes including the starting and ending node named as the depot and $A$ denotes the set of arcs connecting the pair of nodes. Let $N$ be the set of nodes excluding the depot node. Each node $i \in N$ has a reward amount $r_i$. The model uses binary routing variable $x_{ij}$ as decision variable, indicating whether an agent uses the arc connecting pair of nodes $(i, j)$ where $i \in N^+, j \in N^+ \setminus \{i\}$ on his/her tour. Let $t_{ij}$ be the travel time from node $i \in N^+$ to node $j \in N^+ \setminus \{i\}$ and $u_i$ for $i \in N^+$ is defined as
auxiliary decision variable to eliminate any subtour created in the graph. Under deterministic travel time, the model aims to maximize total collected rewards from the specific places within a certain time limit denoted by $H$. Below, we provide the mathematical formulation of OP.

\[
\text{Max } \sum_{j \in N} \sum_{i \in N^+ \setminus \{j\}} r_j x_{ij} \quad (2.5)
\]

s.t.

\[
\sum_{j \in N} x_{0j} = \sum_{j \in N} x_{j0} = 1 \quad (2.6)
\]

\[
\sum_{i \in N^+ \setminus \{j\}} x_{ij} = \sum_{i \in N^+ \setminus \{j\}} x_{ji} \leq 1 \quad \forall j \in N \quad (2.7)
\]

\[
u_i - u_j + 1 \leq (1 - x_{ij})|N^+| \quad \forall i \in N, j \in N^+ \setminus \{i\} \quad (2.8)
\]

\[1 \leq u_i \leq |N^+| \quad \forall i \in N^+ \quad (2.9)
\]

\[
\sum_{j \in N^+ \setminus \{i\}} \sum_{i \in N^+} t_{ij} x_{ij} \leq H \quad (2.10)
\]

\[
x_{ij} \in \{0, 1\} \quad \forall i \in N^+, j \in N^+ \setminus \{i\} \quad (2.11)
\]

The formulation is obtained using typical constraints in vehicle routing with additional time limit restriction. Constraint (2.6) makes sure that the tour starts and ends in depot node. Constraint (2.7) implies that visiting every node is not obligatory and balances flow. Constraints (2.8) and (2.9) are subtour elimination constraints of Miller-Tucker-Zemlin formulation of the Travelling Salesman Problem [30]. Constraint (2.10) restricts the tour to be within the time limit. Lastly, we have domain restriction on $x_{ij}$.

In the literature, some exact solution approaches are proposed for OP. Laporte and Martelo [28] and Ramesh et al. [31] employ the branch and bound algorithm as an exact solution method to solve optimally the problem instances less than 20 and 150 nodes, respectively. In their work, problem instances up to 500 nodes are solved. Note that the OP is proved to be NP-hard by Golden et al. [4]. In other words, there is no solution method designed to solve the problem in a polynomial time. Hence, the exact solution algorithms consume lots of time. In order to benefit from time, some heuristics are developed for practical application. On
the other hand, Gendreau et al. [32] point out reasons on why developing a good heuristic for OP is hard. One of the reasons is the independence of the node’s reward and the time to require that reward. Therefore, selection process of the nodes is not performed through a reasonable scheme. However, many heuristics are proposed in the literature. Tsiligirides [27] introduces two algorithms, namely deterministic and stochastic algorithms. Deterministic algorithm benefits from the modification of the vehicle routing procedure proposed in Wren and Holliday [33]. The stochastic algorithm relies on generating many routes and selecting the best one that gives the higher objective value. Monte-Carlo method is used in the selection of the next node. Other examples of heuristics can be seen in the work of Chao et al. [34], Gendreau et al. [32], Tasgetiren [35], Liang et al. [36], and Schilde et al. [37].

OP is a sub case of Team Orienteering Problem (TOP) where the number of agents is more than one. TOP is first studied by Chao et al. [38]. The formulation of OP is extended as follows. Consider the sets and parameters defined for (2.5)-(2.11), and let $K$ be the set of agents that travel between nodes in $N^+$. Each agent $k \in K$ has to collect the rewards under the time limit $H$. The decision variable in (2.5)-(2.11) is modified with the index set $K$ and it is redefined as $x_{ijk}$ indicating whether an agent $k \in K$ uses the arc connecting pair of nodes $(i, j)$ where $i \in N^+, j \in N^+ \setminus \{i\}$ on his/her tour. Then, TOP model can be formulated as follows:
Max $\sum_{k \in K} \sum_{j \in N} r_j \sum_{i \in N^+ \backslash \{j\}} x_{ijk}$  

s.t. 

$1 \leq \sum_{j \in N} \sum_{k \in K} x_{0jk} = \sum_{j \in N} \sum_{k \in K} x_{j0k} \leq |K|$  

$\sum_{i \in N^+ \backslash \{j\}} \sum_{k \in K} x_{ijk} = \sum_{i \in N^+ \backslash \{j\}} \sum_{k \in K} x_{jik} \leq 1 \quad \forall j \in N$  

$u_i - u_j + 1 \leq (1 - x_{ijk})|N^+| \quad \forall i \in N, j \in N^+ \backslash \{i\}, k \in K$  

$1 \leq u_i \leq |N^+| \quad \forall i \in N^+$  

$\sum_{j \in N^+ \backslash \{i\}} \sum_{i \in N^+} t_{ij} x_{ijk} \leq H \quad \forall k \in K$  

$x_{ijk} \in \{0, 1\} \quad \forall i \in N^+, j \in N^+ \backslash \{i\}, k \in K$  

Constraint (2.13) ensures that at most $|K|$ agents can leave and return the source node by making sure that the tour starts and ends in the source node for each agent. Other constraints are similar to the ones in formulation (2.5)-(2.11).

Butt and Ryan [39] develop column generation to solve TOP. They manage to solve problems up to 100 nodes. Boussier et al. [40] introduce an exact method to solve TOP and TOP with time windows. In the latter variation, customer nodes have specific time windows in which agents can only visit the nodes in these windows. They combine column generation with branch and bound method. To improve the performance, they benefit from the acceleration technique named limited discrepancy search in [41]. To our knowledge, Chao et al. [38] are the first ones to work on a heuristic for TOP. This is an extension of their earlier work [34] on OP that proposes a five-step heuristic. Later on, Ke et al. [42] employ ant colony optimisation for TOP. A feasible solution is obtained at each ant and the quality of the solution advances with a local search procedure. The stopping condition is decided as reaching the maximum number of iterations. Recently, Souffria et al. [43] introduce greedy randomised adaptive search procedure with path relinking. According to their computational study, they manage to obtain
solutions that give relatively close results to optimal value of the problem.

In the mentioned studies up to here, the parameters of the problem are assumed to deterministic. On the other hand, these assumptions may not give the most profound results since in real-life applications, customers may have random service times and rewards, or traffic congestion may influence the course of the decided paths as travel times from one place to another place can be random. However, in the literature, most studies focus on deterministic OP, and stochastic OP is overlooked. Recently, increased attention has been given to stochastic OPs. Randomness in OP is studied with stochastic travel times, service times and/or rewards to collect.

To our knowledge, stochastic components of OP are first studied on by Tang and Miller-Hooks [44]. They consider OPs with stochastic service times, while the profits and the travel times are assumed to be deterministic. In their proposed model, the service time of each customer is a random variable that is sampled from a discrete distribution. They aim to maximize total collected reward with chance constraint implying that the probability of total tour time exceeding a determined time limit is less than a prespecified probability level. They employ the branch and cut algorithm to solve the problem exactly and a near-optimal solution method named as the construct and adjust method. The uncertainty in rewards are first considered by Ilhan et al. [7]. The aim of the suggested model is to maximize the probability that the total collected reward will be greater than a decided reward amount. They propose an exact solution method and a bi-objective genetic algorithm to tackle the problem. Later on, Gupta et al. [45] consider the case where the random rewards can be in relation to waiting time. Their aim is decide a path adaptively so that the total expected reward would be maximum. They propose a algorithms for the non-adaptive and adaptive stochastic problems.

To our knowledge, stochastic travel time and service time in the OPs are first studied by Teng et al. [46]. Their objective is to maximize expected profit, in which they include penalty cost for exceeding the time limit. They propose a two-stage stochastic program with recourse and suggest the integer L-shaped method
for solving it. Later on, Campbell et al. [47] work on a similar problem except that in this time, penalty cost is incurred if the rewards in the nodes are not collected before the determined deadline. They employ a variable neighborhood search heuristic, and compare its performance with dynamic programming. Later on, Papanagiotou et al. [48] study OP with stochastic travel time and service time. They benefit from Monte-Carlo simulation with a hybrid objective function to approximate the optimal solution with minimal loss in accuracy. Papanagiotou et al. [49] work on a similar problem, except in this time, a penalty score is given if the selected customer is not visited. They propose a metaheuristic based on a sampling approximation with Monte Carlo simulation. Later, in [50], a hybrid sampling-based evaluator is worked on for the same problem. They present different techniques to evaluate objective function that combines Monte Carlo sampling, computation of deterministic objective function value and analytical evaluation method from [51].

Evers et al. [52] work on OP with stochastic weights by formulating it as a two-stage stochastic model with recourse where the time limit constraint must be satisfied by any feasible solution of the model. In [52], weights are related to travel expenses, travel time or fuel that is consumed while traveling between nodes. They employ sample average approximation with Monte Carlo simulation to find a feasible solution. Their proposed method is successful for small-size problem instances but not for large ones. They propose a heuristic for large-size problem instances, and obtain higher expected profit by applying the heuristic for the stochastic model rather applying it for the deterministic model. Zhang et al. [53] consider the stochastic OP with time windows on a network of queues. In their work, they consider the waiting time in the queues to be a random variable depending on the arrival time. They construct the tours considering two recourse actions. There are decisions on skipping the node depending on the arrival time and how long the person should wait in the queues. They propose a variable neighborhood search that is a slight variation of the one proposed in [47]. They present lower and upper bounds for the optimal value of the problem by solving the deterministic version with a dynamic programming approach. Dolinskaya et al. [54] study on adaptive OP with stochastic travel times where the agent can
dynamically change the path between customer nodes depending on the observed travel time. They propose three models with different degrees of adaptive decision making. The first one is the base model in which the agent’s decision on the path is unchanged even if the travel time is realized. The only information is the distribution of travel time on the edge. Second one is referred to as Level 1 model where all paths are decided before observing the uncertainty and updated dynamically using the realized travel time. The last one is referred to as Level 2 model in which there is no priori path decision, and the agent dynamically adapts the path as he/she learns more information about the network during the travel. In none of these papers, time-dependent stochastic travel times are not considered.

OP with stochastic independent travel time is popular in the literature, however, there is a limited number of studies on OP with stochastic time-dependent travel times. Stochastic time-dependent travel times in the OP are first introduced by Fomin and Lingas [55]. They present $(2 + \epsilon)$—approximation algorithm to solve the model. Fomin and Lingas [55] consider equal rewards for the nodes; hence the problem is reduced to maximize the total number of visited nodes. OP with time-dependent stochastic travel time with different rewards is studied by Lau et al. [56]. They employ a hybrid variable neighborhood search and simulated annealing as a solution methodology. Their work is extended by Varakantham and Kumar [57]. They propose a method that is based on the sample average approximation technique, which presents approximate solutions to large-sized problems. Later, Verbeeck et al. [58] study on OP with stochastic time-dependent travel times by introducing time windows. They employ iterated local search meta-heuristic to solve TOP with time windows. The solution method is presented as effective, fast and straightforward. This algorithm is a modification of the ant colony heuristic and benefits from a time-dependent local search procedure. Lau et al. [59] introduce a dynamic and stochastic OP model with time-dependent travel times. In that work, they suggest a risk-sensitive criterion that can be used for various risk choices. They employ a local search algorithm to solve dynamic and stochastic OP with risk-sensitive criteria. Mei et al. [60] propose multi-objective time-dependent orienteering problem. In the
proposed model, each node has a reward list indicating its degrees of desirability. In their application, they consider touristic places and set desirability scores accordingly. Two meta-heuristics are studied and it is shown that both heuristics obtain high quality solutions than an existing multi-objective evolutionary algorithm.

2.3 Time-dependent Travel Time in Optimization Models

Time-dependency into models is first introduced by Bowman [61] in scheduling problems. Then, Malandraki and Daskin [8] introduce it to the vehicle routing problem (VRP). They develop a mixed-integer linear programming model for the vehicle routing problem with time-dependent travel time and time windows (TD-VRP). Park [62] formulates this problem as a multi-objective problem. Ichoua et al. [63] and Donati et al. [64] work on solving TDVRP with tabu search and ant colony optimization. Ichoua et al. [63] then write the first paper that introduces a FIFO property into the time-dependent vehicle routing problem. FIFO property ensures that if a vehicle travels from node $i$ to node $j$, later departure from node $i$ implies later arrival to node $j$. Therefore, this construction of the time-dependent travel time model forms arrival times, a strictly monotonic function of the starting time. After this point, FIFO property is started to be used in the problems that consider time-dependency. Huang et al. [65] work on TDVRP with path flexibility considering fuel consumption. They include FIFO property in their model, making use of step speed functions and continuous piecewise linear function proposed in [63]. They employ the route-path approximation method to find out better solutions of the formulated problem. Afterward, Çimen and Soysal [66] study on time-dependent green vehicle routing problem with stochastic vehicle speeds. They consider the problem as a Markov decision process and solve it using a dynamic programming-based heuristic.
Chapter 3

Problem Definition and Formulation

We first provide a general description of Team Orienteering Problem with Time-dependent Stochastic Travel Time (TOP-TST). A group of agents travels between customer locations to collect the rewards, which are determined specifically according to the context of the problem. Those agents can be fleet of trucks of a logistic company or sales representatives of some other companies. The customer locations are set according to the traditional setting of vehicle routing problems. Consider a graph $G(N^+, A)$ defined in Section 2.2 and remember that the depot node is the origin node where each agent starts and ends the tour. The paths between all nodes are presented with the set $A$, and all of the nodes present in the graph $G$ are connected for this application. Each customer $i \in N$ has a reward amount $r_i$ and we have $K$ identical agents, each of which collects the amount $r_i$ whenever customer $i$ is visited. Travel time from node $i$ to node $j$ is random and dependent on the departure time from node $i$. Agents are obliged to collect rewards from the nodes starting from and returning to the depot node. The tours of the agents have to be completed within the specified time limit denoted by $H$. The tours of the agents are considered as feasible if they start from the depot node, collect rewards from each node by visiting the nodes exactly once, and end at the depot node. Our aim is to maximize expected total reward
by finding feasible tours subject to some constraints, which consist of traditional constraints of vehicle routing, time and domain restrictions.

The agent forms a prior tour before the uncertainty on travel times is resolved. The traffic condition of the road segment is seen after the agent starts to travel. The agent can change the course of the path according to the speed specification of the arc by deciding which nodes to cancel in a prior tour. The cancellation of some of the nodes makes sure that the agent can return back to the depot node without exceeding the time limit. Forming the TOP with a prior tour and adding another stage of decisions of which nodes to quit after the realization of the travel time can come across in real life with the applications in the sector of fast moving consumer goods (FMCG). In their traditional setting, sales representatives acquire the list of customers to visit at the beginning of their shifts. On the other hand, the travel time is not deterministic throughout the day, and it depends on the departure time of leaving one customer and heading towards another customer due to traffic congestion and the random events on the roads such as accidents or weather events. Hence, the sales representatives are not able to visit all of the customers on the list during their shifts and decide not to visit some of the customers; in other words, to quit.

To this end, we assume that agents’ capacity to collect rewards is unlimited for this application. We have no waiting time in the nodes, and the time to collect the reward in each node by each agent is assumed to be zero. Hence, we only consider the stochastic travel time for the problem presented in this thesis.

In this chapter, we first present a time-dependent travel time model in Section 3.1 that will be used in modeling Team Orienteering Problem with Time-dependent Travel Time described in Section 3.2.
3.1 Preliminary: Time-Dependent Travel Time Model

In this section, we describe the structure of a time-dependent travel time model and present necessary calculations to form the fundamental of stochastic time-dependent travel time models. Below, we provide the mathematical notation used throughout this section in addition to the ones defined previously.

Sets

\[ C \] : Set of time periods with constant speed
\[ M_{ij} \] : Set of breakpoints of the piecewise linear travel time function for the pair of nodes \((i, j)\) where \(i \in N^+ \) and \(j \in N^+ \setminus \{i\}\)

Parameters

\( v_c \) : Speed in time period \(c, c \in C\)
\([h^{c-1}, h^c] : c^{th} \) interval of time horizon in which the speed is constant, \(c \in C\)
\( d_{ij} \) : Distance between node \(i\) and node \(j\) where \(i \in N^+ \) and \(j \in N^+ \setminus \{i\}\)
\( \tau_{ij}(\cdot) \) : Piecewise linear travel time function of the pair of nodes \((i, j), i \in N^+ \) and \(j \in N^+ \setminus \{i\}\)
\([b^m_{ij}, b^{m+1}_{ij}] : m^{th} \) interval of time horizon in which \(\tau_{ij}(\cdot)\) is linear, where \(i \in N^+, j \in N^+ \setminus \{i\}\) and \(m \in M_{ij}\)
\( \alpha^m_{ij} \) : Slope of the piecewise linear function within interval \([b^m_{ij}, b^{m+1}_{ij}]\), \(i \in N^+, j \in N^+ \setminus \{i\}\) and \(m \in M_{ij}\)
\( \beta^m_{ij} \) : Travel time intercept of the piecewise linear function within interval \([b^m_{ij}, b^{m+1}_{ij}]\), \(i \in N^+, j \in N^+ \setminus \{i\}\) and \(m \in M_{ij}\)

This model is motivated by the notion that when an agent covers a ground between nodes \(i \in N^+ \) and \(j \in N^+ \setminus \{i\}\), the speed of the agent may not be
constant over whole distance $d_{ij}$. The travel time dependency on the departure
time from the nodes is conveyed by setting several intervals of the time horizon
in which the speed level is changed across the intervals however stays constant
within the interval while it takes different values as we move on to the next time
interval. We denote the set of time periods by $C$. Note that it is assumed that
all of the agents will have constant speed $v_c$ in each time period $c \in C$ to cover
the ground $d_{ij}$ between nodes $i \in N^+$ and $j \in N^+ \setminus \{i\}$. Therefore, we can view
travel speed as a step function of time. For better understanding, an example is
given below.

**Example 1.** We consider arbitrary nodes $i \in N^+$ and $j \in N^+ \setminus \{i\}$ with $d_{ij} = 1$,
and a time horizon of $[0,4]$ which is divided into 4 time periods, that is $C = \{1, 2, 3, 4\}$ and $v_1 = 2, v_2 = 1.25, v_3 = 2.5, v_4 = 1.75$. Figure 3.1 provides the
corresponding speed function.

![Figure 3.1: Step Speed Function](image)

Observe that if an agent departs from node $i$ at $t \in [0,0.5)$, he/she can cover
the whole distance $d_{ij} = 1$ with a speed of 2 units. However, if the agent departs
from node $i$ at $t \in [0.5,1)$, the speed of the agent will not be constant, which
suggests that the total travel time depends on the departure time of the agent
from node $i \in N^+$. To this end, the travel time function proposed in [63] is
used for modeling the time-dependent travel time in this study. They specify this function as a piecewise linear continuous function of the time which satisfies the *first-in-first-out* (FIFO) property. The FIFO property ensures that if an agent leaves a node $i$ for a node $j$ at a given time, later departure from node $i$ implies later arrival to node $j$. For more information, the reader is referred to [63].

Travel time function $\tau_{ij}(\cdot)$ outputs the total travel time from node $i$ to node $j$ depending on the departure time from node $i$. In order to evaluate $\tau_{ij}(\cdot)$ for the path connecting the nodes $i$ and $j$, we make use of speed levels $\{v_1, \ldots, v_{|C|}\}$ and their corresponding time components $\{h^0, \ldots, h^{(|C|)}\}$ in the step speed function, for instance, in the Example 1, the points $\{h^0, \ldots, h^{(|C|)}\}$ are set as $h^0 = 0, h^1 = 1, h^2 = 2, h^3 = 3, h^4 = 4$. Since the speed function is a step function, travel time function $\tau_{ij}(\cdot)$ is a piecewise linear function. It consists of $M_{ij}$ intervals defined by a finite index set of time component of breakpoints $[b^m_{ij}, b^{m+1}_{ij}], m = 1, \ldots, M_{ij}$. Within each interval, travel time function $\tau_{ij}(\cdot)$ is linear. Let $t^m_{ij}$ be the travel time from node $i$ to node $j$ if the agent departs at $b^m_{ij}$. The algorithm presented below is used to find the breakpoints $(b^m_{ij}, t^m_{ij}), m \in M_{ij}$.
Algorithm 1 Finding Breakpoints of Piecewise Linear Travel Time Function

$m \leftarrow 1$

for each speed time period $c \in C$ do

$d \leftarrow d_{ij}$

t' $\leftarrow \frac{d}{v_c}$

if $t' < (h^c - h^{c-1})$ then

$b_{ij}^m \leftarrow h^{c-1}$

t_{ij}^m \leftarrow t'$

$m \leftarrow m + 1$

$b_{ij}^m \leftarrow (h^c - t')$

t_{ij}^m \leftarrow t'$

$m \leftarrow m + 1$

else

$k \leftarrow c - 1$

t' $\leftarrow h^k + \frac{d}{v_k}$

t'' $\leftarrow h^k$

while $t' > h^k$ do

$d \leftarrow d - v_k(h^k - t'')$

t'' $\leftarrow h^k$

t' $\leftarrow t'' + \frac{d}{v_k}$

$k \leftarrow k + 1$

end while

$b_{ij}^m \leftarrow h^{c-1}$

t_{ij}^m \leftarrow (t' - h^{c-1})$

$m \leftarrow m + 1$

end if

end for

return $\{b_{ij}^1, b_{ij}^2, \ldots\}, \{t_{ij}^1, t_{ij}^2, \ldots\}$
In Algorithm 1, we let $t'$ to be the total travel time, to cover the distance $d_{ij}$ between the nodes $i \in N^+$ and $j \in N^+ \setminus \{i\}$ with a speed level $v_c, c \in C$. The index $m$ is used to specify the breakpoints of the travel time function, and the number of breakpoints is unknown at the beginning of the algorithm. Additionally, we let $t''$ to be the current time of travel. Firstly, it is checked whether the travel time $t'$ exceeds the length of the time interval in which the speed is constant and equal to $v_c, c \in C$. If it does not exceed, the first time component of the breakpoints is identified as $h^{c-1}$ and the travel time component of the breakpoints equals to $t'$. The speed level will not change if the agent departs within the time interval $[h^{c-1}, h^{c-1} - t']$. Hence, we identify the time component of the breakpoints as $b_{ij}^m = h^{c-1}$ and $b_{ij}^{m+1} = h^{c-1} - t'$ and travel time component of the breakpoints as $t_{ij}^m = t_{ij}^{m+1} = t'$. Then, we continue with the next time period $(c + 1) \in C$.

If the travel time $t'$ exceeds the length of the time interval in which the speed is constant and equal to $v_c, c \in C$, we try to find how many speed zones the agent crosses with the help of while loop. The distance is updated in each loop iteration as the distance between nodes $i$ and $j$ will be covered with different speed levels. Once the while condition is not satisfied, we exit the loop and continue with the next time period. In Figure 3.2, we provide the travel time function $\tau_{ij}(\cdot)$ corresponding to Example 1.

![Figure 3.2: Continous Piecewise Linear Function](image-url)
Now, once we construct the piecewise linear travel time function for each pair of nodes $i \in N^+ \text{ and } j \in N^+ \setminus \{i\}$ using Algorithm 1, we are able to find out the travel time depending on the departure time from node $i$. Remember that $M_{ij}$ denote the finite index set of breakpoints of time such that within each interval, the travel time function from node $i \in N^+$ to node $j \in N^+ \setminus \{i\}$ is linear. We have also defined $t_{ij}^m$ as the travel time if we depart node $i \in N^+$ at time $b_{ij}^m$ where $m \in M_{ij}$. Then travel time, $\tau_{ij}(w)$, when the agent departs from node $i \in N^+$ at time $w \in [b_{ij}^m, b_{ij}^{m+1}]$ is calculated as follows:

$$\tau_{ij}(w) = \frac{t_{ij}^{m+1} - t_{ij}^m}{b_{ij}^{m+1} - b_{ij}^m} (w - b_{ij}^m) + t_{ij}^m$$

$$= \frac{t_{ij}^{m+1} - t_{ij}^m}{b_{ij}^{m+1} - b_{ij}^m} w - \frac{t_{ij}^{m+1} - t_{ij}^m}{b_{ij}^{m+1} - b_{ij}^m} b_{ij}^m + t_{ij}^m$$

$$= \alpha_{ij}^m w + \beta_{ij}^m$$

where $\alpha_{ij}^m$ denotes the gradient of the piecewise linear function within interval $[b_{ij}^m, b_{ij}^{m+1}]$ and $\beta_{ij}^m$ denotes the intercept of the piecewise linear function within interval $[b_{ij}^m, b_{ij}^{m+1}]$.

### 3.2 A Two-Stage Stochastic Model: Team Orienteering Problem with Stochastic Time-dependent Travel Times

Team Orienteering Problem with Stochastic Time-dependent Travel Times (TOP-TST) considered in this work can be defined as follows. We assume that the travel time from node $i \in N^+$ to node $j \in N^+ \setminus \{i\}$ is dependent on the departure time from node $i$ with deterministic rewards and no service time. The time-dependency structure of the travel time is modeled according to the formulation in Section 3.1.

Under stochastic time-dependent travel time, the team orienteering problem
is formulated as a two-stage stochastic mixed integer program. In two-stage models, there are two types of decisions; first-stage and second-stage decisions. If the decisions are made before the uncertainty is realized, those decisions are called the first-stage decisions. Additionally, if the decisions are taken after the uncertainty is resolved, those decisions are referred to as second-stage decisions. Conceptually, in the first stage, each agent \( k \in K \) has to decide on a prior tour, in other words, which nodes to visit at what order. After the travel time is realized, in the second stage, which nodes to quit in a prior tour is decided according the specified time limit.

Below, we provide the mathematical notation used throughout this section in addition to the ones defined previously.

**Sets**

- \( N^+ \): Set of customer nodes including depot node
- \( N \): Set of customer nodes excluding depot node
- \( A \): Set of arcs connecting pairs of customer nodes
- \( K \): Set of agents
- \( \Omega \): Set of scenarios
- \( M_{ij} \): Set of intervals of the piecewise linear travel time function for the pair of nodes \((i, j)\) where \( i \in N^+ \) and \( j \in N^+ \setminus \{i\} \) defined by a finite set of breakpoints

**Parameters**

- \( L \): The beginning of the timeline
- \( H \): Time limit
- \( B \): Relatively large number
- \( r_i \): Reward collected from customer \( i \in N \)
- \( \alpha_{ij}^m \): Slope of the piecewise linear function within interval \([b_{ij}^m, b_{ij}^{m+1}]\), \( i \in N^+ \), \( j \in N^+ \setminus \{i\} \) and \( m \in M_{ij} \)

27
\( \beta_{ij}^m \) : Intercept of the piecewise linear function within interval 
\([b_{ij}^m, b_{ij}^{m+1}]\), \( i \in N^+, j \in N^+ \setminus \{i\} \) and \( m \in M_{ij} \)

\( p(\omega) \) : Probability of scenario \( \omega \in \Omega \)

**Decision Variables**

**First-stage Decision Variables**

\( x_{ijk} \) :

\[
\begin{cases} 
1, & \text{if the arc } (i, j) \in A \text{ is used in a priori tour by agent } k \in K \\
0, & \text{otherwise}
\end{cases}
\]

\( u_i \) : Position of node \( i \) used in the tour, for \( i \in N \)

**Second-stage Decision Variables**

\( y_{ijk}(\omega) \) :

\[
\begin{cases} 
1, & \text{if the arc } (i, j) \in A \text{ used in a priori tour by agent } k \in K \text{ is cancelled under scenario } \omega \in \Omega \\
0, & \text{otherwise}
\end{cases}
\]

\( z_{ijkm}(\omega) \) :

\[
\begin{cases} 
1, & \text{if the departure time from node } i \in N^+ \text{ to node } j \in N^+ \setminus \{i\} \text{ is in the interval } m \in M_{ij} \text{ under scenario } \omega \in \Omega \\
0, & \text{otherwise}
\end{cases}
\]

\( w_{ijkm}(\omega) \) : Actual departure time of agent \( k \in K \) from node \( i \in N^+ \) to node \( j \in N^+ \setminus \{i\} \) in the interval \( m \in M_{ij} \) under scenario \( \omega \in \Omega \)

\( q_{ik}(\omega) \) : Auxiliary binary decision variable for denoting the node \( i \in N^+ \) in which an agent \( k \in K \) exceeds the time limit by traveling to that node under scenario \( \omega \in \Omega \)
Below, we provide the mathematical formulation of a two-stage stochastic mixed integer program.

**Model**

\[
\text{max} \quad \sum_{k \in K} \sum_{j \in N} r_j \sum_{i \in N^+ \setminus \{j\}} x_{ijk} - \sum_{\omega \in \Omega} \left( p(\omega) \sum_{k \in K} \sum_{j \in N} r_j \sum_{i \in N^+ \setminus \{j\}} y_{ijk}(\omega) \right) \\
\text{s.t.}
\]

\[
1 \leq \sum_{j \in N} \sum_{k \in K} x_{0jk} = \sum_{j \in N} \sum_{k \in K} x_{j0k} \leq |K| \\
\sum_{i \in N^+ \setminus \{j\}} \sum_{k \in K} x_{ijk} = \sum_{i \in N^+ \setminus \{j\}} \sum_{k \in K} x_{jik} \leq 1 \quad \forall j \in N \\
u_i - u_j + 1 \leq (1 - x_{ijk})|N^+| \quad \forall i \in N, j \in N^+ \setminus \{i\}, k \in K \\
1 \leq u_i \leq |N^+| \quad \forall i \in N^+ \\
\sum_{m \in M_{ij}} z_{ijkm}(\omega) = x_{ijk} \quad \forall i \in N^+, j \in N^+ \setminus \{i\}, k \in K, \omega \in \Omega \\
y_{ijk}(\omega) \leq x_{ijk} \quad \forall i \in N^+, j \in N^+ \setminus \{i\}, k \in K, \omega \in \Omega \\
w_{0jkm}(\omega) = Lz_{0jkm}(\omega) \quad \forall j \in N, k \in K, m \in M_{ij}, \omega \in \Omega \\
b_{ij}^m(\omega)z_{ijkm}(\omega) \leq w_{ijkm}(\omega) \leq b_{ij}^{m+1}(\omega)z_{ijkm}(\omega) \quad \forall i \in N, j \in N^+ \setminus \{i\}, m \in M_{ij}, k \in K, \omega \in \Omega \\
\sum_{i \in N^+ \setminus \{j\}} y_{ijk}(\omega) \leq \sum_{i \in N^+ \setminus \{j\}} y_{ijk}(\omega) \quad \forall j \in N, k \in K, \omega \in \Omega \\
\sum_{i \in N^+ \setminus \{j\}} \left( z_{ijkm}(\omega)\beta_{ij}^m(\omega) + \alpha_{ij}^m(\omega)w_{ijkm}(\omega) \right) \\
+ w_{ijkm}(\omega) = \sum_{i \in N^+ \setminus \{j\}} \sum_{m \in M_{ij}} w_{ijkm}(\omega) \quad \forall j \in N, k \in K, \omega \in \Omega
\]
\[
\sum_{i \in N^+ \setminus \{j\}} \left( z_{ijkm}(\omega) \beta^m_{ij}(\omega) + \alpha^m_{ij}(\omega) w_{ijkm}(\omega) \right) \\
+ w_{ijkm}(\omega) + \sum_{i \in N^+ \setminus \{j\}} \left( z_{jikm}(\omega) \beta^m_{ij}(\omega) \right) \\
+ \alpha^m_{j0}(\omega) w_{jikm}(\omega) \right) \leq H + Bq_{jk}(\omega) \quad \forall j \in N^+, k \in K, \omega \in \Omega \tag{3.12}
\]

\[
\sum_{i \in N^+ \setminus \{j\}} y_{jik}(\omega) \geq 1 - B(1 - q_{jk}(\omega)) \quad \forall j \in N, k \in K, \omega \in \Omega \tag{3.13}
\]

\[
x_{ijk}, y_{jik}(\omega), z_{ijkm}(\omega), q_{jk}(\omega) \in \{0, 1\} \quad \forall i \in N^+, j \in N, k \in K, \omega \in \Omega \tag{3.14}
\]

\[
w_{jikm}(\omega) \in \mathbb{R}^+ \quad \forall i \in N^+, j \in N^+, k \in K, \omega \in \Omega \tag{3.15}
\]

The model maximizes the total expected reward collected. The first component of (3.1) is deterministic and the other term evaluates the expected total reward not collected due to canceled arcs. Constraint (3.2) ensures that each agent starts the tour and ends at that node from the depot node and there are no more than |K| agents collecting the rewards. Constraint (3.3) is for flow balance and making sure that a node can be visited at most once. Constraints (3.4) and (3.5) are subtour elimination constraints. Constraint (3.6) restricts each agent to depart within exactly one time interval. Constraint (3.7) ensures that an arc cannot be canceled unless it is in a prior tour. Constraint (3.8) does not allow agents to depart from the depot node at the beginning of the time horizon. Constraint (3.9) restricts each departure time from node i to be within one time interval. Constraint (3.10) ensures that if an arc that is coming out from any node j is not canceled, then we cannot cancel an arc that is coming into this node cannot be either. Constraint (3.11) sets the time continuity of a person traveling in the network, that is, the departure time from node j equals the departure time of the agent k traveling from node i to node j plus the travel time from node i to node j. Constraints (3.12) and (3.13) ensure that the whole tour is completed within the time limit by specifying the nodes that are cancelled under a scenario. In constraint (3.12), for each node j, if total travel time to node j plus
travel time from node $j$ to depot node is higher than the time limit, then with constraint (3.13), that node has to be cancelled. Constraints (3.14) and (3.15) are the domain restrictions.
Chapter 4

Integer L-shaped Method

As the complexity of a problem increases, it gets computationally challenging to find an optimal solution of the problem. This can be because of the high solution time or the memory restriction of the environment where the problem is solved. When the two-stage stochastic integer programming are considered, they can be seen as computationally intractable [14]. Therefore, some exact solution methods are proposed in the literature in order to decrease the solution time and easily tackle the problem.

This chapter explains the Integer L-shaped method that is proposed as an exact solution method in this study. In Section 4.1, we give a brief information on the L-shaped method. Then, we continue with the description of the Integer L-shaped method in Section 4.2. Finally, we present the Integer L-shaped method applied to our problem in Section 4.3.

4.1 L-shaped Method

Algorithms for solving the two-stage linear stochastic programs under uncertainty are first developed by Dantzig and Madansky [10], and Wets and Van Slyke [67]. They consider the stochastic problems which have continuous second-stage
decision variables and discrete random variables. The L-shaped method proposed by Wets and Van Slyke \cite{67} exploits the block-angular structure of the two-stage stochastic models. In other words, scenarios are independent of each other; hence, the problem is decomposable over the scenarios. Making use of this nice property, we are able to create a master program and several subproblems, which are second-stage problems for each scenario $\omega \in \Omega$ that are solved given the first-stage decision variables.

L-shaped method aims to approximate the recourse function $Q(x)$ which denotes the expected value of all second-stage problems. Now, for the ease of the explanation, consider the two-stage stochastic program (2.1)-(2.4) in Section 2.1 reformulated as below. The notations used here are in parallel with \cite{68}.

\[
\begin{align*}
\min & \quad c^T x + Q(x) \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

where

\[
Q(x) = \sum_{\omega \in \Omega} p(\omega)Q(x, \omega)
\]

and

\[
Q(x, \omega) = \min \{q(\omega)^T y : Wy = h(\omega) - T(\omega)x, y \geq 0\} \quad \forall \omega \in \Omega
\]

Notice that the dual of the second-stage subproblem (4.5) for a scenario $\omega \in \Omega$ can be written in below with associated dual decision variable $\alpha$:

\[
\begin{align*}
\max & \quad (h(\omega) - T(\omega)x)^T \alpha \\
\text{s.t.} & \quad W^T \alpha \leq q(\omega)
\end{align*}
\]

Let $u_{\omega j}, j \in U_\omega$ be the set of vertices of the set $\{\alpha : W^T \alpha \leq q(\omega)\}$, for $\omega \in \Omega$ and $v_{\omega j}, j \in V_\omega$ be a collection of extreme rays of the set $\{\alpha : W^T \alpha \leq q(\omega)\}$.
for $\omega \in \Omega$. Then the first-stage decision variable $x$ is feasible if and only if $v_{\omega j}(h(\omega) - T(\omega)x) \leq 0$, for $j \in V_\omega$, $\omega \in \Omega$. In addition, as the dual second-stage problem is a linear problem, an optimal solution is attained at one of the vertices $u_{\omega j}$, $j \in U_\omega$. Remembering (4.4), and considering the problem (4.6)-(4.7), we can formulate the problem (4.8)-(4.9) introducing a continuous decision variable $\theta$ for $Q(x)$:

$$\min \theta \quad \text{subject to} \quad \sum_{\omega \in \Omega} p(\omega) u_{\omega j}(h(\omega) - T(\omega)x) \leq \theta, \quad \text{for } j \in U_\omega, \omega \in \Omega$$

Therefore, the expression $\sum_{\omega \in \Omega} p(\omega) u_{\omega j}(h(\omega) - T(\omega)x)$ provides a lower bound for (4.4).

A general structure of the master program used in the L-shaped method is given below.

$$\min \quad c^T x + \theta \quad \text{subject to} \quad Ax = b,$$

$$D_k^T x \geq d_k, \quad k = 1, \ldots, s,$$

$$E_l^T x + \theta \geq e_l, \quad l = 1, \ldots, t,$$

$$x \geq 0, \quad \theta \in \mathbb{R}$$

where

$$D_k^T = v_{\omega jk} T(\omega)$$

$$d_k = v_{\omega jk}^T h(\omega)$$

$$E_l^T = \sum_{\omega \in \Omega} p(\omega) u_{\omega jl} T(\omega)$$

$$e_l = \sum_{\omega \in \Omega} p(\omega) u_{\omega jl}^T h(\omega)$$

Recall that $\theta$ here is another decision variable that helps to approximate the recourse function $Q(x)$. The constraint (4.11) only concerns the first-stage decision variables which are the same as constraint (2.2) in the problem (2.1)-(2.4). The constraint (4.12) is referred as feasibility cuts and they are said to be valid if
there exists some finite value of $s$ such that $x$ is in the domain of the first-stage decision variables if and only if $\{D_kx \geq d_k, k = 1, \ldots, s\}$. The constraint (4.13) is called as *optimality cuts*. The set of optimality cuts is valid if all $x$ and $\theta$ that satisfy this constraint imply $\theta \geq Q(x)$.

### 4.2 Integer L-shaped Method

This section presents how we modify the L-shaped method for two-stage stochastic program with integer second-stage decision variables. When the problem has integer second-stage decision variables, LP relaxation of the subproblems $Q_R(x)$ are used to form optimality cuts (4.13). However, as $Q(x)$ can be discontinuous due to presence of integer second-stage decision variables, optimality cuts (4.13) are not enough on its own to approximate the recourse function. To this end, Laporte and Louveaux [14] modify the L-shaped method for two-stage stochastic mixed-integer programs having binary first-stage variables and has complete recourse. A problem (2.1)-(2.4) is said to have complete recourse property when for all right hand side of the subproblem, there exists feasible second-stage decision variables. The use of a feasibility cut is not needed since the second-stage is always feasible for any feasible first-stage solution. Therefore, they only consider the addition of optimality cuts proposed in [14]. In their proposed method, there are two assumptions. The first assumption is stated as the recourse function to be computable given the first-stage binary variable $x$. This assumption is necessary as the method uses the facial property of binary solutions. In particular, there is a finite set of first-stage solutions with binary first-stage decision variables and the integer optimality cut proposed in [14] is valid for each first-stage feasible solution. Hence, one can observe finite convergence with the Integer L-shaped method used in solving the two-stage stochastic programs having binary first-stage decision variables. The second assumption is on the original problem being bounded. They propose an integer optimality cut and form the fundamental of the Integer L-shaped method.

**Proposition 4.2.1.** [14] Given $x^* \in \{0,1\}^n$, let $S(x^*) := \{i : x^*_i = 1\}$. The
(standard) integer optimality cut at $x^*$ is defined as

$$
\Theta \geq (Q(x^*) - L) \left( \sum_{i \in S(x^*)} x_i - \sum_{i \notin S(x^*)} x_i - |S(x^*)| \right) + Q(x^*) \tag{4.15}
$$

where $Q(x) = \sum_{\omega \in \Omega} p(\omega) Q(x, \omega)$, $L$ is the lower bound on $Q(x)$ and stated as

$$
L = \min_{x, \theta} \{ \theta \mid Ax = b, \ 0 \leq x \leq 1 \ and \ (x, \theta) \ satisfies \ (4.13) \} 
$$

Proof. The reader is referred to [14].

The algorithm for solving a two-stage mixed-integer stochastic program with first-stage binary decision variables with Integer L-shaped method are as follows:

**Algorithm 2** Integer L-Shaped Algorithm

**Step 1:**
Solve LP relaxation of the problem with Benders Decomposition and obtain a master problem

**Step 2:**
Declare $x$ variables as binary in the master problem

**Step 3:**
Solve the master problem and obtain $(x^*, \theta^*)$

**Step 4:**
Compute $Q_R(x^*)$ and add the optimality cut (4.13) to the master problem

**Step 5:**
Compute $Q(x^*)$

**if** $\theta^* < Q(x^*) **then**$

Add the integer optimality cut (4.15) to the master problem,

Go to Step 3

**else**

Go to Step 6

**end if**

**Step 6:**
Stop. $x^*$ is optimal.
4.3 Integer L-shaped Method Applied to Our Problem

In this section, we provide the necessary models and our algorithm for the Integer L-shaped method applied to our problem. As our problem has second-stage integer variables, and first-stage binary decision variables, we can apply Integer L-shaped method described in Section 4.2. Note that our problem has complete recourse property. Therefore, we only consider the cuts (4.15) and (4.13). Remember that we create a master program and subproblems for the scenarios. The subproblem for a scenario $\omega \in \Omega$ is defined as follows:

$$Q(x, \omega) := \min \sum_{k \in K} \sum_{j \in N} r_j \sum_{i \in N^+ \setminus \{j\}} y_{ijk}(\omega)$$ (4.16)

s.t.

$$\sum_{m \in M_{ij}} z_{ijkm}(\omega) = x_{ijk} \quad \forall i \in N^+, j \in N^+ \setminus \{i\}, k \in K$$ (4.17)

$$y_{ijk}(\omega) \leq x_{ijk} \quad \forall i \in N^+, j \in N^+ \setminus \{i\}, k \in K$$ (4.18)

$$w_{0jkm}(\omega) = Lz_{0jkm}(\omega) \quad \forall j \in N, k \in K, m \in M_{ij}$$ (4.19)

$$b_{ij}^m(\omega) z_{ijkm}(\omega) \leq w_{ijkm}(\omega) \leq b_{ij}^{m+1}(\omega) z_{ijkm}(\omega) \quad \forall i \in N, j \in N^+ \setminus \{i\}, m \in M_{ij}, k \in K$$ (4.20)

$$\sum_{i \in N^+ \setminus \{j\}} y_{ijk}(\omega) \leq \sum_{i \in N^+ \setminus \{j\}} y_{jik}(\omega) \quad \forall j \in N, k \in K$$ (4.21)

$$\sum_{i \in N^+ \setminus \{j\}} \left( z_{ijkm}(\omega) \beta_{ij}^m(\omega) + \alpha_{ij}^m(\omega) w_{ijkm}(\omega) + w_{ijkm}(\omega) \right) = \sum_{i \in N^+ \setminus \{j\}} w_{ijkm}(\omega) \quad \forall j \in N, k \in K$$ (4.22)
\[ + w_{ijkm}(\omega) + \sum_{i \in \mathcal{N} \setminus \{j\}, m \in M_{ij}} \left( z_{ijkm}(\omega) \beta^m_{ij}(\omega) \right) \]
\[ + a^m_{ij}(\omega) w_{ijkm}(\omega) \leq H + B q_{jk}(\omega) \quad \forall j \in \mathcal{N}^+, k \in \mathcal{K} \quad (4.23) \]
\[ \sum_{i \in \mathcal{N}^+ \setminus \{j\}} y_{ijk}(\omega) \geq 1 - B(1 - q_{jk}(\omega)) \quad \forall j \in \mathcal{N}^+, k \in \mathcal{K} \quad (4.24) \]
\[ y_{ijk}(\omega), z_{ijkm}(\omega), q_{jk}(\omega) \in \{0, 1\} \quad \forall i \in \mathcal{N}^+, j \in \mathcal{N}^+, \]
\[ m \in M_{ij}, k \in \mathcal{K}, \quad (4.25) \]
\[ w_{ijkm}(\omega) \in \mathbb{R}^+ \quad \forall i \in \mathcal{N}^+, j \in \mathcal{N}^+, \]
\[ m \in M_{ij}, k \in \mathcal{K} \quad (4.26) \]

As the problem (4.16)-(4.26) contains integer decision variables, the function defined in (4.4) is not convex. Hence, the constraint (4.25) is relaxed and the dual of the LP relaxation of the subproblem is used to find extreme points from which optimality cuts are derived. The dual of the second-stage problem when the integrality constraints are relaxed can be formulated as follows:

where \( h_{jk}(\omega), e_{jkm}(\omega), g_{jkm}(\omega), \rho^1_{ijkm}(\omega), \rho^2_{ijkm}(\omega), s_{ijk}(\omega), f_{ik}(\omega), v_{jk}(\omega) \) are dual variables defined for the relaxed optimization problem for a scenario \( \omega \in \Omega, i \in \mathcal{N}^+, j \in \mathcal{N}^+, k \in \mathcal{K}, \) and \( m \in M_{ij} \):

\[
Q_R(x, \omega) := \max \quad \sum_{j \in \mathcal{N}} (H f_{jk}(\omega) + v_{jk}(\omega)(1 - B)) + \sum_{i \in \mathcal{N}^+ \setminus \{i\}, k \in \mathcal{K}} x_{ijk}(s_{ijk}(\omega) + h_{ijk}(\omega)) \quad (4.27)
\]

s.t.

\[
s_{0jk}(\omega) + s_{j0k}(\omega) \leq -h_{0jk}(\omega) - h_{j0k}(\omega) \quad \forall j \in \mathcal{N}, k \in \mathcal{K} \quad (4.28)
\]
\[
h_{ijk}(\omega) - v_{jk}(\omega) \geq r_j \quad \forall i \in \mathcal{N}^+, j \in \mathcal{N} \setminus \{i\}, k \in \mathcal{K} \quad (4.29)
\]
\[
s_{ijk}(\omega) \geq -h_{ijk}(\omega) \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \setminus \{i\}, k \in \mathcal{K} \quad (4.30)
\]
\[
s_{ijk}(\omega) + \rho^1_{ijkm}(\omega) b^m_{ij} \geq (-g_{jk}(\omega) - f_{jk}(\omega)) \beta^m_{ij} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \setminus \{i\}, k \in \mathcal{K},
\]
\[ m \in M_{ij} \quad (4.31) \]
Let $\mathcal{F}_\omega$ be the set of vertices of the feasible set of the problem (4.27) - (4.36). Since the above problem is a linear program, an optimal solution is one of the extreme points (vertices). Hence, the problem can be written as follows:

$$Q_R(x, \omega) = \max_{(f(\omega), h(\omega)) \in \mathcal{F}_\omega} \left\{ \sum_{j \in N} \sum_{k \in K} (Hf_{jk}(\omega) + v_{jk}(\omega)(1 - B)) + \sum_{i \in N^+} \sum_{j \in N^+ \setminus \{i\}} x_{ijk} (s_{ijk}(\omega) + h_{ijk}(\omega)) \right\}$$

Optimality cuts are formed based on the dual solutions of the relaxed subproblems and proposed below:

$$\sum_{\omega \in \Omega} p(\omega) \left( \sum_{j \in N} \sum_{k \in K} (Hf_{jk}(\omega) + v_{jk}(\omega)(1 - B)) + \sum_{i \in N^+} \sum_{j \in N^+ \setminus \{i\}} x_{ijk} (s_{ijk}(\omega) + h_{ijk}(\omega)) \right) \leq \theta$$

(4.37)

The optimality cut (4.37) is valid, and the recourse function can be approximated in its domain of finiteness by the subgradients using optimal dual solutions if the subproblems are linear and $Q(x)$ is convex in $x$. However, since the subproblem (4.16)-(4.26) is a mixed-integer problem, $Q(x)$ can be discontinuous. Therefore, approximation cannot be made. Integer optimality cuts (4.15) are added to overcome this issue.
After forming the optimality cuts, the linear master program for the Integer L-shaped method is given below:

\[
\text{max} \sum_{k \in K} \sum_{j \in N} r_j \sum_{i \in N^+ \setminus \{j\}} x_{ijk} - \theta \tag{4.38}
\]

subject to

\[
1 \leq \sum_{j \in N} \sum_{k \in K} x_{0jk} = \sum_{j \in N} \sum_{k \in K} x_{j0k} \leq |K| \tag{4.39}
\]

\[
\sum_{i \in N^+ \setminus \{j\}} \sum_{k \in K} x_{ijk} = \sum_{i \in N^+ \setminus \{j\}} \sum_{k \in K} x_{jik} \leq 1 \quad \forall j \in N \tag{4.40}
\]

\[
u_i - u_j + 1 \leq (1 - x_{ijk})|N^+| \quad \forall i \in N, j \in N^+ \setminus \{i\}, k \in K \tag{4.41}
\]

\[
1 \leq u_i \leq |N^+| \quad \forall i \in N^+ \tag{4.42}
\]

**Optimality Cuts (4.37)**

**Integer Optimality Cuts (4.15)**

\[
x_{ijk} \in \{0, 1\} \quad \forall i \in N^+, j \in N^+, k \in K \tag{4.45}
\]

\[
\theta \in \mathbb{R} \tag{4.46}
\]

The first part of the objective function (4.38) is the rewards collected from a prior tour and continuous decision variable \(\theta\) is for approximating the recourse function. Constraints (4.39), (4.40), (4.41), and (4.42) are the same as the constraints in (3.1)-(3.15) which are related only to the first-stage decision variables. Constraint (4.43) is in the form of the optimality cut (4.37). Constraint (4.44) is the integer optimality cuts in the form of (4.15). Constraints (4.45) and are the domain restrictions.

We apply the Integer L-shaped algorithm proposed by Angulo et al. [69]. They present two strategies to improve the performance of the method suggested by La- porte and Louveaux [14]. One of the strategies is the modification of the Integer L-shaped method that alternates between linear and mixed-integer subproblems. The main motivation is to avert from time-consuming exact evaluations of the recourse function. This modified version of the Integer L-shaped method is referred to as the Integer L-shaped method with alternating cut strategy and is used in this study.

Here, we define \(V\) as the set of attained first-stage solutions \(x\) for which \(Q(x)\)
is computed and \( V_R \) as the set of visited first-stage solutions for which the value of recourse function of LP relaxation of subproblems \( Q_R(x) \) is calculated. The outline of the algorithm is given below.

**Algorithm 3 Integer L-Shaped Algorithm with Alternating Cut Strategy**

**Step 0**:  
Set \( v = 0, s = 0 \),

**Step 1**:  
Solve LP relaxation of TOP-TST using Benders Decomposition,  
Obtain a master problem,  
Declare \( x \) variables as binary decision variables in the master problem and solve,  
Let \((x^*, \theta^*)\) be the final solution,  

**Step 2**:  
\textbf{if} \( x^* \notin V_R \) \textbf{then}  
\hspace{1cm} Compute \( Q_R(x^*) = \sum_{\omega \in \Omega} p(\omega)Q_R(x^*, \omega) \)  
\hspace{1cm} \( V_R \leftarrow V_R \cup \{x^*\} \),  
\hspace{1cm} \textbf{if} \( \theta^* < Q_R(x^*) \) \textbf{then}  
\hspace{2cm} Add the optimality cut (4.37),  
\hspace{2cm} \( v \leftarrow v + 1 \),  
\hspace{2cm} Go to Step 4.  
\hspace{1cm} \textbf{else}  
\hspace{2cm} Go to Step 3.  
\hspace{1cm} \textbf{end if}  
\textbf{else}  
\hspace{1cm} Go to Step 3.  
\textbf{end if}  

41
Step 3:
Compute \( Q(x^*) = \sum_{\omega \in \Omega} p(\omega) Q(x^*, \omega) \)
\[ V \leftarrow V \cup \{x^*\} \]
if \( \theta^* < Q(x^*) \) then
   Add the integer optimality cut (4.15),
   \[ s \leftarrow s + 1, \]
   Go to Step 4.
else
   Stop. \( x^* \) is the optimal solution.
endif
Step 4:
Solve the master problem,
Return to Step 2 with the new solution \((x^*, \theta^*)\),

We start the algorithm by setting the number of iterations to zero. In step one, LP relaxation of the problem (3.1)-(3.15) is solved using Benders Decomposition, and a master problem is obtained. Then, we claim \( x \) variables as binary variables in the master problem. This model is solved, and the solution is stored in \((x^*, \theta^*)\). The algorithm proceeds into the next step. In step two, we check whether the solution is already evaluated or not. If \( x^* \) is already in \( V_R \), then \( Q_R(x^*) \) is known, and there is no need to compute it again. If not, for each scenario \( \omega \in \Omega \), relaxed second-stage problems are solved, and the optimal value is kept under the variable \( Q_R(x^*, \omega) \) and the solution \( x^* \) is added to the set \( V_R \). If the condition \( (\theta^* < Q_R(x^*)) \) is satisfied, the optimality cut in the form of (4.37) is added to master problem to remove the current solution \((x^*, \theta^*)\). Then, the iteration number for optimality cut is increased by one, and we continue with Step 4 that is solving the master problem again. If not, the algorithm is continued with solving subproblems with binary restrictions, and \( Q(x^*) \) is computed in Step 3. In that step, the convergence on the second-stage cost is checked. If the condition \( (\theta^* < Q(x^*)) \) is satisfied, then the solution is not optimal for our problem. Then, an integer optimality cut of the form (4.15) is added to discard the current solution. In this way, this solution will not be considered again. Otherwise, we
accept the current solution as the optimal solution to our problem. This strategy
that alternates between the relaxed model and integer model is considered as an
initial step to check whether a candidate solution is feasible or not. It takes away
more time-consuming evaluations of $Q(x^*)$. 
Chapter 5

Computational Study

In this chapter, we first explain the experimental design in Section 5.1, then continue with the results of the problem instances. We tackle the deterministic model to evaluate the value of stochastic solution and compare the differences between tours created by stochastic and deterministic models in Sections 5.2.1 and 5.2.2. We present numerical examples of how the Integer L-shaped method performs for our problem in Section 5.2.3.

Mathematical models are coded in JAVA SE 1.8 with ECLIPSE 2019-12 (4.14.0) and solved using IBM ILOG CPLEX version 12.10. Parameters for scenarios are generated in MATLAB R2019b. A PC with Intel(R) Core(TM) i5-9600KF CPU 3.70GHz and 16 GB RAM running Windows 10 is used to perform numerical study.

5.1 Experimental Design

We create test instances using the data set provided by Tsiligirides [70]. The data set provides the coordinates of 32 nodes and deterministic reward values of each node. Based on the euclidean distance matrix \((d_{ij})_{32x32}\), we generate new distance matrix using normal distribution with mean of \(2d_{ij}\) and standard deviation of
$0.15d_{ij}$. The first node of this data is taken as the depot node. The reward of this node is given as 0. Then, for a problem instance with a total of $|N|$ nodes, we randomly select $(|N| - 1)$ nodes out of 31 nodes. The position of the customer nodes and the associated rewards for our experimental design are presented in Figure 5.1. The node selected as depot node is highlighted with red color.

The following part of the problem setting is constructed similar to [66]. We assume that each agent leaves the depot node at the start of the shift and traffic conditions on the roads are divided into three categories: High Congestion, Medium Congestion, and No Congestion. We set a fraction of arcs to be congested arcs that can experience stochastic congestion according to the traffic congestion status. In the test instances, all arcs starting from or ending in the node numbers, which are multiples of 3, e.g. $(0,1),(0,2),..,(1,3),..$, are classified as congested.
arcs. The rest of the arcs are categorized as deterministic arcs that have almost no congestion at any time of the day. The planning horizon is stated as in Table 5.1 with the corresponding congestion status of the roads.

Table 5.1: Congestion Status of the Roads

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Congestion Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(^{st}) two hours</td>
<td>High</td>
</tr>
<tr>
<td>2(^{nd}) two hours</td>
<td>Medium</td>
</tr>
<tr>
<td>3(^{rd}) two hours</td>
<td>No Congestion</td>
</tr>
<tr>
<td>4(^{th}) two hours</td>
<td>Medium</td>
</tr>
<tr>
<td>5(^{th}) two hours</td>
<td>High</td>
</tr>
<tr>
<td>6(^{th}) two hours</td>
<td>Medium</td>
</tr>
</tbody>
</table>

We consider 20 scenarios, and for each scenario, the speed realizations are sampled from a Normal Distribution with a mean that is set according to traffic condition in the time periods and a coefficient of variation. In the arcs experiencing high congestion, the average speed that a traveler is allowed is 25 kmph, whereas with medium congestion on the arc, the agents can travel with 30 kmph on average. The traffic conditions in a non-congested arc let 50 kmph to travel. The coefficient of variation of high congestion is assumed as 0.30. For medium congestion, it is 0.20, whereas, for the time period in which there is almost no congestion, the coefficient of variation is considered as 0.05. The speed limit is 50 kmph in every arc. Therefore, once travel speed is realized, we take the minimum of 50 and the realized travel speed value.

5.2 Experimental Results and Analysis

We provide the experimental results and analysis. For the test instances generated, we use different \( H \) (time limit) values and change the number of agents according to the number of nodes. In all of the experiments, we assume that
agents leave the depot at the start of the shift, and there is no waiting and service time in the customer nodes.

A sample tour is given in Figure 5.2. In the sample instance, we have 7 nodes in the graph $G$, and 1 agent who has 7.5 hours to complete the tour.

Figure 5.2: A sample tour containing 7 nodes, 1 agent having a time limit of 7.5 hours

The arrows in Figure 5.2 represent the paths that the agent has decided in a priori tour. After the realization of the travel times, the arcs (6-2) and (2-0) are quitted in scenarios 2, 5 and 9. In those scenarios, total travel time exceeds the time limit of 7.5 hours if we include node 2.

In the following sections, we analyze the benefit of solving a two-stage stochastic program and compare the performances of the Integer L-shaped method with alternating cut strategy and CPLEX using different test instances.
5.2.1 Value of Stochastic Solution

As the stochastic models are harder to tackle computationally compared to the deterministic models, in applications, it is often preferred to use models that are computationally simpler than stochastic models. One way to create such a model is to replace random variables with their expected values. This formulation is referred to as Mean Value Problem in the literature. Value of Stochastic Solution ($VSS$) is the difference between the expected value of stochastic solution $E[SS]$ and the expected value of stochastic model using the first-stage solution of Mean Value Problem $E[EVS]$. In this section, we present the value of stochastic solution of different problem instances. The first three columns of Tables A.1 - A.4 show the number of the nodes, time limit, and the total number of agents in the problem instance, respectively. The latter columns represent the expected value of stochastic solution, the expected value of stochastic model using the first-stage solution of Mean Value Problem, and the value of stochastic solution, respectively. The last column shows the improvement in percentages. It is observed that we benefit 16.81% improvement in expected profit by using the stochastic model instead of using the mean value solutions as first-stage solutions. For better analysis, the results are visualized in Figures 5.3 - 5.4.
Figure 5.3: Value of Stochastic Solution for 1 agent

(a) VSS for 8 nodes

(b) VSS for 9 nodes

(c) VSS for 10 nodes

(d) VSS for 11 nodes

(e) VSS for 12 nodes

(f) VSS for 13 nodes
(g) \( VSS \) for 14 nodes

As the time limit increases, \( VSS \) converges to zero. This is an expected observation since more hours to complete the tour makes the time limit constraint redundant in the problem. In general, we also observe later convergence to zero in \( VSS \) as the number of customer nodes increases. This is because the time limit constraint is not redundant up to a point due to increase in the total travel time with more inclusion of customer nodes in a priori tour. However, this observation does not hold when we move from 8 nodes to 9 nodes.
Figure 5.4: Value of Stochastic Solution for 2 agents

(a) $V_{SS}$ for 8 nodes

(b) $V_{SS}$ for 9 nodes

(c) $V_{SS}$ for 10 nodes

(d) $V_{SS}$ for 11 nodes

(e) $V_{SS}$ for 12 nodes

(f) $V_{SS}$ for 13 nodes
It is observed that if we increase the number of agents in the problem, using stochastic solutions is not advised for the case with relatively few number of customer nodes. The increase in the alternative solutions and the redundancy of the time limit constraint makes the stochastic model superfluous. However, if we have larger amount of customer nodes with less time to complete the tour, stochastic solutions result in higher expected profit as seen in Figure 5.4(g).

5.2.2 Stochastic and Deterministic Tours

In this section, we consider optimal stochastic and deterministic solutions and compare a prior tours obtained from those models. In general, we observe that
tours created in deterministic and stochastic models consist of the same customer nodes but in different order of visiting. The order of the nodes is crucial due to the time-dependent travel time and agents experiencing higher variation on traffic congestion on some arcs compared to the other arcs as described in Section 5.1. From the order, we observe that through the end of the tour, the arcs classified as deterministic arcs are included in a prior tour of the stochastic model. This observation is justified through Figure 5.5.

Figure 5.5: A priori tour for 10 nodes and 1 agent having 10 hours to complete the tour

![Deterministic Tour](a) Deterministic Tour

![Stochastic Tour](b) Stochastic Tour

We observe that all of the customer nodes are included in a prior tours of stochastic and deterministic models in Figure 5.5. However, the order is significantly different. Remember from Section 5.1, the arcs connecting the nodes which are multiples of 3 are classified as congested arcs and experience higher variation in travel speed according to congestion status of the road. As seen in Figure 5.5(a), those arcs are used towards the end of the tour. However, in a prior tour of the stochastic model, the arcs referred to as deterministic arcs which have less variation in travel speed are included in the tour through the end. In this way, the possibility of canceling an arc that follows congested arcs is minimized.

There are some cases such that a prior tours obtained by stochastic and deterministic models are slightly different in the aspect of inclusion of the customer
nodes. Now, consider Figure 5.6.

Figure 5.6: A prior tour for 9 nodes and 1 agent having 8 hours to complete the tour

As the arcs connecting node 3 and other nodes experience higher variation in travel speed, this node is excluded in the stochastic tour. Moreover, customer 3 has a relatively low reward. Therefore, the addition of customer 3 into the tour does not maximize the expected profit.

5.2.3 Integer L-shaped Algorithm

We provide a comparison on performances of Integer L-shaped method and CPLEX with 20 scenarios in Table 5.2. The first three columns of Table 5.2 show the number of nodes, time limit, and the total number of agents in the problem instance, respectively. The latter columns of the table present the computation times of the CPLEX and the Integer L-shaped method in seconds. GAP values in percentages are reported next to CPLEX column. In order to benefit from time, the computation of the problem with the CPLEX is stopped once the time for solving the problem instance with Integer L-shaped method is exceeded. This is done especially for the high number of customer nodes in the problem instance. In those cases, GAP values in percentages are reported. The problem instances where CPLEX is run with at least the solution time of the Integer
L-shaped method are marked by *.

Table 5.2: Computational Results of CPLEX and Integer L-shaped Method

<table>
<thead>
<tr>
<th>Test Instance</th>
<th>Computation Time (secs)</th>
<th>CPLEX</th>
<th>CPLEX GAP (%)</th>
<th>Int.Lshaped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>H</td>
<td>K</td>
</tr>
<tr>
<td>7 5 1</td>
<td>1,477</td>
<td>0%</td>
<td>404.412</td>
<td></td>
</tr>
<tr>
<td>7 5 2</td>
<td>660</td>
<td>0%</td>
<td>9.613</td>
<td></td>
</tr>
<tr>
<td>7 6 1</td>
<td>5,216</td>
<td>0%</td>
<td>112.498</td>
<td></td>
</tr>
<tr>
<td>7 6 2</td>
<td>9,327</td>
<td>0%</td>
<td>1.326</td>
<td></td>
</tr>
<tr>
<td>7 7 1</td>
<td>6,728</td>
<td>0%</td>
<td>762</td>
<td></td>
</tr>
<tr>
<td>7 7 2</td>
<td>9,642</td>
<td>0%</td>
<td>3.547</td>
<td></td>
</tr>
<tr>
<td>8 6 1</td>
<td>15,964</td>
<td>0%</td>
<td>7,842</td>
<td></td>
</tr>
<tr>
<td>8 6 2</td>
<td>5,810</td>
<td>0%</td>
<td>1.776</td>
<td></td>
</tr>
<tr>
<td>8 7 1</td>
<td>27,520</td>
<td>0%</td>
<td>50,691</td>
<td></td>
</tr>
<tr>
<td>8 7 2</td>
<td>8,470</td>
<td>0%</td>
<td>1.755</td>
<td></td>
</tr>
<tr>
<td>8 8 1</td>
<td>33,120</td>
<td>0%</td>
<td>1,550</td>
<td></td>
</tr>
<tr>
<td>8 8 2</td>
<td>8,940</td>
<td>0%</td>
<td>1.820</td>
<td></td>
</tr>
<tr>
<td>9 6 1</td>
<td>84,628</td>
<td>0%</td>
<td>2,428</td>
<td></td>
</tr>
<tr>
<td>9 6 2</td>
<td>10,038</td>
<td>0%</td>
<td>2.018</td>
<td></td>
</tr>
<tr>
<td>9 7 1</td>
<td>3,045*</td>
<td>17.17%</td>
<td>3,045</td>
<td></td>
</tr>
<tr>
<td>9 7 2</td>
<td>9,174*</td>
<td>48.97%</td>
<td>4.466</td>
<td></td>
</tr>
<tr>
<td>9 8 1</td>
<td>174,092*</td>
<td>5.56%</td>
<td>15,360</td>
<td></td>
</tr>
<tr>
<td>9 8 2</td>
<td>8,484</td>
<td>0%</td>
<td>4.903</td>
<td></td>
</tr>
<tr>
<td>10 8 1</td>
<td>23,021*</td>
<td>480%</td>
<td>23,021</td>
<td></td>
</tr>
<tr>
<td>10 8 2</td>
<td>2.912*</td>
<td>999.99%</td>
<td>2.912</td>
<td></td>
</tr>
<tr>
<td>10 9 1</td>
<td>21,173*</td>
<td>513.33%</td>
<td>21,173</td>
<td></td>
</tr>
<tr>
<td>10 9 2</td>
<td>3.070*</td>
<td>999.99%</td>
<td>3.070</td>
<td></td>
</tr>
<tr>
<td>10 10 1</td>
<td>320,580*</td>
<td>1.15%</td>
<td>12,245</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 5.2 – Continued from previous page

<table>
<thead>
<tr>
<th>(N)</th>
<th>(H)</th>
<th>(K)</th>
<th>CPLEX</th>
<th>CPLEX GAP (%)</th>
<th>Int.Lshaped</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>2</td>
<td>4.754*</td>
<td>998.99%</td>
<td>4.754</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>1</td>
<td>328,045*</td>
<td>560.00%</td>
<td>328,045</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>2</td>
<td>6.718*</td>
<td>600.00%</td>
<td>6.718</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>1</td>
<td>172,800*</td>
<td>480.00%</td>
<td>172,800</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>2</td>
<td>6.530*</td>
<td>600.00%</td>
<td>6.530</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>1</td>
<td>4729</td>
<td>600.00%</td>
<td>4,729</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>2</td>
<td>8.772*</td>
<td>600.00%</td>
<td>8.772</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>1</td>
<td>361,997*</td>
<td>560.00%</td>
<td>361,997</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4.236*</td>
<td>766.67%</td>
<td>4.236</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>1</td>
<td>147,634*</td>
<td>600.00%</td>
<td>147,634</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>2</td>
<td>5.132*</td>
<td>999.99%</td>
<td>5.132</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>1</td>
<td>65,089*</td>
<td>766.67%</td>
<td>65,089</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>2</td>
<td>5.957*</td>
<td>580.00%</td>
<td>5.957</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>2</td>
<td>13.172*</td>
<td>766.67%</td>
<td>13.172</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>2</td>
<td>8.605*</td>
<td>866.67%</td>
<td>8.605</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>2</td>
<td>7.051*</td>
<td>–</td>
<td>7.051</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>2</td>
<td>11.159*</td>
<td>–</td>
<td>11.159</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>2</td>
<td>20.355*</td>
<td>–</td>
<td>20.355</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>2</td>
<td>No Solution</td>
<td>–</td>
<td>15.255</td>
</tr>
</tbody>
</table>

From Table 5.2, we can observe that the Integer L-shaped method shows a better performance in terms of computational time for the test instances that we created. It can be pointed out that the Integer L-shaped method becomes superior to CPLEX as the complexity of the problem increases. From Table 5.2, we can deduct that, in general, as the number of nodes increases and the time limit decreases, the computational time increases for both CPLEX and Integer L-shaped method keeping the number of agents constant. This can be linked to
having more constraints and less number of alternative solutions. Moreover, as we increase the number of the agents while keeping the other parameters of the problem constant, the computation time decreases. By increasing the number of agents, even though we increase the number of constraints in the problem, the feasible region gets larger and leads to abundance in alternative solutions, which affects the computational time.

Later, we look at the number of optimality cuts and integer optimality cuts generated in the Integer L-shaped Algorithm. Tables 5.3-5.9 show the solution time and the number of cuts formed under different test instances with 20 scenarios where $N$ represents the number of customer nodes, $H$ is for the time limit and $K$ is the number of agents.

Table 5.3: Analysis of Integer L-shaped Algorithm with 20 Scenarios, 2 Agents and Time Limit of 5 Hours

<table>
<thead>
<tr>
<th>$N$</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>84,887</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>96,129</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>12,006</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>13</td>
<td>65,984</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>425</td>
<td>458,624</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>465</td>
<td>2,909,143</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>512</td>
<td>5,407,561</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>469</td>
<td>5,408,817</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>7693</td>
<td>8,909,143</td>
</tr>
</tbody>
</table>

$H = 5; K = 2$
Table 5.4: Analysis of Integer L-shaped Algorithm with 20 Scenarios, 2 Agents and Time Limit of 6 Hours

<table>
<thead>
<tr>
<th>N</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>10099</td>
<td>2,909,143</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>152</td>
<td>158,791</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>267</td>
<td>447,711</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>442</td>
<td>607,387</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>601</td>
<td>914,344</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>7532</td>
<td>18,909,143</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>9402</td>
<td>20,607,392</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>10384</td>
<td>22,781,566</td>
</tr>
</tbody>
</table>

\[ H = 6; K = 2 \]

Table 5.5: Analysis of Integer L-shaped Algorithm with 20 scenarios, 2 Agents and Time Limit of 7 Hours

<table>
<thead>
<tr>
<th>N</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>10</td>
<td>8,192,683</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>255</td>
<td>9,945,632</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>3450</td>
<td>16,452,913</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>5436</td>
<td>28,836,456</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>450</td>
<td>72,480</td>
</tr>
</tbody>
</table>

\[ H = 7; K = 2 \]

From above tables, we can observe that as the number of customer nodes increases, the number of integer optimality cuts and optimality cuts increases along with the solution time. However, there are some exceptions such as the problem instance \((N = 19; H = 7; K = 2)\). This exceptional behaviour is seen if by chance, a starting point is chosen in the surrounding of an optimal solution,
we can obtain less number of optimality and integer optimality cuts.

Table 5.6: Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3 Agents and Time Limit of 7 Hours

<table>
<thead>
<tr>
<th>N</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>3</td>
<td>106,940</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>128</td>
<td>1,012,289</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>41</td>
<td>61,672</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>56</td>
<td>469,600</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>3</td>
<td>494,737</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>300</td>
<td>2,424,196</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>10</td>
<td>181,803</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>13</td>
<td>210,812</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>17</td>
<td>305,011</td>
</tr>
</tbody>
</table>

\( H = 7; K = 3 \)

Table 5.7: Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3 Agents and Time Limit of 8 Hours

<table>
<thead>
<tr>
<th>N</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1</td>
<td>4</td>
<td>99,436</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>23</td>
<td>559,659</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>22</td>
<td>554,427</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>26</td>
<td>769,021</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>28</td>
<td>791,634</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>34</td>
<td>206,528</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>36</td>
<td>876,206</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>41</td>
<td>921,431</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>44</td>
<td>940,793</td>
</tr>
</tbody>
</table>

\( H = 8; K = 3 \)
Table 5.8: Analysis of Integer L-shaped Algorithm with 20 Scenarios, 3 Agents and Time Limit of 9 Hours

<table>
<thead>
<tr>
<th>(N)</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1</td>
<td>0</td>
<td>48,615</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0</td>
<td>58,889</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>2</td>
<td>59,687</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>61,224</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>66,792</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>67,733</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>2</td>
<td>99,624</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>4</td>
<td>100,146</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>17</td>
<td>189,895</td>
</tr>
</tbody>
</table>

\(H = 9; K = 3\)

We observe a decrease in the number of optimality and integer optimality cuts along with a decrease in the solution time when we increase the number of agents. This can be due to the abundance of the alternative solutions and easiness of reaching an optimal solution in which none of the nodes are quitted in the second stage.
Table 5.9: Analysis of Integer L-shaped Algorithm with 20 scenarios and 30 customer nodes

<table>
<thead>
<tr>
<th>$H$</th>
<th>$K$</th>
<th># Optimality Cuts</th>
<th># Integer Optimality Cuts</th>
<th>Time (milisec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>96,617</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>45,909</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>27</td>
<td>0</td>
<td>297,933</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>26,226</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>20,105</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>21,377</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>20,063</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>26,662</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>31,226</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>27,565</td>
</tr>
</tbody>
</table>

$N = 30$

Here, we work on a special case for our problem with 30 nodes. Table 5.9 is provided to show that when the time limit and the number of agents are chosen in a way that none of the nodes are quitted, solving the problem instances with Integer L-shaped method gets so easy so that any solution found after the implementation of the optimality cuts is a good solution. This table shows the benefit of the Integer L-shaped method such that even though we get an out-of-memory error when the two-stage stochastic problem is solved with the CPLEX method, the Integer L-shaped method converges to an optimal solution if none of the nodes are quitted.
Chapter 6

Conclusion and Future Work

In this study, we consider the team orienteering problem with stochastic time-dependent travel time. We assume that the rewards are deterministic and collected by homogeneous agents. We model our problem as a two-stage mixed-integer stochastic program and aim to maximize our expected total reward. As CPLEX fails to find an optimal solution for the large-size problem instances, we propose the Integer L-shaped method as an exact solution algorithm to tackle large-size problem instances. We provide experimental study that gives insights on the performances of the solution approaches and use of stochastic solutions. In our computational study, we observe the superiority of the Integer L-shaped method over CPLEX.

In our experimental study, we show that there is a 16.81% increase in the objective function value on average when we use the stochastic solutions instead of deterministic ones. Hence, the use of two-stage mixed integer programming is advised in this study. Additionally, we compare the deterministic and stochastic prior tours obtained in different problem instances. We observe in general the order of visiting customer nodes is significantly different and in some cases, stochastic prior tour contains less number of customer nodes. In the computational study, we also analyze the computational performance of the Integer
L-shaped algorithm. We observe an increase in the number of optimality and integer optimality cuts along with the computational time as the problem instance becomes more challenging with different parameter settings.

A future research direction of this study is to model our problem as a discrete stochastic dynamic program. In this study, we assume that the realization of the randomness on the travel time is happening abruptly by formulating the problem as a two-stage stochastic program. However, this assumption may not reflect the real life. We assume that all of the uncertainty is resolved in the second stage even though the randomness in the travel times is realized gradually. In literature, there are some studies on formulation of the OP as a discrete stochastic dynamic program. Dolinskaya et al. [54] study stochastic OP with stochastic travel time by formulating as a dynamic program. Moreover, Zhang et al. [53] introduce a stochastic OP on a network of queues and model the problem as a Markov decision process. To our knowledge, our problem is not modeled as a Markov decision process in the literature. Hence, a future research direction of this study can be formulating the problem as a discrete stochastic dynamic program and developing solution approaches by keeping the mentioned studies here in mind.
Bibliography


Appendix A

Results

Table A.1: Value of Stochastic Solution-1

<table>
<thead>
<tr>
<th>Instance</th>
<th>$E[SS]$</th>
<th>$E[EVS]$</th>
<th>VSS</th>
<th>Increase (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 5 1</td>
<td>150</td>
<td>128</td>
<td>22</td>
<td>14.67%</td>
</tr>
<tr>
<td>7 5 2</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>7 6 1</td>
<td>170</td>
<td>102</td>
<td>68</td>
<td>40%</td>
</tr>
<tr>
<td>7 6 2</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>7 7 1</td>
<td>170</td>
<td>136</td>
<td>34</td>
<td>20%</td>
</tr>
<tr>
<td>7 7 2</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>7 8 1</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>7 8 2</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>8 5 1</td>
<td>190</td>
<td>78</td>
<td>112</td>
<td>58.95%</td>
</tr>
<tr>
<td>8 5 2</td>
<td>210</td>
<td>183</td>
<td>27</td>
<td>12.86%</td>
</tr>
<tr>
<td>8 6 1</td>
<td>200</td>
<td>102.5</td>
<td>97.5</td>
<td>48.75%</td>
</tr>
<tr>
<td>8 6 2</td>
<td>210</td>
<td>210</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>8 7 1</td>
<td>210</td>
<td>208.5</td>
<td>1.5</td>
<td>0.71%</td>
</tr>
<tr>
<td>8 7 2</td>
<td>210</td>
<td>207</td>
<td>3</td>
<td>1.43%</td>
</tr>
<tr>
<td>8 8 1</td>
<td>210</td>
<td>210</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>8 8 2</td>
<td>210</td>
<td>210</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>
Table A.2: Value of Stochastic Solution-2

<table>
<thead>
<tr>
<th>Instance</th>
<th>$N$</th>
<th>$H$</th>
<th>$K$</th>
<th>$\mathbb{E}[SS]$</th>
<th>$\mathbb{E}[EVS]$</th>
<th>VSS</th>
<th>Increase (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 5 1</td>
<td>203</td>
<td>76.5</td>
<td>126.5</td>
<td>62.32%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 5 2</td>
<td>210</td>
<td>203.5</td>
<td>6.5</td>
<td>3.10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 6 1</td>
<td>210</td>
<td>126</td>
<td>84</td>
<td>40%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 6 2</td>
<td>230</td>
<td>230</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 7 1</td>
<td>215</td>
<td>108</td>
<td>107</td>
<td>49.77%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 7 2</td>
<td>230</td>
<td>230</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 8 1</td>
<td>230</td>
<td>230</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 8 2</td>
<td>230</td>
<td>230</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 5 1</td>
<td>142</td>
<td>126</td>
<td>16</td>
<td>11.27%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 5 2</td>
<td>210</td>
<td>189</td>
<td>21</td>
<td>10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 6 1</td>
<td>158</td>
<td>138</td>
<td>20</td>
<td>12.66%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 6 2</td>
<td>230</td>
<td>208</td>
<td>22</td>
<td>9.57%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 7 1</td>
<td>190</td>
<td>125</td>
<td>65</td>
<td>34.21%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 7 2</td>
<td>250</td>
<td>250</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 8 1</td>
<td>220</td>
<td>137.5</td>
<td>82.5</td>
<td>37.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 8 2</td>
<td>250</td>
<td>250</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 9 1</td>
<td>250</td>
<td>250</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 9 2</td>
<td>250</td>
<td>250</td>
<td>0</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td>$E[SS]$</td>
<td>$E[EVS]$</td>
<td>VSS</td>
<td>Increase (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>-----</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 5 1</td>
<td>160</td>
<td>91</td>
<td>69</td>
<td>43.13%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 5 2</td>
<td>200</td>
<td>183</td>
<td>17</td>
<td>8.5 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 6 1</td>
<td>180</td>
<td>76.5</td>
<td>103.5</td>
<td>57.5 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 6 2</td>
<td>220</td>
<td>203.5</td>
<td>16.5</td>
<td>7.5 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 7 1</td>
<td>220</td>
<td>123.5</td>
<td>96.5</td>
<td>43.86 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 7 2</td>
<td>220</td>
<td>157.5</td>
<td>62.5</td>
<td>28.41 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 8 1</td>
<td>240</td>
<td>189</td>
<td>51</td>
<td>21.25 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 8 2</td>
<td>280</td>
<td>280</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 9 1</td>
<td>250</td>
<td>151</td>
<td>99</td>
<td>39.6 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 9 2</td>
<td>280</td>
<td>210</td>
<td>70</td>
<td>25 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 10 1</td>
<td>280</td>
<td>192</td>
<td>88</td>
<td>31.43 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 10 2</td>
<td>280</td>
<td>280</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 5 1</td>
<td>185</td>
<td>161</td>
<td>24</td>
<td>12.97 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 5 2</td>
<td>286</td>
<td>161</td>
<td>125</td>
<td>43.71 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 6 1</td>
<td>198</td>
<td>182</td>
<td>16</td>
<td>8.08 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 6 2</td>
<td>340</td>
<td>340</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 7 1</td>
<td>210</td>
<td>190</td>
<td>20</td>
<td>9.52 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 7 2</td>
<td>340</td>
<td>340</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 8 1</td>
<td>244</td>
<td>210</td>
<td>34</td>
<td>13.93 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 8 2</td>
<td>340</td>
<td>340</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 9 1</td>
<td>252</td>
<td>219</td>
<td>33</td>
<td>13.10 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 9 2</td>
<td>340</td>
<td>340</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 10 1</td>
<td>296</td>
<td>249</td>
<td>47</td>
<td>15.88 %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 10 2</td>
<td>340</td>
<td>340</td>
<td>0</td>
<td>0 %</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table A.4: Value of Stochastic Solution-4

<table>
<thead>
<tr>
<th>Instance</th>
<th>E[SS]</th>
<th>E[EVS]</th>
<th>VSS</th>
<th>Increase (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 5 1</td>
<td>175</td>
<td>144</td>
<td>31</td>
<td>17.71%</td>
</tr>
<tr>
<td>13 5 2</td>
<td>230</td>
<td>186.5</td>
<td>43.5</td>
<td>18.91%</td>
</tr>
<tr>
<td>13 6 1</td>
<td>190</td>
<td>105</td>
<td>85</td>
<td>44.74%</td>
</tr>
<tr>
<td>13 6 2</td>
<td>260</td>
<td>204</td>
<td>56</td>
<td>21.54%</td>
</tr>
<tr>
<td>13 7 1</td>
<td>230</td>
<td>120</td>
<td>110</td>
<td>47.83%</td>
</tr>
<tr>
<td>13 7 2</td>
<td>290</td>
<td>209</td>
<td>81</td>
<td>27.93%</td>
</tr>
<tr>
<td>13 8 1</td>
<td>259</td>
<td>156</td>
<td>103</td>
<td>39.77%</td>
</tr>
<tr>
<td>13 8 2</td>
<td>290</td>
<td>258.5</td>
<td>31.5</td>
<td>10.86%</td>
</tr>
<tr>
<td>13 9 1</td>
<td>264</td>
<td>154</td>
<td>110</td>
<td>41.67%</td>
</tr>
<tr>
<td>13 9 2</td>
<td>290</td>
<td>290</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>13 10 1</td>
<td>260</td>
<td>221</td>
<td>39</td>
<td>15%</td>
</tr>
<tr>
<td>13 10 2</td>
<td>290</td>
<td>290</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>