

Multi-vehicle Arc Routing Problems to Restore Post-Disaster Network Connectivity

by

Vahid Akbarighadikolaei

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Industrial Engineering and Operations Management



October 2016

**Multi-vehicle Arc Routing Problems to Restore Post-Disaster Network
Connectivity**

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Vahid Akbarighadikolaei

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Assoc. Prof. F. Sibel Salman

Prof. Emre Alper Yıldırım

Assoc. Prof. Burcu Balçık

Assoc. Prof. Deniz Aksen

Prof. Ümit Bilge

Date: _____



To my family

ABSTRACT

After a natural disaster roads can be damaged or blocked by debris, while bridges and viaducts may collapse. This commonly observed hazard causes some road sections to be closed and may even disconnect the road network. In the immediate disaster response phase work teams are dispatched to open a subset of roads to reconnect the network. Closed roads are traversable only after they are unblocked/cleared by one of the teams. The main objective is to provide an efficient solution method to generate a synchronized work schedule for the road clearing teams so that none of the closed roads are traversed unless their unblocking/clearing procedure is finished. We address two arc routing problems that find the route of multiple work-troops dispatched to clear blocked roads and achieve connectivity. In the first problem, we minimize the time to reconnect the network entirely. In the second problem, we maximize the total prize gained by reconnecting disconnected network components within a specified time limit. For each problem, we develop an exact Mixed Integer Programming (MIP) formulation. Furthermore, for the first problem, we propose a matheuristic that is based on an MIP-relaxation and a local search algorithm. We prove that the optimality gap of the relaxation solution is bounded by K times the lower bound obtained from the relaxed model, where K is the number of teams. For the second problem, we develop a matheuristic method. The matheuristic solves single vehicle problems sequentially with updated prizes. To obtain an upperbound, we first relax the timing elements in the exact formulation and then solve its relaxed MIP, which decomposes into single vehicle problems by Lagrangian Relaxation. We show the effectiveness of the proposed methods computationally on both random Euclidean and three distinct Istanbul road network data generated with respect to predicted earthquake scenarios.

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my adviser Dr. F. Sibel Salman for her kind and useful comments, remarks and engagement through the learning process of this thesis. She has inspired me to become an independent researcher and helped me overcome all the difficulties and shortcomings.

My sincere thanks must also go to the members of my thesis advisory committee: Professors Emre Alper Yildirim and Burcu Balçık. They generously gave their time to offer me valuable comments in the entire process of improving my work. I would also like to thank my other thesis committee members: Dr. Deniz Aksen and Dr. Ümit Bilge for their time and support by reading my thesis, listening to my work, and giving great feedback on it.

I would especially like to thank my family. My wife, Nazila has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point. My dear parents, my siblings, Masume, Majid, Zahra, Roya and Nadia deserve special thanks for their continued support and encouragement. Without such a team behind me, I doubt that I would be in this place today.

I am indebted to my many student colleagues for providing a stimulating and fun environment in which to learn and grow. I am sincerely thankful to Maziar Kasaei, one of my closest friends, for encouraging and supporting each other during the process of our undergraduate and graduate studies. I am especially grateful to Meraaj Ajam, Mahdi Mostajabdaveh, Davood Shiri, Mohammad Nasr, Sasan Asadi, Milad Mokhtaridoost and all my other friends at Koç University.

In the end, I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting this thesis by the grants 111M537 and 114M373.

TABLE OF CONTENTS

List of Tables	ix
List of Figures	x
Nomenclature	xi
Chapter 1: Introduction	1
Chapter 2: Literature Review	5
2.1 Arc Routing Problems	5
2.1.1 Synchronization Problems	7
2.1.2 Prize Collection Problems	9
2.2 Road Maintenance and Recovery Problems	12
Chapter 3: Problem Definitions	16
Chapter 4: Mathematical Models	19
4.1 MIP models for K-ARCP	19
4.1.1 E-MIP-I	20
4.1.2 R-MIP-I	25
4.1.3 Argument on the Relaxation	30
4.2 MIP models for <i>KPC-ARCP</i>	35
4.2.1 E-MIP-II	36
4.2.2 R-MIP-II	38

Chapter 5:	Solution Methods and Algorithms	39
5.1	Heuristic Solution Method for K-ARCP	39
5.1.1	The Feasibility Algorithm.	40
5.1.2	The Local Search Algorithm.	44
5.1.3	The Relaxation-Based Heuristic (RBH).	46
5.2	Solution Methods and Algorithms to KPC-ARCP	50
5.2.1	Lagrangian Relaxation Method to Solve R-MIP-II	51
5.2.2	A Matheuristic for KPC-ARCP	55
Chapter 6:	Design of Computational Experiments	62
6.1	Istanbul Networks	62
6.2	Random Data Sets	66
6.2.1	Random Data Sets for K-ARCP	66
6.2.2	Random Data Sets for KPC-ARCP	67
Chapter 7:	Computational Results	69
7.1	Results of Tested Instances on K-ARCP	69
7.1.1	Results for the Simplified Istanbul Network.	69
7.1.2	Detailed Istanbul Network.	72
7.1.3	Southwestern Istanbul Network.	72
7.1.4	Result of Random Data Sets for K-ARCP	73
7.2	Results of Tested Instances on KPC-ARCP	76
7.2.1	Results of the Detailed Istanbul Network Instances	76
7.2.2	Results of the Southwestern Istanbul Network Instances	78
7.2.3	Results of the Simplified Istanbul Network Instances	80
7.2.4	Result of Random Data Sets for KPC-ARCP	82
Chapter 8:	Conclusions and Remarks	84
	Bibliography	88

LIST OF TABLES

4.1	Decision Variables Used in <i>E-MIP-I</i>	20
4.2	Decision Variables Used in <i>R-MIP-I</i>	25
4.3	Decision Variables Used in <i>E-MIP-X</i>	31
4.4	Decision Variables Used in <i>E-MIP-II</i>	37
4.5	Decision Variables Used in <i>R-MIP-II</i>	38
5.1	Order of Edges from B Visited by Vehicles (1 to K)	40
5.2	Traversing and Clearing Times of the Example	57
6.1	Characteristics of the data sets	63
7.1	Results for the Istanbul Roads Networks	70
7.2	Results for the First Set of Random Data	75
7.3	Results for the Second Set of Random Data (50 nodes)	76
7.4	Results for Detailed Istanbul Network Instances on KPC-ARCP . . .	78
7.5	Result for Southwestern Istanbul Road Network Instances on KPC- ARCP	79
7.6	Results for Simplified Istanbul Network Instances	81
7.7	Summary of the Results of KPC-ARCP (Istanbul Networks)	82
7.8	Random Data results	83

LIST OF FIGURES

4.1	Demonstration of the Flow Formulation	23
4.2	A Worst-Case Instance for $R-MIP-I$	30
4.3	Number of Constraints in $E-MIP-I$ and $E-MIP-X$	34
4.4	Number of Decision Variables in $E-MIP-I$ and $E-MIP-X$	35
5.1	Illustration of Procedure Cut	45
5.2	Example to Illustrate the Local Search Procedures	47
5.3	R-MIP-I Solution	49
5.4	Make Routes Feasible for First Time	49
5.5	Cut and Remove Procedure	49
5.6	Make Routes Feasible for Second Time	50
5.7	Procedure Change Route	50
5.8	An Example to Illustrate Algorithms 3 and 4.	58
5.9	Demonstration of Algorithm 3	59
5.10	Solution Output by Algorithm 3	60
5.11	Demonstration of Algorithm 4.	60
5.12	Solution Output by Algorithm 4.	61
6.1	Istanbul network nodes	63
6.2	Detailed Istanbul Network and the Fault Line	64
6.3	Simplified Istanbul Network	65
6.4	Southwest Istanbul Road Network	65

NOMENCLATURE

ArcGIS : A geographic information system (GIS) for working with maps and geographic information

ARP : Arc Routing Problem

CPU : Central Processing Unit

E-MIP-I : Exact MIP formulation to solve K-ARCP

E-MIP-II : Exact MIP formulation to solve KPC-ARCP

FA : Feasibility Algorithm

Gurobi : An optimization solver for various types of optimization problems

IP : Integer Programming

JICA : Japan International Cooperation Agency

K-ARCP : Multi-vehicle Arc Routing for Connectivity Problem

KPC-ARCP : Multi-vehicle Prize Collecting Arc Routing for Connectivity Problem

LB : Lower bound

MIP : Mixed Integer Programming

NP-hard : Non-deterministic Polynomial-time hard

RBH : Relaxation Based Heuristic

RPP : Rural Postman Problem

R-MIP-I : Time-relaxed MIP formulation for K-ARCP

R-MIP-II : Time-relaxed MIP formulation for KPC-ARCP

SSV : Successive Single Vehicle Algorithm

UB : Upper bound

VGI : Volunteered Geographic Information

WEA : Walk Extraction Algorithm

Chapter 1

INTRODUCTION

The experience from past and recent disasters reveals that the roadway elements are quite vulnerable and the damages can seriously affect the transportation of people and commodities. Millions of people suffer from such disasters every year. From destruction of buildings and infrastructure to spread of diseases, natural disasters can devastate countries overnight. After a disaster, it is critical to reach the affected areas to provide relief operations such as search and rescue, medical response, aid delivery and establishing temporary shelters. Disasters cause the road networks to be disconnected due to having roads blocked by means of building debris, fallen lampposts, displaced cars, etc. These conditions impede accessibility to casualties, hospitals and supply locations and cause severe handicaps on providing essential resources to the affected people, as well as preventing evacuation activities. Depending on the damage, time to clear certain road segments may take long, and victims would be waiting to get connected as the teams work on the roads.

Several cases exemplify this situation; the Kocaeli Earthquake in Turkey (1999), the Indonesian Tsunami and Earthquake in 2004, the Haiti Earthquake in 2010, the "Triple Disaster" (2011) in Japan. The 2011 earthquake off the Pacific coast of Tohoku was a magnitude 9.0 (Mw) undersea mega-thrust earthquake occurred on Friday 11 March 2011. As a result of this earthquake Japan's transport network suffered severe disruptions. Almost 4000 road segments, 78 bridges and 29 railway locations were reported to be damaged. Accumulated debris in the downtown of Kamaishi City, Iwate Prefecture, and a damaged arterial (national highway 45) virtually isolated the community from rescue efforts and about 76 percent of the highways in the area were

closed due to damage. More recently, in the Nepal Earthquake (2015), some rural areas such as Gorkha and Lamjung were inaccessible for several days after the disaster due to the blockage of the road segments.

In the immediate disaster response phase, to facilitate emergency transportation, a critical subset of the blocked roads should be cleared or repaired to restore network connectivity as much as possible. When certain connections take too long to restore, alternative modes such as helicopters would be utilized to reach the isolated areas. Hence, they can be reconnected after on in the recovery phase. The required information about damaged roads and the amount of damages can be obtained by satellite images or Volunteered Geographic Information (VGI) such as up-to-date maps by Open Street Map ([Coleman et al., 2009]). Work-troops equipped with required machinery and personnel should be mobilized to clear the roads and regain the connectivity of the network in the shortest time. For instance, in the case of the mentioned earthquake in Japan, prior to the disaster, contracts with construction companies were made so that the teams could be dispatched as quickly as possible, e.g. within one day. We refer to these work-troops as vehicles from now on. In our problems we assume that the vehicles are originally positioned or gathered at specific nodes of the network, referred to as depots. There might be more than one vehicle positioned at a depot. Shortly after information on road conditions is gathered, the vehicles are dispatched to traverse the roads by a walk and work on the blocked edges assigned to them.

The main goal of our first problem is to find the set of closed roads that need to be opened to regain the connectivity of the network, and to construct synchronized routes for the vehicles to clear or unblock these edges in the shortest time. Note that since we consider a post-disaster situation, we assume that the required information about the damaged roads is available. As information is updated over time, the problem can be resolved for the remaining isolated network components. A complexity factor arises noticing that blocked edges are not traversable unless they are completely unblocked and a team has to wait to enter an edge while it is being unblocked. Hence, node

arrival times should be part of the solution. Calculating the arrival time of the vehicles to the nodes complicates the problem, since edges can be traversed multiple times by different teams.

The first problem we study is called *Multi-vehicle Arc Routing for Connectivity Problem* (K-ARCP) (here K refers to the number of vehicles) which is the problem of optimizing the route of multiple teams by minimizing the completion time of the last unblocking task among all the vehicles resulting in a connected network. There might be more than one depot in the network and vehicles might be dispatched from distinct depots. The problem is to find which blocked edges to open to ensure network connectivity and a walk for each vehicle starting at its depot and ending in a dummy sink node. The walk includes all selected open and blocked edges both inside and outside of the components as necessary. The objective is to minimize the total time of the longest walk among the vehicles.

We develop an MIP model to solve the *K-ARCP* but the timing related elements restrain solution of even moderate-sized instances. Considering that the problem should be solved in short time after the incident takes place, we propose a heuristic approach that obtains a feasible solution from the output of a relaxed model and improves it by local search procedures.

In the second problem, we again aim to provide an efficient solution method that finds a subset of the closed roads to open and the order in which they should be opened for each vehicle. Different from *K-ARCP*, in this problem, all the vehicles are located in a single depot. Moreover, in order to make the most number of people accessible in short time, we model the problem with the objective of maximizing the collected prize within an imposed time constraint. The prize gained from connecting a network component could be set to the number of people in the isolated component or a priority weight factoring in locations of facilities such as airports, ports, hospitals and relief supply points within the component. In this problem, calculating the arrival time of the vehicles to the nodes should be considered again as well.

We call the second problem of this thesis, the *Multi-vehicle Prize Collecting Arc*

Routing for Connectivity Problem (KPC-ARCP). We develop an MIP model to solve the addressed problem. However, the timing related conditions prevent us from finding an optimal solution in realistic-sized instances. We propose a matheuristic to obtain a near-optimal feasible solution and a relaxation method to derive a tight upper bound to *KPC-ARCP* which decomposes the problem into single vehicle problems. The heuristic is based on solving K single vehicle MIP models and is easily implementable. Furthermore, the optimality gaps of the heuristic have been found to be zero or very close to zero in the computational study.

In this thesis we contribute to the literature by defining and studying two different multi-vehicle arc routing problems with the connectivity objectives for the first time. The single vehicle cases of these problems were introduced and studied in [Asaly, 2013] and [Kasaei, 2015]. However, in the single vehicle problems we do not need to calculate the arrival times to the nodes and the resulting complexity factors from timing parameters are avoided. The problems are at least as hard as the *Min-Max K-vehicle Windy Rural Postman Problem* but it also involves the selection of the required arcs and its cost structure is more complex. Our solution methods resolves the difficulties arising from the synchronization of the vehicles. We provide efficient heuristic methods that obtain optimal or near-optimal solutions on randomly generated instances and those based on city networks. The proposed algorithms can be utilized for decision support in disaster response.

Chapter 2

LITERATURE REVIEW

The motivation of this chapter is to discuss the related literature to the problems discussed in this work. First, we give an overview of the Arc Routing Problems (ARP). Since synchronization of the vehicles is one of the main issues in the introduced problem in this thesis, a brief overview of the studies on usage of synchronization in network is given next. Next, since *KPC-ARCP* focuses on prize collection, some Prize Collecting studies with their differences and similarities to *KPC-ARCP* are discussed. Finally, some problems which focus on decision problems in network restoration and recovery in humanitarian operations are given.

2.1 Arc Routing Problems

The problems discussed in this thesis fall into the class of *Arc Routing Problems* (ARP). In ARP, the aim is to determine a least-cost traversal of a specified arc subset of a graph, with or without constraints. These problems have attracted the interest of researchers and have many application areas such as network design, transportation and snow plowing. The main motivation of this section is to give an overview of arc routing problems and to introduce some problems related to Arc Routing and Vehicle Routing. Before discussing these problems, first we give some necessary definitions. A mixed graph has both directed and undirected links between nodes. A windy graph is defined on an undirected graph but the edge costs differ depending on the direction of traversal through an edge and an even graph is a graph with even degrees in all its nodes.

The *Chinese Postman Problem* (CPP) [Eiselt et al., 1995a] is an ARP that seek a least-cost traversal of all of the arcs. Although on the undirected and directed graphs,

CPP can be solved by a polynomial time algorithm, CPP is NP-hard if it is defined on a mixed [Papadimitriou, 1976] or a windy graph [Brucker, 1981]. However, CPP can be solved by a polynomial time algorithm on an even windy graph [Win, 1987]. If CPP is defined on a windy graph, it is called *Windy Postman Problem* (WPP) which was first introduced in [Minieka, 1979]. There exists a $3/2$ - approximation algorithm for WPP [Raghavachari and Veerasamy, 1999]. Moreover, there exists a $4/3$ - approximation algorithm for mixed CPP [Martinez, 2003]. However, for the mixed and windy versions of CPP, there are promising exact approaches such as the branch-and-cut algorithms proposed by [Grötschel and Win, 1992].

If there is a fleet of identical vehicles, say K vehicles, then the problem of finding K tours such that all the edges are covered in the graph with minimum total cost is called K-CPP. If K-CPP is defined on undirected or directed networks, then there exist polynomial-time algorithms for them [Assad et al., 1987]. On even mixed graphs K-CPP can be solved in polynomial time too [Pearn, 1994]. However, if the input graph is mixed or windy, then K-CPP is NP-hard [Pearn, 1994].

Another class of ARPs is the *Rural Postman Problem* (RPP). This class of problems are defined on a graph or on a multigraph $G = (V, A)$, where V is the vertex set, A is the arc set, and a none negative cost matrix $C = (C_{ij})$ is associated with A . Arcs can be directed, undirected or both. Unlike CPP that all arcs are required to be traversed in RPP it is required to traverse only a subset $R \subseteq A$ of arcs. RPP is a well-studied version of ARP that was proposed by Orloff [Orloff, 1974]. RPP is shown to be NP-hard on an undirected or directed graph [Lenstra and Kan, 1976]. Exact algorithms were developed in [Christofides et al., 1986b], [Corberán and Sanchis, 1994], [Letchford, 1997] and [Ghiani and Laporte, 2000]. If the arc costs satisfy the triangle inequality, there exists a $3/2$ - approximation algorithm for RPP [Frederickson, 1979].

Various types of RPP can be classified from different perspectives. Similar to CPP, undirected, directed, mixed and windy versions of RPP has been introduced and studied. Moreover, there can be generalized or clustered models. Additional real world constraints such as turn penalties, deadlines and number of vehicles also add

to the varieties. [Akoudad and Jawab, 2013] provide a survey which presents some variations and applications of RPP (see [Eiselt et al., 1995b] for a detailed review and [Ghiani and Laporte, 2014] and [Corberán et al., 2014] for recent ones).

If there is a fleet of K vehicles, then the problem of covering the required edges with K tours that minimize the total cost is called K -RPP. In particular, *Min-Max K-vehicle Windy Rural Postman Problem* studied by [Benavent et al., 2009] is closer to K -ARCP as it minimizes the maximum tour cost. The main difference of K -ARCP from *Min-Max K-vehicle Windy Rural Postman Problem* and other related studies which look for tours including required edges is that our main concern is the connectivity of the network. In addition, the set of required arcs are not known in advance, and there is no requirement for the walks to be closed. Also, the elapsed time for traversal and serving an edge in RPP is assumed to be the same, but in our case, the first traversal of a blocked edge involves unblocking it and takes the vehicle more cost/time. Moreover, the edge is ready for further use only after being unblocked. In other words, traversing a blocked edge before unblocking it is infeasible. Also, the shortest path calculations are not static as in RPP, and the related versions mentioned above, since after each unblocking operation there can be potentially shorter paths.

In general, other than disaster management ARPs have wide range of usages in real-life problems. From Street Sweeping [Bodin and Kursh, 1978] and [Richard W. Eglese, 1991] to Snow Plowing [Campbell and Langevin, 2000], Garbage Collection [Kim et al., 2006] and Mail Delivery [Bouliane and Laporte, 1992] are all different real-life situations that ARP can be implemented. ARP has been used in other areas such as Meter Reading [Wunderlich et al., 1992] as well.

2.1.1 Synchronization Problems

In KPC -ARCP a blocked edge can be traversed only after it is opened by a vehicle. This condition, together with the existence of multiple vehicles, complicates the problem since a certain type of synchronization (coordination) of the vehicle routes should be achieved. Although various synchronization issues have been addressed and

studied in the literature as discussed below, to the best of our knowledge, this type of synchronization, which involves incurring extra costs only for the first traversal of an edge among multiple vehicles, has been addressed before only in [Akbari and Salman, 2016].

Given a network of streets and a fleet of snow plowing vehicles based at a depot, *Synchronized Arc Routing Snow Plowing Problem* introduced in [Salazar-Aguilar et al., 2012] consists of determining a set of routes such that all streets some of which have multiple lanes are plowed by using fleets of synchronized vehicles. The objective is to minimize the duration of the longest route, namely the makespan. The street segments have one or two directions, and each direction has a number of lanes, typically between one and three. Synchronization is interpreted as plowing all lanes of a particular street simultaneously. Although synchronization has different application in [Salazar-Aguilar et al., 2012], it is similar to *K-ARCP* and *KPC-ARCP* in terms of having timing concerns in the mathematical formulation and the heuristic algorithm. Another important difference between this problem and the problems in this thesis is that in *K-ARCP* and *KPC-ARCP* we do not know about the required edges to be opened to enable connectivity, while in [Salazar-Aguilar et al., 2012] all streets should be plowed.

[Salazar-Aguilar et al., 2013] is another related study in terms of synchronization. It introduces the synchronized arc and node routing problem which is on a directed graph with two sets of arcs: 1) street segments that must be painted, 2) street segments that do not need to be painted. In this problem, the authors also use two types of vehicles in the painting process: 1) a replenishment vehicle to supply paint, 2) a set of homogeneous painting vehicles. While the painting is being processed by the painting vehicles, they might run out of paint and the replenishment vehicle should provide them with more paint. The synchronization in this problem refers to choosing the appropriate refill nodes so that the painting vehicles and the replenishment vehicle can meet at the same time without incurring too much waiting. One of the main differences between this problem and *K-ARCP* and *KPC-ARCP* is that in

these problems, the set of edges that need to receive a service are not predetermined. Another important difference is the fact that in *K-ARCP* and *KPC-ARCP* a blocked edge can not be traversed unless it is opened but in this problem, the vehicles can traverse an arc without providing the service.

2.1.2 Prize Collection Problems

KPC-ARCP is built upon the RPP but differs from it in several aspects. The main difference is that in *KPC-ARCP* achieving connectivity is the main concern and the required edges are not known initially and are selected to enable connectivity. Since the objective in *KPC-ARCP* maximizes the sum of the collected prizes, below we discuss variants of RPP that have a prize collecting or profit maximization objective and synchronization of the vehicles.

While several versions of ARP concentrate on prize collection, node routing problems that address the same type of objectives were studied earlier. [Balas, 1989] introduced the *Prize Collecting Traveling Salesman Problem* (PCTSP), where a salesman gets a prize for each city that he visits and pays a penalty for every city that he does not manage to visit while minimizing the corresponding travel costs and penalties. The *Orienteering Problem* (OP) is another node routing problem that considers a prize collecting type of objective function. In OP, the objective is to collect as much prize as possible in a given tour length limit. The *Team Orienteering Problem* (TOP) is an extension of OP in which multiple teams and their start and end nodes as well as scores (prizes) of the nodes are specified. The teams should leave the start node and traverse several nodes and end their walks in the end node in a given amount of time. The objective is to determine the path for each of the teams from the start point to the end point in order to maximize the total score. For references on TOP see [Gavalas et al., 2015].

[Archetti et al., 2013] introduced the arc routing version of TOP which is called *Team Orienteering Arc Routing Problem* (TOARP). Different from TOP in which the prizes can be collected by traversing nodes, in TOARP the prizes should be collected

from the arcs of the input graph. [Archetti et al., 2013] proposed a metaheuristic that finds an optimal solution in almost all the tested instances having up to 27 nodes and 296 arcs.

Several ARPs having a single vehicle and prize collecting or profit maximization objectives have been studied within the last decade. [Aráoz et al., 2006] introduced the *Privatized Rural Postman Problem* (PRPP), which is also known as the *Prize Collecting Rural Postman Problem* (PCRPP). In PCRPP, a tour starting and ending at the depot, and maximizing the difference between the total prizes from serving required arcs and the total traversing costs is found. The authors developed an integer programming (IP) model with an exponential number of constraints. [Aráoz et al., 2009] proposed an iterative algorithm together with a heuristic procedure to solve this problem efficiently. The proposed algorithm solves a relaxation of PCRPP with fewer number of constraints repetitively while the heuristic generates feasible solutions that provide lower bounds at each iteration.

[Feillet et al., 2005] study the *Profitable Arc Tour Problem* (PATP) in which a maximum tour length is imposed and there is an upper bound on the number of times a prize can be collected from an arc. Instances with up to 65 nodes were solved using a branch-and-price algorithm. The *Time-Dependent Prize Collecting Arc Routing Problem* (TD-PARP) is another ARP that focuses on collecting prizes within a given tour length limit. [Black et al., 2013] introduced the TD-PARP in which time dependency refers to travel times that change with the time of day. Two metaheuristic algorithms, one based on Variable Neighborhood Search and one based on Tabu Search are proposed and tested. The authors solved instances with up to 600 prize arcs using the metaheuristic algorithms.

Our problem differs from TOARP, PCRPP, PATP and TD-PARP in several different ways. For instance, in *KPC-ARCP* connecting components to the supply (depot) node results in gaining prizes which is different from all of these problems. Furthermore, edges are categorized into two groups as blocked and unblocked, and traversing the blocked edges incurs an additional unblocking/clearing time for the first time.

Hence, the routes of the vehicles should be coordinated.

[Asaly, 2013] proposed a model for the single vehicle case of *K-ARCP*. In this thesis an exact model considering the required timing conditions is developed for the first time. This formulation can solve *K-ARCP* for small instances exactly within an hour of solution time. We call this exact formulation *E-MIP-I*. We also introduce a relaxed MIP formulation to derive an initial solution for *K-ARCP*. We call the relaxed MIP formulation *R-MIP-I*. Furthermore, we propose a heuristic algorithm to obtain feasible walks from the solution of the relaxed optimization model. The heuristic solution is improved by a neighborhood search algorithm with novel moves specific to the problem. We prove that the optimal objective value of *K-ARCP* is at most K times the lower bound obtained from the relaxed model.

The *Prize Collecting Arc Routing for Connectivity Problem* (*PC-ARCP*), which is the single vehicle case of *KPC-ARCP*, was first introduced in [Kasaei, 2015]. In that study an MIP formulation which extends those offered by [Asaly, 2013] is given and a heuristic algorithm is proposed. Note that the problem in [Asaly, 2013] has the makespan objective. Since *PC-ARCP* deals with the single vehicle case, naturally timing conflicts do not occur. In this thesis, we develop an exact model, *E-MIP-II*, to *KPC-ARCP*. However, without relaxation of timing parameters, the model can be solved only for small-sized instances of *KPC-ARCP* in reasonable time. Since the timing conditions complicates the formulation significantly, we relax these conditions and develop a relaxed MIP (*R-MIP-II*) to derive an upper bound for *KPC-ARCP*. Yet, even *R-MIP-II* can not be solved in computationally reasonable time, which is in our case one hour. Hence, we develop a Lagrangian Relaxation procedure to derive an upper bound on *R-MIP-II* and a heuristic algorithm to derive lower bounds on *R-MIP-II* by means of feasible solutions to *R-MIP-II*.

We propose a matheuristic algorithm to obtain feasible solutions to *KPC-ARCP* that solves single vehicle models sequentially. The corresponding lower bound together with the upper bound obtained by decomposing *R-MIP-II* into single vehicle problems using the Lagrangian Relaxation procedure yields an optimality gap to

KPC-ARCP. In this article we use the idea of decomposing the multi-vehicle problem into single vehicle problems and solving the single vehicle problems under problem-specific conditions to obtain both feasible solutions and upper bounds. This approach can be implemented in other multi-vehicle problems as well.

2.2 Road Maintenance and Recovery Problems

Recently, several studies focused on upgrading a road network or improving accessibility after a disaster. Some of these studies do not address routing and instead, they focus on selecting road segments to be upgraded or repaired. For instance, [Duque and Sörensen, 2011] investigate the case where there is a budget constraint, and there is a number of non-operative roads which need to be repaired after a disaster. They assign weights to the rural towns depending on the importance of the towns. The objective in this problem is to minimize the weighted sum of time to travel from each rural town to its closest regional centre. [Liberatore et al., 2014] proposed a model for the optimization of recovery and distribution operations in a coordinated manner and solved it on the Haiti disaster case study. In this hierarchical model, they plan for recovering the damaged roads while considering the consequent distribution planning to the affected areas. The criteria include helping the highest number of people, cost, delivery time, security and reliability. Testing the model in a case study, they show the importance of cooperation among agents by comparing the solutions to the coordinated and sequential optimization models.

Several performance measures have been proposed and analyzed for post-disaster rehabilitation operations. In [Chen and Tzeng, 1999], they aim to model the repairing of all the damaged roads while considering the convenience of traveling for people by minimizing the travel time of travelers during reconstruction period, secondly to minimize the working time of each work troop and thirdly to minimize the idle time between work-troops. Their fuzzy multi-objective model is established as a simulation model of reconstruction scheduling with many work troops. They proposed a modified genetic algorithm which solves the problem efficiently. [Chang and

Nojima, 2001] develop transportation system performance measures and apply them to evaluate the previous experiences and gain insights from the recent earthquakes. Their post-disaster performance measures emphasize physical condition and network functionality. The first one reflects the length of the network that is open to traffic at any point at any time, and is defined as a ratio to the pre-disaster length open. The second measure is based on minimum network travel distances and takes into account the extent and the location of the damage. It attempts to measure changes in accessibility at all nodes of the network. Areal distance-based accessibility pertains to system performance or accessibility from the subareas point of view. They focus on rail and highway transportation systems which are affected in an earthquake.

Scheduling of roadway repair operations is studied by [Yan and Shih, 2009], where transportation of relief items is optimized as well. The constraint that road segments should be repaired before specific times is imposed. Therefore, certain tasks have to be scheduled before the other repair tasks due to their importance. In addition, time windows exist and there are supply points from which relief distribution originates. A set of demand points is chosen and each point has to receive a minimum required amount of relief items. The objectives are to minimize the total time needed for emergency repair at all repair points and for sending relief supplies to all demand points. The authors construct a multi-objective, multi-commodity network flow model on a time-space network and develop a heuristic method. Different from our study, the authors focus on mainly upgrading a road network and are not concerned with routing of the vehicles.

[Özdamar et al., 2014] generate roadside debris clean up plans with a given number of vehicles in the post-disaster stage. They propose an accessibility measure that considers shortest path lengths between pair of nodes over time. The shortest paths may change while the roads are restored. They develop a rule-based constructive heuristic to maximize accessibility. This study does not consider the routing of the vehicles but only assigns the road restoration tasks to the vehicles. [Aksu and Özdamar, 2014] propose a path-based time-indexed integer programming (IP) model that identifies

the blocked links to be restored over time until a deadline using multiple vehicles. The objective is to maximize the total weighted earliness of all paths' restoration completion times. They do not consider the walks of the vehicles and are not concerned with their lengths. Instead, they impose that at most K links can be restored at a time. Hence, they do not encounter the synchronization issue in our problem. They solve the IP model for the road networks of two districts in Istanbul, Turkey. The largest instance has around 500 paths and 18 vehicles at most 5% optimality gap is obtained within an hour of solution time.

[Sahin et al., 2016] is a recent study in which the authors focus on providing emergency relief supplies to disaster affected regions as soon as possible by removing the accumulated debris. The aim is to provide assistance to critical nodes by travelling along a path that may include blocked edges to be unblocked. The authors considered the single vehicle problem and solved instances with a limited number of critical nodes. An exact mathematical formulation and a heuristic algorithm are developed and used to tackle the problem. [Berktas et al., 2016] address another version of the same problem. They minimize the weighted sum of visiting times of the critical nodes, where the weights indicate the priorities of the critical nodes. An exact model and a heuristic algorithm using the shortest path distances between the critical nodes is proposed. Computational tests are analyzed on the same data sets with the [Sahin et al., 2016] study. *KPC-ARCP* and *K-ARCP* differs from the problems defined in these two studies as they consider only a single vehicle and hence there is no need for synchronization. Moreover, in *K-ARCP* we look into the problem of connecting a disconnected graph entirely while in [Sahin et al., 2016] the input graph might be connected. Furthermore, while in *KPC-ARCP* a set of blocked edges should be opened to gain prizes, in [Sahin et al., 2016] and [Berktas et al., 2016], the vehicle might not open any of the blocked roads. Finally, in these two studies the nodes that should be visited are known, whereas in *K-ARCP* and *KPC-ARCP* we select which nodes should be visited.

[Çelik et al., 2015] define a multi-period *stochastic debris clearance problem* (SDCP)

with limited information on the debris amounts along the roads. The information is updated in each period and the main decision is to determine a sequence of roads to clear in each period. The problem is also concerned with sending relief items from supply point to demand points and the objective is to maximize total relief demand satisfied. They develop a Markov Decision Process and a heuristic policy for larger instances. None of the above discussed problems addresses the routing of the vehicles explicitly nor aims to reconnect a disconnected network.

Many researchers in the transportation domain have worked on measuring and/or improving accessibility (e.g. [Sohn, 2006] and [Faturechi and Miller-Hooks, 2015]). Since we focus on reconnecting the network, we can say that we also provide accessibility, specifically in terms of connectivity between every pair of nodes, by means of generating vehicle routes and scheduling the unblocking tasks. However, our focus is on road clearance operations.

To test the performance of the proposed methods, we used three different networks from the Istanbul road network using ArcGIS and Google Earth and in addition we generated random Euclidean data sets for *K-ARCP* and *KPC-ARCP* separately. We generate multiple instances with different sets of blocked edges for each network.

Chapter 3

PROBLEM DEFINITIONS

We model our road network as an undirected connected graph $G = (V, E)$. There is a positive cost (time) c_{ij} associated with traversing each edge $e = (i, j) \in E$. Each edge (i, j) can be traversed in the direction from i to j and from j to i with the same cost c_{ij} . After the disaster, a set of edges, B , will be blocked and removal of them from the graph G , will make it disconnected. The opening operation may involve clearing of the debris or repairing damaged segments. We represent the associated extra opening cost (time) of a blocked edge $e = (i, j) \in B$ for $i < j$ as b_{ij} . We assume that the edge opening times will be estimated by collecting post-disaster information on road conditions; for instance by satellite images or volunteered geographical information (VGI).

Traversing a blocked edge is not possible unless it is opened. We assume that if a vehicle arrives at a node incident to a blocked edge while another vehicle is opening it, then it has to wait until the edge is unblocked. Hence, if the walk of two vehicles include the same blocked edge, the one that arrives first to the edge will unblock it.

Let us represent the walk of vehicle k by W_k . The total time of W_k consists of: 1) time of traversing the edges, $C(W_k)$; 2) time of road clearance, $B(W_k)$; and 3) waiting time, $A(W_k)$. The total walk time for vehicle k is calculated as: $T_{W_k} = C(W_k) + B(W_k) + A(W_k)$.

Definition 1. (*Opener*) Let T_{ek} be the arrival time of vehicle k to edge $e = (i, j)$ from either node i or j . In a solution to K-ARCP, if both W_k and $W_{k'}$ include an edge $e = (i, j) \in B$, and if $T_{ek} < T_{ek'}$, then vehicle k is the opener of edge e . Furthermore if for $k \neq k'$; $T_{ek} = T_{ek'}$, then the opener of edge e is the vehicle with smallest k .

Definition 2. (*Waiting Time*) Let Ω_{ek} be the waiting time of vehicle k at edge $e =$

$(i, j) \in B$. If k is the opener of e , then $\Omega_{ek} = 0$. Otherwise, if k' is the opener of e , then: $\Omega_{ek} = \max \{0, T_{ek'} + c_{ij} + b_{ij} - T_{ek}\}$.

Definition 3. (*Feasibility*) A solution to K-ARCP represented by W_k and $T_{W_k}, k = 1, \dots, K$, is feasible if and only if: i) $G_R = (V, (E \setminus B) \cup R)$ is connected. ii) The walks $W_k, k = 1, \dots, K$ are synchronized, meaning that there is exactly one opener vehicle of each blocked edge and T_{W_k} are calculated according to Definitions 1 and 2.

Multi-vehicle Arc Routing for Connectivity Problem (K-ARCP):

The edges that are not blocked form a subgraph $G_B = (V, E \setminus B)$ with Q connected components. Let C_q ($q = 1, \dots, Q$) be the q^{th} connected component with node set V_q and edge set E_q .

In *K-ARCP*, we are looking for a subset of blocked edges, R , such that $G_R = (V, (E \setminus B) \cup R)$ is connected. The solution identifies R and constructs an open walk for each vehicle that starts at its depot. The walks collectively cover R . D denotes the set of depots and P_d is the set of vehicles which are initially positioned in depot $d \in D$. The objective is to minimize the maximum cost (time) of the walks.

K-ARCP is NP-hard since the single vehicle problem, ARCP, is known to be NP-hard ([Kasaei, 2015]).

Multi-vehicle Prize Collecting Arc Routing for Connectivity Problem (KPC-ARCP):

Edges that are not blocked initially form the graph G_B having Q connected components. Let $\tilde{Q}$ be the set of all connected components, $\tilde{Q} = \{1, \dots, Q\}$ and $C_q, q \in \tilde{Q}$ be the q^{th} connected component with node set V_q and edge set E_q .

Without loss of generality, we assume that the depot node denoted by D is in the first connected component ($D \in V_1$). Note that in *KPC-ARCP*, different from *K-ARCP*, there is only one depot in the network. K vehicles that are initially positioned in D are dispatched to open a subset of blocked edges. It is possible for a vehicle to stay put in D . The objective of *KPC-ARCP* is to maximize the total prize gained from connecting components within a time limit, T_{max} . The prize of the q^{th} component, $P_q, q \in \tilde{Q}$ is gained when the component C_q is connected to the first component. The

solution to *KPC-ARCP* constructs the routes of the vehicles, starting from the depot node D and ending in a dummy sink node indexed as $n + 1$, where n is the number of nodes. A vehicle that goes directly to the sink node from D is meant not to be dispatched.



Chapter 4

MATHEMATICAL MODELS

In this section, we present the MIP models that we developed to solve *K-ARCP* and *KPC-ARCP* exactly. These formulations are called *E-MIP-I* and *E-MIP-II* for *K-ARCP* and *KPC-ARCP*, respectively. Calculating the arrival time of the vehicles to the nodes complicates these model since edges can be traversed multiple times. Therefore, we consider a relaxed problem for each of them (*R-MIP-I* and *R-MIP-II*) such that the timing of the vehicles is ignored.

In our MIP models, we distinguish the direction of traversal of the edges in the walks. Therefore, we first define an arc set A such that: $\forall(i, j) \in E : (i, j) \in A$ and $(j, i) \in A$ and furthermore, for all $j \in V : (j, n + 1) \in A$.

4.1 MIP models for K-ARCP

The MIP model for *K-ARCP* determines K open walks such that all the separated components in the network become connected after unblocking a subset of the blocked edges. The walks start from the depots and end in a dummy sink node, indexed as $n + 1$. We address the multi-depot and K vehicle case. The K walks altogether traverse a subset of the edges in B , denoted by R , so that the graph G_R defined in Chapter 3 is connected.

In *R-MIP-I* and *E-MIP-I* we will use the following dominance property.

Proposition 1. *There exists an optimal solution to K-ARCP such that an edge will not be traversed in the same direction more than once.*

Proof. [Christofides et al., 1986a], proved a similar dominance relation for the Rural Postman Problem. Now let us assume a feasible solution to *K-ARCP* opens a set of

blocked edges, R_f . Moreover, each vehicle $k \in \{1, \dots, K\}$ opens a set of blocked edges, R_f^k (R_f^k can be empty). For each vehicle k we can construct an RPP problem when the set of edges that vehicle k should open are determined and the set R_f^k becomes the set of required edges in the RPP instance. In these RPP instances, the dominance property holds. The resulting single vehicle problems of K -ARCP differ in two aspects from the corresponding RPP problems. The walks in K -ARCP are open and the first traversal of a blocked edge incurs the additional cost b_{ij} . However, these differences do not cause the violation of the dominance property. A closed walk can be turned into an open one by removing one of the traversals, hence the property still holds. Secondly, the total unblocking cost/time of all selected blocked edges, R_f^k , should be added to the length of the walk of vehicle k , which is a constant when the set R_f^k is known. Therefore, only the traversal costs c_{ij} will be used in the corresponding single vehicle problems. Hence, the cost structure becomes the same as in RPP. As a result, the dominance property holds for K -ARCP. $\square$

4.1.1 E -MIP-I.

E -MIP-I is the exact MIP formulation that gives the optimal solution of K -ARCP. It prevents timing conflicts by calculating the arrival time to blocked edges. Note that if a vehicle unblocks an edge from B , it can be traversed in both directions. We define arc set $\bar{B}$ such that: $\forall (i, j) \in B : (i, j) \in \bar{B}$ and $(j, i) \in \bar{B}$. We define the decision variables in Table 4.1 and present the constraints next.

Table 4.1: Decision Variables Used in E -MIP-I

Notation	Type	Definition
$x_{ijl}^k, \forall (i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Binary (Proposition 1)	If vehicle k crosses (j, l) right after (i, j) in its walk
$z_{ij}^k, \forall (i, j) \in \bar{B}$	Binary	If (i, j) is unblocked by vehicle k
$f_{ij}^k, \forall (i, j) \in A$	Continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j
$v_{i,j}^k, i \in V \cup \{n+1\}, k \in \{1, \dots, K\}$	Continuous	Number of times vehicle k visits node i
$t_{ijl}^k, \forall (i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Continuous	The time that vehicle k arrives to node j from i before going to l
$s_{ij}^k, \forall (i, j) \in \bar{B}$	Continuous	If arc (i, j) is in the walk of vehicle k ; it is the earliest time that vehicle k can enter $\{i, j\}$. Otherwise, it is 0
w	Continuous	The value of the objective function

Variable s_{ij}^k is used to impose the necessary waiting time for vehicle $k \in \{1, 2, \dots, K\}$

before entering an arc $(i, j) \in \bar{B}$. The following relation should hold due to Definitions 1 and 2 in Chapter 3:

$$s_{ij}^k = \begin{cases} T_{(i,j)k} + \Omega_{(i,j)k}, & \text{if arc } (i, j) \in \bar{B} \text{ is crossed by vehicle } k = 1, 2, \dots, K \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

Min w

$$t_{ij(n+1)}^k \leq w, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in A \quad (4.2)$$

$$x_{ijl}^k \leq \sum_{h:(l,h) \in A} x_{jhl}^k, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K, \quad l : (j, l) \in A \quad \forall k \notin P_i \quad (4.3)$$

$$x_{ijl}^k \leq \sum_{h:(h,i) \in A} x_{hij}^k, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K, \quad l : (j, l) \in A \quad \forall k \notin P_i \quad (4.4)$$

$$\sum_{h:(h,i) \in A} x_{hij}^k \leq 1, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K \quad (4.5)$$

$$\sum_{l:(j,l) \in A} x_{ijl}^k \leq 1, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K \quad (4.6)$$

$$\sum_{j:(d,j) \in A} \sum_{l:(j,l) \in A} x_{djl}^k = 1, \quad \forall d \in D, \quad \forall k \in P_d \quad (4.7)$$

$$\sum_{(i,j) \in A} x_{ij(n+1)}^k = 1, \quad k = 1, 2, \dots, K \quad (4.8)$$

$$\sum_{l:(j,l) \in A} x_{ijl}^k \geq z_{ij}^k, \quad \forall (i, j) \in \bar{B}, \quad k = 1, 2, \dots, K \quad (4.9)$$

$$\sum_{l:(j,l) \in A} x_{ijl}^k + \sum_{h:(i,h) \in A} x_{jih}^k \leq 2 \left(\sum_{\kappa=1}^K z_{ij}^{\kappa} + \sum_{\kappa=1}^K z_{ji}^{\kappa} \right), \quad \forall (i, j) \in B, \quad k = 1, 2, \dots, K \quad (4.10)$$

$$\sum_{\kappa=1}^K (z_{ij}^{\kappa} + z_{ji}^{\kappa}) \leq 1, \quad \forall (i, j) \in B \quad (4.11)$$

$$\sum_{j: \in V \cup \{(n+1)\}: (i,j) \in A} (f_{ij}^k - f_{ji}^k) = -v_i^k, \quad k = 1, 2, \dots, K, \quad \forall i \in V \cup \{(n+1)\} \setminus D \quad (4.12)$$

$$\sum_{j \in V \cup \{(n+1)\}} (f_{dj}^k - f_{jd}^k) = \sum_{i \in V \cup \{(n+1)\} \setminus \{d\}} v_i^k, \quad \forall k \in P_d, \quad \forall d \in D \quad (4.13)$$

$$\sum_{j \in V} f_{j(n+1)}^k = 1, \quad k = 1, 2, \dots, K \quad (4.14)$$

$$f_{ij}^k \leq (n-1) \sum_{l: (j,l) \in A} x_{ijl}^k, \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in A, \quad i, j \in V \cup \{(n+1)\} \quad (4.15)$$

$$f_{ij}^k \geq \sum_{l: (j,l) \in A} x_{ijl}^k, \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in A, \quad i, j \in V \cup \{(n+1)\} \quad (4.16)$$

$$\sum_{j: (j,i) \in A} \sum_{l: (i,l) \in A} x_{jil}^k = v_i^k, \quad k = 1, 2, \dots, K, \quad \forall i \in V \cup \{(n+1)\} \quad (4.17)$$

$$\sum_{\kappa=1}^K \sum_{(i,j) \in \delta(S)} z_{ij}^\kappa \geq 1, \quad \forall S \subset \{1, \dots, Q\}, S \neq \emptyset \quad (4.18)$$

$$t_{ijl}^k \leq M x_{ijl}^k, \quad k = 1, 2, \dots, K, \quad i: (i,j) \in A, \forall l: (j,l) \in A \quad (4.19)$$

$$s_{ij}^k \leq \sum_{h: (h,i) \in A} t_{hij}^k + M(1 - z_{ij}^k), \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in \bar{B} \quad (4.20)$$

$$s_{ij}^k \geq \sum_{h: (h,i) \in A} t_{hij}^k, \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in \bar{B} \quad (4.21)$$

$$s_{ij}^k \geq \sum_{l: (j,l) \in A} t_{ijl}^k - M(1 - z_{ij}^k) - 2M(1 - \sum_{h: (h,i) \in A} x_{hij}^k), \quad (4.22)$$

$$\forall k, \kappa \in \{1, 2, \dots, K\}, \quad \forall (i,j) \in \bar{B}, \quad \text{and} \quad \kappa \neq k$$

$$\sum_{l: (j,l) \in A} t_{ijl}^k \geq s_{ij}^k + b_{ij} z_{ij}^k + \sum_{l: (j,l) \in A} x_{ijl}^k c_{ij}, \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in \bar{B} \quad (4.23)$$

$$\sum_{l: (j,l) \in A} t_{ijl}^k \geq \sum_{h: (h,i) \in A} t_{hij}^k + \sum_{l: (j,l) \in A} x_{ijl}^k c_{ij}, \quad k = 1, 2, \dots, K, \quad \forall (i,j) \in A \setminus \bar{B} \quad (4.24)$$

Constraint (4.2) sets the objective. Note that all walks end in the sink node indexed as $n+1$. Constraints from (4.3) to (4.6) are flow balance equations. A vehicle should enter an arc first, to leave it in the next step of its walk. It also can

not leave an arc unless it is traversed in the previous step of the walk. Constraints (4.3) and (4.4) set these properties. Each arc is traversed at most once by a particular vehicle, Constraints (4.5) and (4.6) enforce this property. Each vehicle starts its walk in its depot and ends it in the sink node. This property is ensured by Constraints (4.7) and (4.8). Constraints from (4.9) to (4.11) set the relation between x_{ijl}^k and z_{ij}^k decision variables. Constraint (4.9) forces that a blocked edge gets open only if one of the vehicles crosses it. Constraints (4.10) ensure that traversing a blocked edge is only possible after unblocking it by one of the vehicles. By (4.11), a blocked edge should be opened by at most one vehicle. In order to ensure connectivity of the walks, we define flow variables f_{ij}^k for every vehicle and for each arc $(i, j) \in A$. The net flow out of a depot node is the total number of visits to all nodes except the depot. For other nodes, net flow equals to the number of visits to the corresponding node. Figure 4.1 gives an example of the usage of the flow variables and how they enable the route of a vehicle to be connected. In the given walk in Figure 4.1, total number of visits is 12. A flow of 12 leaves the depot node and at each visit to any node, one unit of flow is left. Finally one unit of flow enters the sink node. Constraints (4.12) and (4.13) ensure

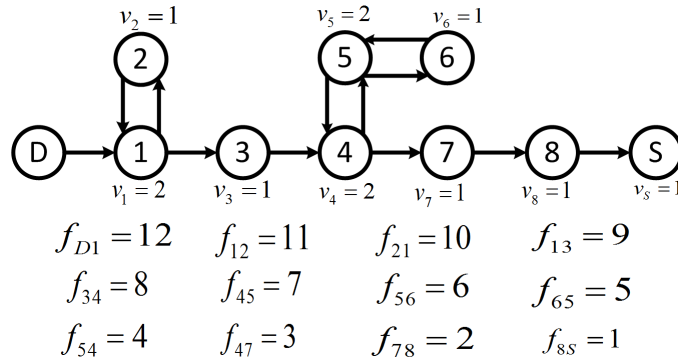


Figure 4.1: Demonstration of the Flow Formulation

that a vehicle leaves one unit of flow each time it visits a node. Constraints (4.14) ensures that walks end in the sink node. Constraint (4.15) does not allow flow on an edge unless it is traversed. Constraint (4.16) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Constraint (4.17) counts

the number of times vehicle k visits node $i \in V$. In order to ensure the connectivity of the network, we consider the cutsets separating each subset of connected components from the rest of the nodes in G . Let $S \subset \{1, \dots, Q\}$, $S \neq \emptyset$ and $V_S = \bigcup_{q \in S} V_q$ then $\delta(S) = (i, j) \in \bar{B} : i \in V_S, j \notin V_S$. Constraint (4.18) enforces that at least one edge in $\delta(S)$ should be unblocked to guarantee the connectivity of G_R . Constraint sets (4.19) to (4.24) are for setting time limits. Constraint (4.19) ensures that, if $x_{ijl}^k = 0$, then $t_{ijl}^k = 0$, where M can be set to $\sum_{(i,j) \in A} c_{ij} + \sum_{(i,j) \in B} b_{ij}$. Constraints (4.20), (4.21) and (4.22) are used to calculate s_{ij}^k values. Variable s_{ij}^k shows the earliest time that edge $(i, j) \in B$ is traversable by vehicle k , if vehicle k is planned to cross it. On the other hand, s_{ij}^k should be forced to zero, if vehicle k does not traverse $(i, j) \in B$. Traversing a blocked edge can only start by a vehicle when either the particular vehicle is the *opener* of the edge, or the edge is already unblocked by another vehicle. Due to constraints (4.20) and (4.21), if vehicle k is the *opener* of edge $(i, j) \in B$, the earliest time that it is traversable by k is the time when it arrives to (i, j) . On the other hand, if $z_{ij}^k = 0$, then (4.20) is redundant, and by (4.21) vehicle k can start traversing (i, j) no sooner than its arrival time to node i . If a vehicle traverses a blocked edge without opening it, it is only possible if the edge is unblocked earlier by another vehicle. In addition, $s_{ij}^k = 0$ should be satisfied, if vehicle k is not planned to cross $(i, j) \in \bar{B}$. Constraint sets (4.23) and (4.24) calculate the arrival times to an edge (i, j) for $(i, j) \in \bar{B}$ and $(i, j) \in A \setminus \bar{B}$, respectively.

The following constraints are the restrictions on the variables.

$$x_{ijl}^k \in \{0, 1\}, \quad k = 1, 2, \dots, K, \quad i : (i, j) \in A, \forall l : (j, l) \in A \quad (4.25)$$

$$z_{ij}^k \in \{0, 1\}, \quad \forall (i, j) \in \bar{B}, \quad k = 1, 2, \dots, K \quad (4.26)$$

$$f_{ij}^k \geq 0, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K \quad (4.27)$$

$$v_i^k \geq 0, \quad \forall i \in V, \quad k = 1, 2, \dots, K \quad (4.28)$$

$$t_{ijl}^k \geq 0, \quad k = 1, 2, \dots, K, \quad i : (i, j) \in A, \forall l : (j, l) \in A \quad (4.29)$$

$$s_{ij}^k \geq 0, \quad \forall (i, j) \in \bar{B}, \quad k = 1, 2, \dots, K \quad (4.30)$$

This formulation has $O(n^3K)$ decision variables.

4.1.2 R -MIP-I.

In this section we present R -MIP-I which solves a relaxation of K -ARCP where timing conditions are ignored. We define the decision variables and present the constraints next.

Table 4.2: Decision Variables Used in R -MIP-I

Notation	Type	Definition
$x_{ij}^k, \forall (i, j) \in A, k \in \{1, \dots, K\}$	Binary	If vehicle k traverses (i, j) from node i to j
$z_{ij}^k, \forall (i, j) \in B$	Binary	If (i, j) is unblocked by vehicle k
$f_{ij}^k, \forall (i, j) \in A$	Continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j
$v_i^k, i \in V \cup \{n+1\}, k \in \{1, \dots, K\}$	Continuous	Number of times vehicle k visits node i
y	Continuous	The value of the objective function

Minimize y

subject to

$$\sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in B} b_{ij} z_{ij}^k \leq y, \quad k = 1, 2, \dots, K \quad (4.31)$$

$$\sum_{j \in V \cup \{(n+1)\} : (d,j) \in A} (x_{dj}^k - x_{jd}^k) = 1, \quad \forall d \in D, \quad \forall k \in P_d \quad (4.32)$$

$$\sum_{j \in V \cup \{(n+1)\} : (i,j) \in A} (x_{ij}^k - x_{ji}^k) = 0, \quad k = 1, 2, \dots, K, \quad \forall i \in V \setminus D \quad (4.33)$$

$$\sum_{j \in V} x_{j(n+1)}^k = 1, \quad k = 1, 2, \dots, K \quad (4.34)$$

$$x_{ij}^k + x_{ji}^k \geq z_{ij}^k, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in B \quad (4.35)$$

$$x_{ij}^k + x_{ji}^k \leq 2 \sum_{\kappa=1}^K z_{ij}^{\kappa}, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in B \quad (4.36)$$

$$\sum_{\kappa=1}^K z_{ij}^{\kappa} \leq 1, \quad \forall (i, j) \in B \quad (4.37)$$

$$\sum_{j: (j,i) \in A} x_{ji}^k = v_i^k, \quad k = 1, 2, \dots, K, \quad \forall i \in V \cup \{(n+1)\} \quad (4.38)$$

$$f_{ij}^k \leq (n-1)x_{ij}^k, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in A, \quad i, j \in V \cup \{(n+1)\} \quad (4.39)$$

$$f_{ij}^k \geq x_{ij}^k, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in A, \quad i, j \in V \cup \{(n+1)\} \quad (4.40)$$

$$(4.12), (4.13), (4.14), (4.18) \quad (4.41)$$

Constraints (4.32) and (4.33) are vehicle flow balance equations. Constraint (4.32) ensures that every vehicle leaves its depot. Constraint (4.34) forces each walk to end in the sink node, which is indexed by $n+1$, where n is the number of nodes. By constraint (4.35), if a blocked edge is opened, it must be traversed at least once from one of the directions. Constraint (4.36) prevents traversing a blocked edge, if it is not unblocked by any of the vehicles. Since the graph is assumed to be undirected, if a blocked edge is selected to be opened, it is enough to open it in one direction. This is enforced by (4.37). A node is visited if and only if an edge entering that node is traversed (4.38). Constraint (4.39) does not allow flow on an edge unless it is traversed. Constraint (4.40) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Note that since the indexes of variable x in *E-MIP* and *R-MIP* are different, all the constraints are not similar to each other. The indexes of variable x is decreased in *R-MIP* to obtain the solution faster. Note that it is not possible to formulate timing constraints with x variables that have three indexes.

The following constraints are the restrictions on the variables.

$$x_{ij}^k \in \{0, 1\}, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K \quad (4.42)$$

$$z_{ij}^k \in \{0, 1\}, \quad \forall (i, j) \in B, \quad k = 1, 2, \dots, K \quad (4.43)$$

$$f_{ij}^k \geq 0, \quad \forall (i, j) \in A, \quad k = 1, 2, \dots, K \quad (4.44)$$

$$v_i^k \geq 0, \quad \forall i \in V, \quad k = 1, 2, \dots, K \quad (4.45)$$

This formulation has $O(n^2K)$ binary variables and an exponential number of constraints due to (4.18). However, when the number of components is small (as expected in our application), the number of constraints is computationally manageable.

Proposition 2. *If $Z_{R-MIP-I}^*$ shows the optimal objective value of R-MIP-I and Z_{K-ARCP}^* denotes the optimal objective value of K-ARCP. Then the following inequality holds.*

$$Z_{R-MIP-I}^* \leq Z_{K-ARCP}^* \leq K \cdot Z_{R-MIP-I}^* \quad (4.46)$$

Proof. The first inequality $Z_{R-MIP-I}^* \leq Z_{K-ARCP}^*$, follows by the fact that any optimal solution that gives Z_{K-ARCP}^* is also feasible for $R - MIP - I$ and gives the same objective value. In order to prove the second inequality, $Z_{K-ARCP}^* \leq K \cdot Z_{R-MIP-I}^*$, first we define a modified version of $K-ARCP$ and then prove two lemmas.

Definition 4. *We define a modified version of K-ARCP, where only vehicle k leaves its depot and the vehicle is allowed to travel from any node to a depot node in zero time at most a specific number of times. In this problem, from each node in V , a dummy arc with zero cost to each depot node is added. If a vehicle goes to some depot in zero time, we call such a step a jump. Remember that P_d shows the set of vehicles that are initially located at depot d . Moreover, in the modified version of the problem for vehicle k , if $k \notin P_d$, vehicle k can jump to node d at most $|P_d|$ times; and if $k \in P_d$, vehicle k can jump back to its depot $|P_d| - 1$ times. Note that the modified version of K-ARCP is a specific problem for each particular vehicle k . We call this version k-Single Vehicle Modified Problem (k-SVMP). Let Z_k^* , be the optimal value of k-SVMP.*

Lemma 1.

$$\frac{\min_{k=1, \dots, K} (Z_k^*)}{K} \leq Z_{R-MIP-I}^* \quad (4.47)$$

Proof. In order to prove this inequality, one should notice that any optimal solution of R-MIP-I can be changed to a feasible solution of k-SVMP. Let us assume $w_1^*, w_2^*, \dots, w_K^*$ are the K optimal walks in the R-MIP-I solution. We can obtain a feasible solution to 1-SVMP such that vehicle 1 first finishes w_1^* , then directly goes to the depot of the

second vehicle with zero cost and traverses w_2^* . After traversing w_2^* , it goes directly to the depot of vehicle 3 and traverses w_3^* . This procedure finishes when this vehicle finishes traversing w_K^* . The overall cost/time for this walk will be equal to $\sum_{i=1}^K T_{w_i^*}$. Note that $T_{w_i^*}$ shows the cost/time of vehicle i in the optimal solution of R-MIP-I. Hence: $Z_1^* \leq \sum_{i=1}^K T_{w_i^*}$. This implies that:

$$\min_{k=1,\dots,K} Z_k^* \leq \sum_{i=1}^K T_{w_i^*} \quad (4.48)$$

Due to the objective of R-MIP-I, we also have:

$$Z_{R-MIP-I}^* = \max_{i=1,\dots,K} T_{w_i^*}. \quad (4.49)$$

Since $\frac{\sum_{i=1}^K T_{w_i^*}}{K} \leq \max_{i=1,\dots,K} T_{w_i^*}$, by equations (4.48) and (4.49):

$$\frac{\min_{k=1,\dots,K} Z_k^*}{K} \leq \frac{\sum_{i=1}^K T_{w_i^*}}{K} \leq \max_{i=1,\dots,K} T_{w_i^*} = Z_{R-MIP-I}^* \quad (4.50)$$

Inequality (4.50) confirms the validity of inequality (4.47). $\square$

Lemma 2.

$$Z_{K-ARCP}^* \leq \min_{k=1,\dots,K} Z_k^* \quad (4.51)$$

Proof. There is a particular vehicle $\hat{k} \in \{1, \dots, K\}$ s.t. $Z_{\hat{k}}^* = \min_{k=1,\dots,K} (Z_k^*)$. $Z_{\hat{k}}^*$ corresponds to a walk, $\hat{w}_{\hat{k}}$, that reconnects all the components. Note that $\hat{w}_{\hat{k}}$ consists of at most $K - 1$ jumps to the depots (let $0 \leq \kappa < K$). Thus, we can separate the steps of $\hat{w}_{\hat{k}}$ as follows:

$$\hat{w}_{\hat{k}} \equiv \hat{w}_1 \xrightarrow{1^{st} jump} \hat{w}_2 \xrightarrow{2^{nd} jump} \dots \xrightarrow{\kappa-2^{nd} jump} \hat{w}_{\kappa-1} \xrightarrow{\kappa-1^{st} jump} \hat{w}_{\kappa} \quad (4.52)$$

We construct a feasible solution, $\bar{w}$, for K-ARCP such that whenever a jump occurs, a new vehicle from that depot starts traversing the edges. More specifically, if we denote the walk of vehicle i , $i = 1, \dots, K$, in $\bar{w}$ by $\bar{w}_i$, then $\bar{w}$ will be as follows:

$$\bar{w} = \begin{cases} \hat{w}_{\hat{i}}, & i \leq \kappa \\ \emptyset, & \kappa < i \leq K \end{cases} \quad (4.53)$$

Note that when the walk of the i^{th} vehicle finishes, $(i+1)^{st}$ vehicle starts its walk. Hence, no timing conflict occurs in the steps of $\bar{w}$. Since in $\hat{w}_{\hat{k}}$ all the components are connected and all of their unblocking tasks are now in $\bar{w}$, $\bar{w}$ is a feasible solution for K-ARCP. In this solution of K-ARCP, we can see that $T_{\bar{w}_1} < T_{\bar{w}_2} < \dots < T_{\bar{w}_\kappa}$, where $T_{\bar{w}_i}$ shows the length of $\bar{w}_i$. Since $T_{\bar{w}_i} = 0$, $\forall i > \kappa$, the corresponding objective value of $\bar{w}$ for K-ARCP is $T_{\bar{w}_\kappa}$. On the other hand, due to the separation procedure of $\hat{w}_{\hat{k}}$ in (4.52), $Z_{\hat{k}}^* = T_{\bar{w}_\kappa}$. Since the optimal value of K-ARCP can be less than the objective value of this feasible solution:

$$Z_{K-ARCP}^* \leq Z_{\hat{k}}^* = \min_{k \in 1, \dots, K} (Z_k^*) \quad (4.54)$$

Thus, we confirm Lemma 2. □

From inequalities (4.47) and (4.51):

$$Z_{K-ARCP}^* \leq \min_{k \in 1, \dots, K} (Z_k^*) \leq K \cdot Z_{R-MIP-I}^* \quad (4.55)$$

As a result, we confirm the second inequality in 4.46 as well and the proof of Proposition 2 is complete. □

Figure 4.2 gives an instance that shows the worst performance of $R-MIP-I$. In this figure, (a) shows the original problem, where three vehicles are originally located in the depot node denoted by D. The edges including $(D, 1)$, $(1, 2)$ and $(2, 3)$ are blocked and the regular cost of traversing each of them is c . The extra opening cost on any of these edges is b . The optimal solution of $R-MIP-I$ is $b + 3c$ and the optimal

solution of $E\text{-MIP-I}$ is $3(b+c)$. These results are shown in part (b) and (c) of Figure 4.2, respectively. It is easy to see that inequality 4.46 holds and if $\frac{c}{b} \rightarrow 0$, then $Z_{K\text{-ARCP}}^* \approx K \times Z_{R\text{-MIP-I}}^*$, where $K = 3$.

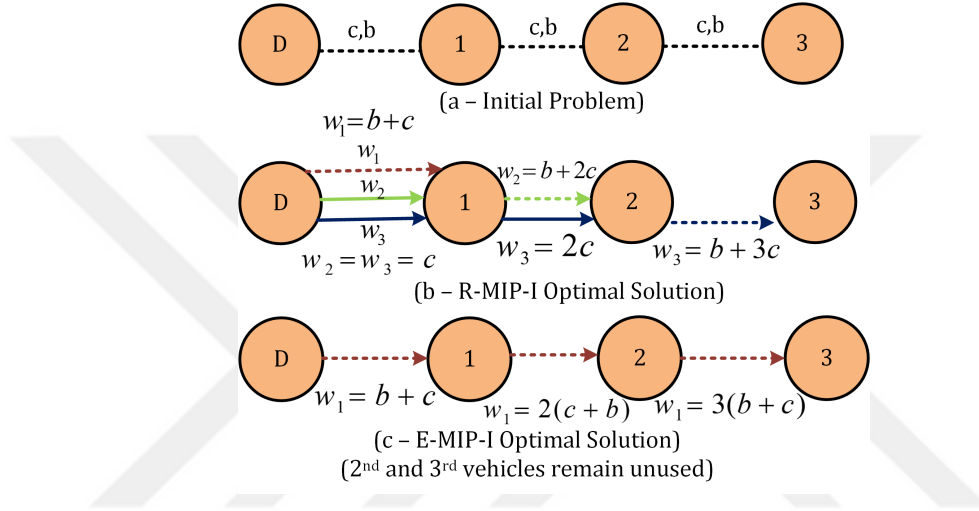


Figure 4.2: A Worst-Case Instance for $R\text{-MIP-I}$

4.1.3 Argument on the Relaxation

The motivation of this section is to clarify the relation between $E\text{-MIP-I}$ and $R\text{-MIP-I}$ and to show that $R\text{-MIP-I}$ is in fact a relaxation to $E\text{-MIP-I}$. While a common understanding of relaxation is defined by removing a set of constraints or changing the objective function, the difference between $R\text{-MIP-I}$ and $E\text{-MIP-I}$ is more complicated. The step variables (x variables) with four indexes in $E\text{-MIP-I}$ are changed to x variables with three indexes in $R\text{-MIP-I}$ and all the constraints that include these variables are changed. However, we can see that a feasible walk from any of these formulations can be transformed to a feasible walk to the other formulation. For instance, if the set $\mathbb{K}$ gives the walk of vehicle k and is defined as $\mathbb{K} = \{(i, j, l) : x_{ijl}^k = 1\}$. Then $\mathbb{K}$ gives the set of (i, j, l) triples such that $(i, j), (j, l) \in A$ and vehicle k traverses from node i to j and then to l . Then this route to $E\text{-MIP-I}$ can be transformed into a route for $R\text{-MIP-I}$ as follows: $\forall (i, j, l) \in \mathbb{K} : x_{ij}^k = 1$ and $x_{jl}^k = 1$ and otherwise x

variables for the k^{th} vehicle are set to be zero.

In this section, we expand $R-MIP-I$ to obtain an equivalent formulation to $E-MIP-I$. For that we define another formulation called $E-MIP-X$. $E-MIP-X$ has the same set of decision variables as $R-MIP-I$ in Table 4.2 plus the required decision variables to calculate the arrival time to nodes and avoid timing conflicts. All the decision variables used in $E-MIP-X$ are given in Table 4.3.

Table 4.3: Decision Variables Used in $E-MIP-X$

Notation	Type	Definition
$x_{ij}^k, \forall(i, j) \in A, k \in \{1, \dots, K\}$	Binary	If vehicle k traverses (i, j) from node i to j
$z_{ij}^k, \forall(i, j) \in B$	Binary	If (i, j) is unblocked by vehicle k
$f_{ij}^k, \forall(i, j) \in A$	Continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j
$v_i^k, i \in i \in V \cup \{n+1\}, k \in \{1, \dots, K\}$	Continuous	Number of times vehicle k visits node i
$u_{ijl}^k, \forall(i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Binary	If vehicle k crosses (j, l) right after (i, j) in its walk
$t_{ijl}^k, \forall(i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Continuous	The time that vehicle k arrives to node j from i before going to l
$s_{ij}^k, \forall(i, j) \in \bar{B}$	Continuous	If arc (i, j) is in the walk of vehicle k ; it is the earliest time that vehicle k can enter $\{i, j\}$. Otherwise, it is 0.
y	Continuous	The value of the objective function

Since $E-MIP-X$ is an expansion of $R-MIP-I$, all the constraints from 4.31 to 4.41 (all the constraints of $R-MIP-I$) are similarly used in $E-MIP-X$. However, the objective function changes as following:

$$\text{Min } y$$

$$t_{ij(n+1)}^k \leq y, \quad k = 1, 2, \dots, K, \quad \forall(i, j) \in A \quad (4.56)$$

Constraint sets numbered from 4.57 to 4.70 are added to the $R-MIP-I$ constraints to provide the timing conditions and they collectively form $E-MIP-II$.

$$v_j^k = \sum_{i:(i,j) \in E} \sum_{l:(j,l) \in E} u_{ijl}^k, \quad \forall j \in V, \quad \forall k \in 1, \dots, k \quad \text{and} \quad k \notin P_j \quad (4.57)$$

$$v_i^k - x_{i(n+1)}^k = \sum_{l:(j,l) \in E} \sum_{j:(i,j) \in E} u_{ijl}^k, \quad \forall i \in V, \quad \forall k \in 1, \dots, k \quad (4.58)$$

$$v_l^k - x_{dl}^k = \sum_{i:(i,j) \in E} \sum_{j:(j,l) \in E} u_{ijl}^k, \quad \forall l \in V, \quad k \in P_d, \quad (d, l) \in E \quad (4.59)$$

$$v_l^k = \sum_{i:(i,j) \in E} \sum_{l:(j,l) \in E} u_{ijl}^k, \quad \forall l \in V, \quad \forall k \in 1, \dots, K, \quad (d, l) \notin E \quad (4.60)$$

$$u_{iji}^k \leq 2 - v_j^k, \quad \forall k \in 1, 2, \dots, K, \quad \forall j \in V, \quad \forall i : (i, j) \in E \quad (4.61)$$

In constraint sets from 4.57 to 4.61, we build up the relation between v_i^k variable sets and u_{ijl}^k variables. In Constraint 4.57, number of times vehicle k appears at node j , is equal to number of times it appears in the middle of two other nodes in the walk of vehicle k . In Constraint 4.58, number of times vehicle k appears at node i , which is the first node in u_{ijl}^k is considered. If the vehicle goes from node i to sink node directly, it can appear in the first set of three nodes in u_{ijl}^k values, less than it was in node i for 1 unit. Same behavior is expected considering the last node in the set of these 3 nodes in u_{ijl}^k . The only difference is that if we go from depot directly to this node, number of times it appears as the last node decrease by 1 unit. Constraint sets 4.59 and 4.60 ensure this relation. Constraint 4.61 is to set the rule for returning nodes in the walk.

$$\sum_{(i,j) \in E} u_{ij(n+1)}^k = 1, \quad \forall k \in 1, \dots, K \quad (4.62)$$

Every walk should ends in the sink node index as $n + 1$. Constraint 4.9 is to assure this property.

$$\sum_{(i,j) \in E} x_{ij}^k - \sum_{i:(i,j) \in E} \sum_{j:(j,l) \in E} u_{ijl}^k = 1, \quad \forall k \in 1, 2, \dots, K, \quad \forall l \in V \quad (4.63)$$

$$2u_{ijl}^k \leq x_{jl}^k + x_{ij}^k, \quad \forall k \in 1, 2, \dots, K, \quad \forall j \in V, \quad \forall (i, j) \in E, \quad \forall (j, l) \in E \quad (4.64)$$

Constraints 4.10 and 4.11 are to construct the relation between x_{ij}^k and u_{ijl}^k variable sets. Constraint 4.10 is the basic summation relation between x_{ij}^k and u_{ijl}^k values in the walk of k^{th} vehicle. Constraint 4.11 ensure the basic relation expected by definition of x_{ij}^k and u_{ijl}^k variables.

$$t_{ijl}^k \leq M u_{ijl}^k, \quad k = 1, 2, \dots, K, \quad i : (i, j) \in A, \forall l : (j, l) \in A \quad (4.65)$$

$$s_{ij}^k \leq \sum_{h:(h,i) \in A} t_{hij}^k + M(1 - z_{ij}^k), \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in \bar{B} \quad (4.66)$$

$$s_{ij}^k \geq \sum_{h:(h,i) \in A} t_{hij}^k, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in \bar{B} \quad (4.67)$$

$$s_{ij}^k \geq \sum_{l:(j,l) \in A} t_{ijl}^\kappa - M(1 - z_{ij}^\kappa) - 2M(1 - \sum_{h:(h,i) \in A} u_{hij}^k), \quad (4.68)$$

$$\forall k, \kappa \in \{1, 2, \dots, K\}, \quad \forall (i, j) \in \bar{B}, \quad \text{and} \quad \kappa \neq k$$

$$\sum_{l:(j,l) \in A} t_{ijl}^k \geq s_{ij}^k + b_{ij} z_{ij}^k + \sum_{l:(j,l) \in A} u_{ijl}^k c_{ij}, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in \bar{B} \quad (4.69)$$

$$\sum_{l:(j,l) \in A} t_{ijl}^k \geq \sum_{h:(h,i) \in A} t_{hij}^k + \sum_{l:(j,l) \in A} u_{ijl}^k c_{ij}, \quad k = 1, 2, \dots, K, \quad \forall (i, j) \in A \setminus \bar{B} \quad (4.70)$$

Constraints from 4.66 to 4.70 are the same timing constraints as 4.19 to 4.24.

We can see that *E-MIP-X* is an expansion of *R-MIP-I* and clearly *R-MIP-I* is a relaxation to it. We can also see that *E-MIP-I* and *E-MIP-X* are equivalent formulations. For that we need to: 1) change x variables in *E-MIP-I* to u variables in *E-MIP-X*, 2) define x variables with three indexes for each arc in A and each vehicle k , 3) add the required constraints from 4.57 to 4.64 to set the relation between u and x variables. It is trivial that if an essential sets of constraints are added to calculate the step variables (u) properly (similar to constraints from 4.3 to 4.8), the additional x variables in *E-MIP-X* are not necessary for calculation of the arrival times to the nodes and hence they can be removed from the formulation. Since *E-MIP-I* and *E-MIP-X* are equivalent formulations and *R-MIP-I* is a relaxation to *E-MIP-X*, we can confirm that *R-MIP-I* is a relaxation to *E-MIP-I* as well.

In order to compare the number of variables and constraints in multiple samples with different number of vehicles for *E-MIP-I* and *E-MIP-X*, we tested 90 distinct sample networks. These networks are randomly generated Euclidean graphs with $10k$ nodes while $k = 1, 2, \dots, 9$. First, we assigned random coordinates in a 100×100 plane to every node. In each case with number of nodes varying from 10 to 90, an edge (i, j) exists if distance between nodes i and j is lower than $\frac{50}{k}$, $k = 1, 2, \dots, 9$. Since

connection of the original input graph is a primary assumption, in case the generated graph is not connected, we add random edges between disconnected components to make it connected. We generated various instances with number of vehicles varying from 1 to 5.

Figure 4.3 shows the number of constraints in all the tested instances for different number of vehicles. There are 2 noticeable facts in this graph: 1) the number of constraints has an exponentially increasing tend as the number of nodes increases, 2) in all the cases with the same number of vehicles, *E-MIP-I* has less constraints than *E-MIP-X*, as expected.

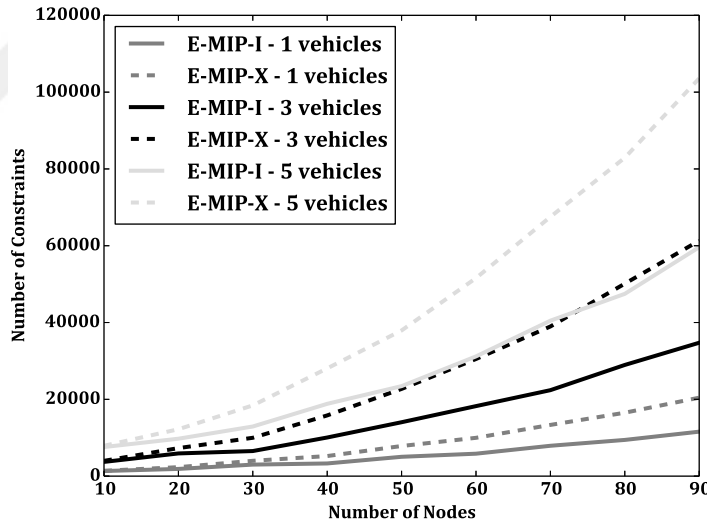


Figure 4.3: Number of Constraints in *E-MIP-I* and *E-MIP-X*

Figure 4.4 shows the number of variables in the tested instances. It can be seen that the number of variables does not differ much in these two formulations for the same number of vehicles. It shows the efficiency of defining the variables for each particular edge instead of walking steps. We can also obtain the following relation between the number of variables in these 2 formulations for a particular number of vehicles.

$$(\text{Number of Variables in } E\text{-MIP-X}) = (\text{Number of Variables in } E\text{-MIP-I}) + \frac{|E|}{2}$$

Where $|E|$ is number of edges in $G = (V, E)$. This relation holds since, $x_{ij}^k \quad \forall (i, j) \in E$ in $E\text{-MIP-}X$ but we do not have these variables in $E\text{-MIP-}I$.

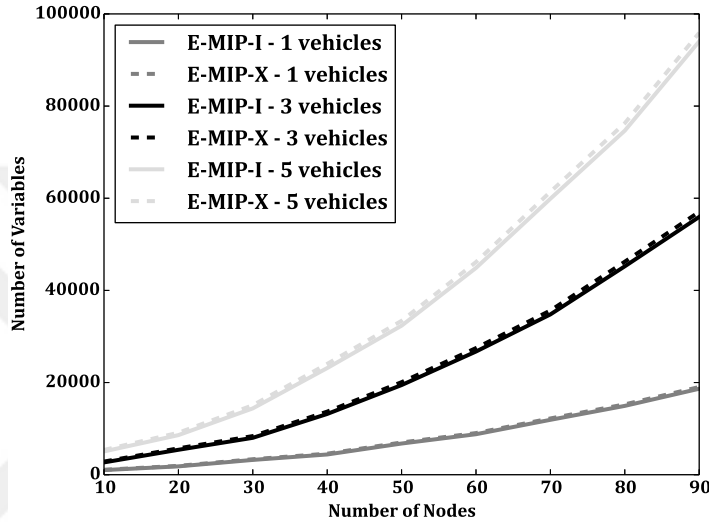


Figure 4.4: Number of Decision Variables in $E\text{-MIP-}I$ and $E\text{-MIP-}X$

Since $E\text{-MIP-}I$ is generally a simpler version of $E\text{-MIP-}X$ and $R\text{-MIP-}I$ is a relaxation to it, we only use $E\text{-MIP-}I$ in our computational study.

4.2 MIP models for KPC-ARCP

In this section we first present an MIP formulation to solve $KPC\text{-ARCP}$ exactly named $E\text{-MIP-II}$. Since the exact formulation for $KPC\text{-ARCP}$ can be solved for only small-sized instances according to our computational tests, we propose a matheuristic method to solve larger instances in short time. The initial solution to this matheuristic is obtained by solving a relaxation to $E\text{-MIP-II}$, which is called $R\text{-MIP-II}$.

We utilize the following dominance property in the formulations.

Proposition 3. *There exists an optimal solution to KPC-ARCP such that an edge will not be traversed in the same direction more than once.*

Proof. Let us assume a feasible solution to $KPC\text{-ARCP}$ opens a set of blocked edges,

R_f . Moreover, each vehicle $k \in \{1, \dots, K\}$ opens a set of blocked edges, R_f^k (R_f^k can be empty). For each vehicle k , we can construct an RPP problem when the set of edges that vehicle k should open are determined and the set R_f^k becomes the set of required edges in the RPP instance. In these RPP instances, the dominance property holds. Note that RPP finds the shortest closed walk containing the required edges, but in *KPC-ARCP* the requirement is that the walk length should be less than T_{max} . Since this solution is feasible to *KPC-ARCP*, the obtained shortest route from the single vehicle RPP instance should be less than T_{max} as well. The resulting single vehicle problems of *KPC-ARCP* differ in two other aspects from the corresponding RPP problems. The walks in *KPC-ARCP* are open and the first traversal of a blocked edge incurs the additional cost b_{ij} . However, these differences do not cause the violation of the dominance property. A closed walk can be turned into an open one by removing one of the traversals, hence the property still holds. Secondly, the total unblocking cost (time) of all selected blocked edges, R_f^k , should be added to the length of the walk of vehicle k , which is a constant when the set R_f^k is known. Moreover, this extra cost is calculated while obtaining the feasible solution and adding it to the tour length does not make it larger than T_{max} . Therefore, only the traversal costs c_{ij} will be used in the corresponding single vehicle problems. Thus, the cost structure becomes the same as in RPP. As a result, the dominance property holds for *KPC-ARCP*. $\square$

4.2.1 *E-MIP-II*

E-MIP-II is the exact MIP formulation that gives an optimal solution to *KPC-ARCP*. It prevents timing conflicts among the vehicle routes by calculating the arrival times to blocked edges and the required waiting times.

We define the decision variables and present the constraints next.

Given the decision variables in Table 4.4, we can write the constraints and the objective function as follows. Note that since several constraints of *E-MIP-II* are similar to *E-MIP-I*, we only give their numbers.

Table 4.4: Decision Variables Used in *E-MIP-II*

Notation	Type	Definition
$x_{ijl}^k, \forall (i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Binary (Proposition 3)	Equals 1 if vehicle k crosses (j, l) right after (i, j) in its walk; 0, otherwise.
$z_{ij}^k, \forall (i, j) \in \bar{B}$	Binary	Equals 1 if (i, j) is unblocked by vehicle k ; 0, otherwise.
$f_{ij}^k, \forall (i, j) \in A$	Continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j .
$v_i^k, i \in V \cup \{n+1\}, k \in \{1, \dots, K\}$	Continuous (Integrality Relaxed)	Number of times vehicle k visits node i .
$t_{ijl}^k, \forall (i, j), (j, l) \in A, k \in \{1, \dots, K\}$	Continuous	The time that vehicle k arrives to node j from i before going to l .
$s_{ij}^k, \forall (i, j) \in \bar{B}$	Continuous	If arc (i, j) is in the walk of vehicle k ; it is the earliest time that vehicle k can enter $\{i, j\}$. Otherwise, it is 0.
$y_q, q \in \{2, \dots, Q\}$	Binary	Equals 1 if component q is connected to the first component; 0, otherwise.

$$\text{Max} \quad \sum_{q=2}^Q P_q y_q \quad (4.71)$$

$$t_{ij(n+1)}^k \leq T_{\max}, \forall (i, j) \in A : i, j \neq n+1, \quad k = 1, 2, \dots, K \quad (4.72)$$

$$y_q \leq \sum_{(i,j) \in \bar{B} : i \notin V_q, j \in V_q} \sum_{k=1}^K z_{ij}^k, \quad q = 2, \dots, Q \quad (4.73)$$

$$\text{All the constraints from (4.3) to (4.24) except (4.18)} \quad (4.74)$$

The objective function maximizes the total prize collected by connecting components to the component with the depot node. Since all walks end in the sink node indexed as $n+1$, there exists an edge $(i, j) \in A, j \neq n+1$ such that $t_{ij(n+1)}^k > 0$. Then $t_{ij(n+1)}^k$ gives the length of the k^{th} vehicle's walk and since it should be less than the maximum allowable length, $T_{\max}$, Constraint (4.2) holds. Constraint (4.73) ensures that y_q is set to 1 only if component q is connected to other components that are connected to the depot node. Since all the vehicles start their walk from the depot node, which is located in the first component, if they enter an isolated component, the component will be connected to the depot node. In order to enter an isolated component, a blocked edge with only one node in the component (i.e. a cutset edge) should be cleared.

4.2.2 *R-MIP-II*

Although *E-MIP-II* gives an optimal solution to *KPC-ARCP*, since calculating the arrival times to the nodes complicates the model, large instances can not be solved in reasonable time. Therefore, we first relax the timing related components in *E-MIP-II*. We call this relaxed formulation the Relaxation Mixed Integer Program, *R-MIP-II*. We can change the four indexed x variables of *E-MIP-II* to three indexed variables in *R-MIP-II* since we do not need to find the waiting times and node arrival times. In the following, first we define the decision variables in Table 4.5 and then present the objective and the constraints of *R-MIP-II*.

Table 4.5: Decision Variables Used in *R-MIP-II*

Notation	Type	Definition
$x_{ij}^k, \forall (i, j) \in A, k \in \{1, \dots, K\}$	Binary	Equals 1 if vehicle k traverses (i, j) from node i to j ; 0, otherwise.
$z_{ij}^k, \forall (i, j) \in B$	Binary	Equals 1 if (i, j) is unblocked by vehicle k ; 0, otherwise.
$f_{ij}^k, \forall (i, j) \in A$	Continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j .
$v_i^k, i \in V \cup \{n+1\}, k \in \{1, \dots, K\}$	Continuous (Integrality Relaxed)	Number of times vehicle k visits node i .
$y_q, q \in \{2, \dots, Q\}$	Binary	Equals 1 if component q is connected to the first component; 0, otherwise.

$$\text{Max} \quad \sum_{q=1}^Q P_q y_q \quad (4.75)$$

subject to

$$\sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in B} b_{ij} z_{ij}^k \leq T_{\max}, \quad k = 1, 2, \dots, K \quad (4.76)$$

$$\text{Constraints (4.73)} \quad (4.77)$$

$$\text{Constraints from (4.32) to (4.40)} \quad (4.78)$$

$$(4.12), (4.13) \text{ and } (4.14) \quad (4.79)$$

In *R-MIP-II*, the length of the walk of the k^{th} vehicle can be calculated as $\sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in B} b_{ij} z_{ij}^k$. We know that this value should be at most $T_{\max}$. Constraint (4.76) sets this relation for each vehicle $k \in \{1, \dots, K\}$.

Chapter 5

SOLUTION METHODS AND ALGORITHMS

The experience from similar routing problems suggests that for large-scale instances, heuristic methods are needed to provide good solutions in an acceptable computing time. After solving the relaxed models, *R-MIP-I* and *R-MIP-II*, we need to extract the steps of the walks for every vehicle in the network. We can obtain the walk of each vehicle using the *Walk Extraction Algorithm*. This algorithm is given in Algorithm 1.

Algorithm 1 Walk Extraction Algorithm

- 1: For the k^{th} vehicle if $\exists i \in V : v_i^k \geq 2$, go to 3. Otherwise, go to 2.
 - 2: The decreasing order of f_{ij}^k values gives us the walk. Output the walk.
 - 3: In the graph $\Gamma = (V, \hat{E})$ such that V is the set of all nodes as before and $\hat{E} = \{(i, j) | x_{ij}^k = 1\}$; find the shortest path from the depot of vehicle k to the sink node.
 - 4: Using a *cycle detection* method, find all the loops in Γ .
 - 5: Go through the nodes in the shortest path obtained in 3. If any of the nodes are in one of the loops derived from 4, add the loop to the shortest path and remove the loop from the set of the loops and then start 5 again. If the size of loops set is 0, the walk is complete. Output the walk.
-

5.1 Heuristic Solution Method for K-ARCP

Since we ignored time-related parameters in *R-MIP-I*, the walks obtained from Algorithm 1 might not be synchronized. In Section 5.1.1, we explain the heuristic algorithm to obtain a feasible solution. In Section 5.1.2, we explain the local search algorithm developed to improve the feasible solution found according to Section 5.1.1.

5.1.1 The Feasibility Algorithm.

We derive a feasible solution by modifying the solution obtained from *R-MIP-I* as follows. First, we find the time at which the vehicles arrive to each blocked edge and determine those edges with timing conflict, that is, we detect whether a vehicle traverses an edge before it is unblocked. Next, we find the blocked edge with the smallest completion time of its unblocking task over all vehicles. We insert waiting times whenever it is necessary by shifting the corresponding completion times forward. Each time we process the next edge on the next smallest completion time over all vehicles. We repeat this procedure until all the time conflicts are prevented. In Figure 5.3, we present an illustrative example where the solution to *R-MIP-I* is infeasible to *K-ARCP* and a feasible solution is derived.

We next describe the *Feasibility Algorithm* in detail. With the results obtained from the *Walk Extraction Algorithm*, we form a solution table such as Table 5.1. In Table 5.1, edges in B that are traversed by different vehicles and the corresponding times are given. Here, $B_{i,j}$ shows the j^{th} blocked edge in the walk of vehicle i and $\tau_{i,j}$ is the time when traversing this edge is completed.

Table 5.1: Order of Edges from B Visited by Vehicles (1 to K)

Walk	Order of Blocked Edges	Finishing Time of the walks
W_1	$B_{1,1} \Rightarrow B_{1,2} \Rightarrow B_{1,3} \Rightarrow \dots \Rightarrow B_{1,m(1)}$	$\tau_{1,1} \Rightarrow \tau_{1,2} \Rightarrow \tau_{1,3} \Rightarrow \dots \Rightarrow \tau_{1,m(1)}$
W_2	$B_{2,1} \Rightarrow B_{2,2} \Rightarrow B_{2,3} \Rightarrow \dots \Rightarrow B_{2,m(2)}$	$\tau_{2,1} \Rightarrow \tau_{2,2} \Rightarrow \tau_{2,3} \Rightarrow \dots \Rightarrow \tau_{2,m(2)}$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
W_K	$B_{K,1} \Rightarrow B_{K,2} \Rightarrow B_{K,3} \Rightarrow \dots \Rightarrow B_{K,m(K)}$	$\tau_{K,1} \Rightarrow \tau_{K,2} \Rightarrow \tau_{K,3} \Rightarrow \dots \Rightarrow \tau_{K,m(K)}$

Note that a blocked edge may be repeated in the walk of a vehicle, or it may appear in the walks of more than one vehicle. During the algorithm, as we process the edges given in Table 5.1, we shift the finishing time of those edges that are affected and remove the processed edges from Table 5.1. If vehicle k is not the *opener* of any edge, then $w_{k,1}$ consists of only the edge from the depot of vehicle k to $n + 1$ and

$t_{k,1} = 0$. Consequently, the k^{th} row of Table 5.1 is empty.

Feasibility Algorithm

Step 1: If Table 5.1 is empty, the routes are synchronized and the algorithm terminates. Otherwise, choose $\psi = \min\{\tau_{i,j}\} - z_{uv}^i b_{uv} - c_{uv}$ over all $i = 1, \dots, K, j = 1, \dots, m(K)$, where edge (u, v) is the j^{th} blocked edge of W_i . Recall that b_{uv} and c_{uv} are the extra cost of unblocking (u, v) and the cost of traversing (u, v) , respectively.

Step 2: Determine the vehicle index, κ , associated with ψ s.t. $\psi = \min\{\tau_{\kappa,j}\} - z_{uv}^\kappa b_{uv} - c_{uv}$. Note that (u, v) is the j^{th} edge from the set of blocked edges, B , traversed by vehicle κ . Go to one of the following cases considering z_{uv}^κ value:

Case 1. **If $z_{uv}^\kappa = 1$:** No timing conflict happens since vehicle κ is opening edge (i, j) in its first visit and it is also the first time that (u, v) is traversed.

(a) Update Table 5.1 by removing all $B_{i,j}$ and $\tau_{i,j}$ values s.t. j is the step of W_i that is traversing (u, v) . Go to Step 1.

Case 2. **If $z_{uv}^\kappa = 0$:** In this case, vehicle κ is the first vehicle that traverses (u, v) without unblocking it. Since we do not have a feasible solution for $K\text{-ARCP}$ yet, we synchronize the walks by setting κ as the *opener* of (u, v) and update the walks of other affected vehicles as follows:

(a) Determine vehicle ρ such that $z_{uv}^\rho = 1$. It means that ρ is the *opener* of (u, v) in the optimal solution of $R\text{-MIP-I}$. (For simplicity of notation let us assume vehicle ρ completes opening edge (u, v) at time τ_ρ .)

(b) (*Making changes in the walk of vehicle κ .*) Since vehicle κ is the first vehicle that traverses edge (u, v) , it should be set as the *opener* of (u, v) instead of ρ . Furthermore, let us assume that vehicle κ traverses edge (u, v) at step β of its walk. Next, we shift forward the time of all walk steps after β for vehicle κ by b_{uv} . Thus, $\forall \tau_{\kappa,j} : j \geq \beta$, set $\tau_{\kappa,j} = \tau_{\kappa,j} + b_{uv}$.

(c) (*Making changes in the walk of vehicle ρ .*) Vehicle ρ should not be the *opener* of (u, v) in the updated solution. Changes should be made for the steps of W_ρ

that are after opening (u, v) . These changes should handle either of the following situations: 1) vehicle ρ waits at edge (u, v) for vehicle κ to finish the unblocking task, or 2) when vehicle ρ arrives to edge (u, v) , its opening is already finished by vehicle κ . Then, the shifts should be as follows: $\forall \tau_{\rho,j}, j \in 1, \dots, m(\rho) : \tau_{\rho} \leq \tau_{\rho,j}$, set $\tau_{\rho,j} = \tau_{\rho,j} - \tau_{\rho} + \max\{\tau_{\kappa,\beta} + c_{uv}, \tau_{\rho} - b_{uv}\}$. Note that $\tau_{\kappa,\beta}$ is the time when unblocking task of vehicle κ on edge (u, v) is completed due to the previous step of the algorithm. If $\tau_{\kappa,\beta} < \tau_{\rho} - b_{uv} - c_{uv}$, then when vehicle ρ arrives to edge (u, v) , its unblocking task by vehicle κ is completed and ρ can finish traversing (u, v) at $\tau_{\rho} + c_{uv}$. On the other hand, if $\tau_{\kappa,\beta} > \tau_{\rho} - b_{uv} - c_{uv}$, vehicle ρ should wait at edge (u, v) for vehicle κ to finish the opening task, and ρ can enter (u, v) at time $\tau_{\kappa,\beta}$. Consequently, its finishing time on edge (u, v) should be at least $\tau_{\kappa,\beta} + c_{uv}$.

- (d) (*Making changes in the walk of other affected vehicles*) If $\exists \eta : \eta \neq \kappa \neq \rho$ and $x_{uv}^{\eta} + x_{vu}^{\eta} \geq 1$. This means that vehicle η also traverses edge (u, v) . Let us assume this traversal occurs at the θ^{th} blocked edge, traversed by η . Furthermore, if $\tau_{\eta,\theta} - c_{uv} \leq \tau_{\kappa,\beta}$, vehicle η traverses edge (u, v) before its unblocking task is finished. Therefore, the steps of W_{η} that are after θ should be shifted forward as follows: $\forall \tau_{\eta,j} : j \geq \theta$, set $\tau_{\eta,j} = \tau_{\eta,j} + \tau_{\kappa,\beta} - \tau_{\eta,\theta} + c_{uv}$.
- (e) Update Table 5.1 by removing all $B_{i,j}$ and $\tau_{i,j}$ values s.t. j is the step of W_i that is traversing (u, v) , then go back to Step 1.

Let us denote the objective function value obtained from the result of the feasibility algorithm by Z_f . Note that this value corresponds to the time of the longest walk among all the vehicles.

Proposition 4. *The Feasibility Algorithm always gives a feasible solution to K-ARCP whose objective value is at most $K \cdot Z_{R-MIP-I}^*$, which is at most $K \cdot Z_{K-ARCP}^*$. The following inequality always holds:*

$$Z_f \leq K \cdot Z_{K-ARCP}^* \quad (5.1)$$

Proof. The optimal solution of R-MIP-I corresponds to a set of walks, $w_{R-MIP-I}$. For a set of K walks to be feasible for $K-ARCP$, they need to: 1) have connected steps (the walks should not jump from one node to another), 2) the collection of the walks should be synchronized meaning that a blocked edge is not traversed unless it is cleared. Since the Feasibility Algorithm does not change the steps of $w_{R-MIP-I}$, they remain connected after the update and the updated walks satisfy the first characteristic of a feasible set of walks for $K-ARCP$ given above. Furthermore, since the procedure in Feasibility Algorithm sets the first vehicle that traverses a blocked edge as its opener, the output of this algorithm holds the second characteristic of a feasible solution as well. Hence, the output of the Feasibility Algorithm is feasible for $K-ARCP$.

Now let us assume the walk of the i^{th} vehicle in $w_{R-MIP-I}$ is shown by w_i^* . In consistence with current notations, $T_{w_i^*}$ shows the time/cost of the walk of vehicle i . In the worst case of the feasibility algorithm, each vehicle i should wait for another vehicle to finish its walk entirely and then start traversing w_i^* . It means that the length of the longest walk among all the vehicles in the updated solution after implementing feasibility algorithm on $w_{R-MIP-I}$ is at most the summation of all vehicles' walks in $w_{R-MIP-I}$, i.e. $Z_f \leq \sum_{i=1, \dots, K} T_{w_i^*}$. On the other hand, due to the definition of K-ARCP,

$Z_{R-MIP-I}^* = \max_{i=1, \dots, K} T_{w_i^*}$. Since $\frac{\sum_{i=1}^K T_{w_i^*}}{K} \leq \max_{i=1, \dots, K} T_{w_i^*}$, we obtain:

$$Z_f \leq \sum_{i=1}^K T_{w_i^*} \leq K \cdot \max_{i=1, \dots, K} T_{w_i^*} = K \cdot Z_{R-MIP-I}^* \quad (5.2)$$

Since $Z_{R-MIP-I}^* \leq Z_{K-ARCP}^*$, hence;

$$Z_f \leq K \cdot Z_{R-MIP-I}^* \leq K \cdot Z_{K-ARCP}^* \quad (5.3)$$

This confirms the validity of (5.1) and proves Proposition 4. $\square$

5.1.2 The Local Search Algorithm.

We developed a *Local Search* algorithm to improve the solution obtained by the *Feasibility Algorithm*. Various procedures were proposed and used for the design of heuristic methods in the arc routing context (see [Hertz et al., 1999] and [Hertz et al., 2000]). None of these procedures is directly applicable for *K-ARCP*.

In the following we define new procedures that we developed particularly for *K-ARCP*. In the first three moves, *Remove*, *Shorten* and *Cut*, the input is a walk by a vehicle and the output is possibly a shorter walk for the same vehicle.

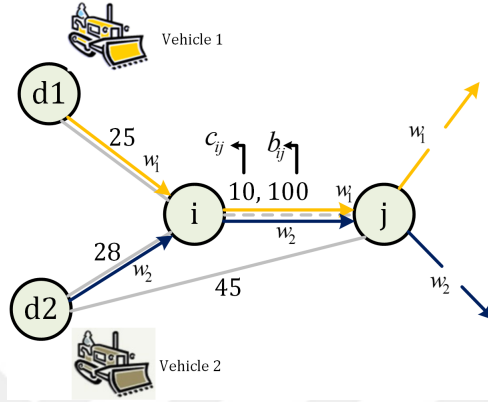
Procedure REMOVE: Find the last edge traversed in the walk of a particular vehicle. If this edge involves unblocking, the output is the current walk. On the other hand, if it does not involve unblocking, remove the last edge from the walk and repeat the procedure again.

Procedure SHORTEN: Consider the walks generated in Table 5.1 and the path between two consecutive blocked edges in each walk. Check if all such paths are shortest paths from their origin node to destination node. If not, modify the route to satisfy this property.

Procedure CUT: Using the walks generated in Table 5.1, check to see if there is any shorter path to avoid traversing blocked edges by vehicles that are not opening them. For instance, as it is shown in Figure 5.1, the optimal solution of the relaxed model chooses $d2 \rightarrow i \rightarrow j$ for the second vehicle since it can arrive to node j at time 38, considering that the first vehicle is opening the blocked edge (i, j) and ignoring the necessary waiting time. However, it should wait until the opener vehicle unblocks (i, j) . Applying the *Cut* procedure, the second vehicle goes from $d2$ to j directly to avoid waiting at node i . Unlike the case in the *Shorten* procedure where we try to choose the shortest path from a blocked edge to the next one, in procedure *Cut*, to prevent waiting, the current vehicle may skip traversing some edges that require unblocking by other vehicles.

In all of the following three procedures, namely, *Change Route*, *Change Opener*

Figure 5.1: Illustration of Procedure Cut



and *Change Route and Opener*, the input is two walks by two different vehicles traversing the same blocked edge and the output is also two walks where the maximum time of the new walks is at most the maximum time of the input walks.

Procedure CHANGE ROUTE: Let $W_{k'}$ be the longest walk among all vehicles. Check if there is any other vehicle, k'' whose walk intersects with $W_{k'}$ on a blocked edge (i, j) . If so, pick one of those vehicles such that changing the routes of k' and k'' gives the best improvement in the objective function and then inter-change the walks of these two vehicles. Considering that an edge might be traversed by two vehicles k' and k'' , either in the same direction or in the opposite direction, one of the cases shown in Figure 5.2 happens. In Case (A1) of Figure 5.2, vehicles 1 and 2 traverse (i, j) in the same direction, while edge $(i, j) \in B$ and vehicle 1 is its *opener* in the current solution. (A2) and (B2) show an implementation of *Change Route* procedure. In (A2) the walks of vehicle 1 and 2 are inter-changed after node j . A similar procedure can be implemented while two different vehicles traverse a common blocked edge in opposite directions. In Figure 5.2, Case (B2) shows an instance where vehicles 1 and 2 traverse $(i, j) \in B$ in opposite directions. It is also assumed that vehicle 1 is the *opener* of (i, j) in the initial solution (B1). In (B2) walks of vehicles 1 and 2 are inter-changed at node i . Note that in (A2) and (B2), although the walks of vehicles 1 and 2 are changed, the *opener* of (i, j) remains the same.

Procedure CHANGE OPENER: As in the *Change Route* procedure, in *Change Opener* we look into $W_{k'}$ such that it is the longest walk among all vehicles. We check if there is any other vehicle k'' that traverses a common blocked edge with k' either in the same direction or opposite directions. In the *Feasibility Algorithm*, we set the vehicle traversing an edge for the first time to unblock it. It is possible that if this vehicle waits for the second vehicle to unblock the edge, we can decrease the maximum time among the two vehicles. If the vehicle that includes the longest walk in the routes is one of these two vehicles, we can obtain a better solution. An instance of *Change Opener* procedure is given in Cases (A3) and (B3) of Figure 5.2. In (A1) and (B1), vehicle 1 is the *opener* of (i, j) in $i \rightarrow j$ and $j \rightarrow i$ directions, respectively. In (A3) and (B3), the *opener* is changed from vehicle 1 to vehicle 2.

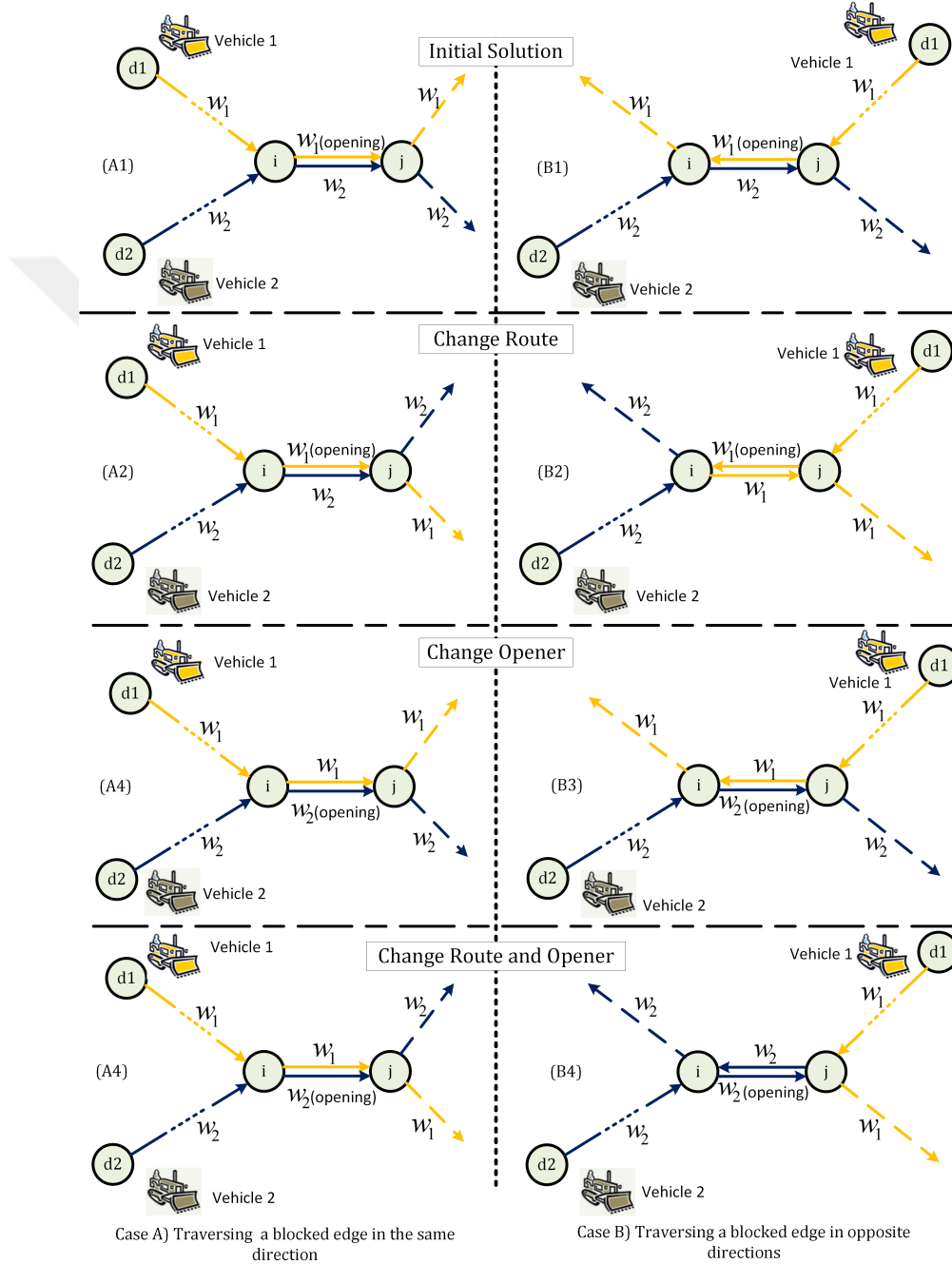
Procedure CHANGE ROUTE AND OPENER: This procedure is a hybrid of *Change Route* and *Change Openers* procedures. This procedure is depicted in Figure 5.2 (Case (A4) and (B4)). In (A4), not only the *opener* of (i, j) changes, but also the walks of vehicles 1 and 2 are inter-changed after node j . Recall that in *Change Opener* only the *opener* of (i, j) changes from vehicle 1 to 2 and the vehicles follow their original walks after node j (A3). On the other hand, in *Change Route and Opener*, not only the *opener* of (i, j) , but also the walks will change after node j . Case (B4) shows an instance of this procedure where vehicle 1 and 2 traverse (i, j) in the opposite directions. Note that it differs from Case (B2) of Figure 5.2, as the *opener* of (i, j) also changes from vehicle 1 to vehicle 2.

5.1.3 The Relaxation-Based Heuristic (RBH).

The inputs of this heuristic algorithm are the K walks derived from solving *R-MIP-I* and the output is the set of K updated and synchronized walks without timing conflicts. The steps of the proposed heuristic algorithm are given below.

The steps of Algorithm 2 are explained on an instance in Figures from 5.3 to 5.7. In these figures, the walks of three vehicles are shown by w_1, w_2 and w_3 . The two numbers on blocked edges show c_{ij} and b_{ij} corresponding to traversing and unblocking

Figure 5.2: Example to Illustrate the Local Search Procedures



(i, j) , respectively. For edges that are not blocked, only the c_{ij} value is given on the edge. The direction of traversing the edges and the current times in each walk are shown in each figure for every vehicle as well. For the considered instance, by using

Algorithm 2 Relaxation Based Heuristic for K-ARCP

Input:

the solution of *R-MIP-I*

- 1: Apply the *Walk Extraction Algorithm* on the input.
 - 2: Apply the *Feasibility Algorithm*.
 - 3: Apply the *Remove Procedure* on each individual walk of the solution.
 - 4: Apply the *Shorten and Cut Procedures* on each individual walk of the solution.
 - 5: Apply the *Feasibility Algorithm* again on the solution to get a feasible solution.
 - 6: Check if any of the *Change Route*, *Change Opener* or *Change Route and Opener* procedures are possible to apply for the current data.
 - 7:
 - **If No:** Current solution is the output of *RBH*.
 - **If Yes:** Check if this change results in better bounds:
 - *If No:* Current solution is the output of *RBH*.
 - *If Yes:* Apply the best option out of *Change Route*, *Change Opener* or *Change Route and Opener* procedures considering the length of the walks after making them feasible using the *Feasibility Algorithm*.
 - 8: Repeat Step 6.
-

Algorithm 1, we obtain the solution of *R-MIP-I* shown in Figure 5.3, where timing conflicts occur on edges (6,9) and (4,5). For this infeasible solution, the optimal value, which is the longest walk length among the three vehicles, is 80.

In the second step, after implementing the *Feasibility Algorithm*, timing of w_2 changes. The first vehicle that arrives to a blocked edge opens it and the second vehicle that traverses the same edge, should wait until the unblocking procedure is finished. As we see in Figure 5.4, the length of the longest walk, i.e. the objective function value, is 110.

In the third step shown in Figure 5.5, the route of w_2 changes. The last step of w_2 is removed since it does not involve unblocking a blocked edge due to *Remove* procedure (the third step in *RBH*). The same vehicle follows the path $8 \rightarrow 5$ instead of $8 \rightarrow 6 \rightarrow 9 \rightarrow 5$ to avoid waiting on edge (6,9), due to *Cut* procedure (the fourth step of *RBH*).

In Figure 5.6, we implement the *Feasibility Algorithm* again and it involves changing the opener on edge (4,5). We achieve 85 as the objective value in this step.

In the final step, the procedure *Change Route* is applied at node 4 so that the

Figure 5.3: R-MIP-I Solution

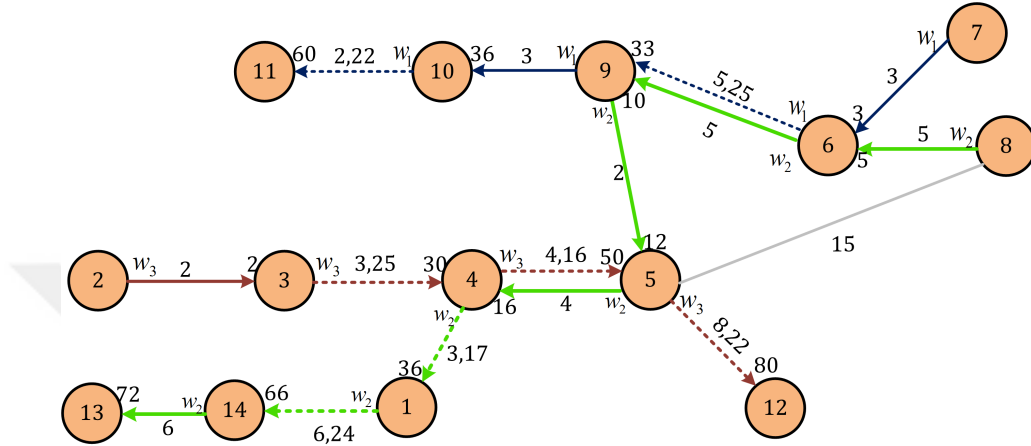


Figure 5.4: Make Routes Feasible for First Time

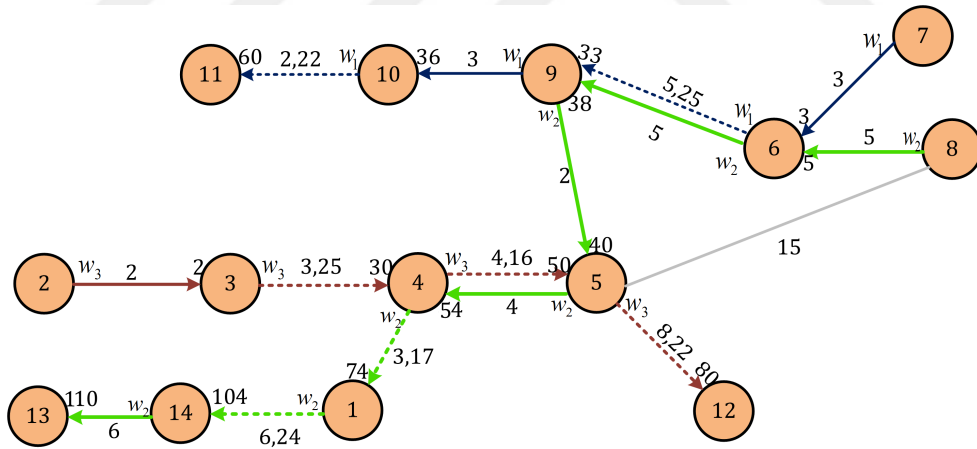


Figure 5.5: Cut and Remove Procedure

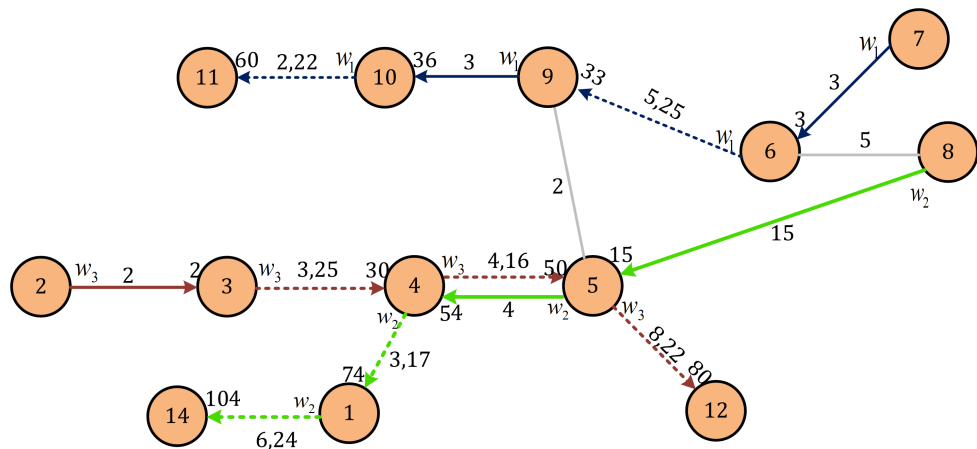


Figure 5.6: Make Routes Feasible for Second Time

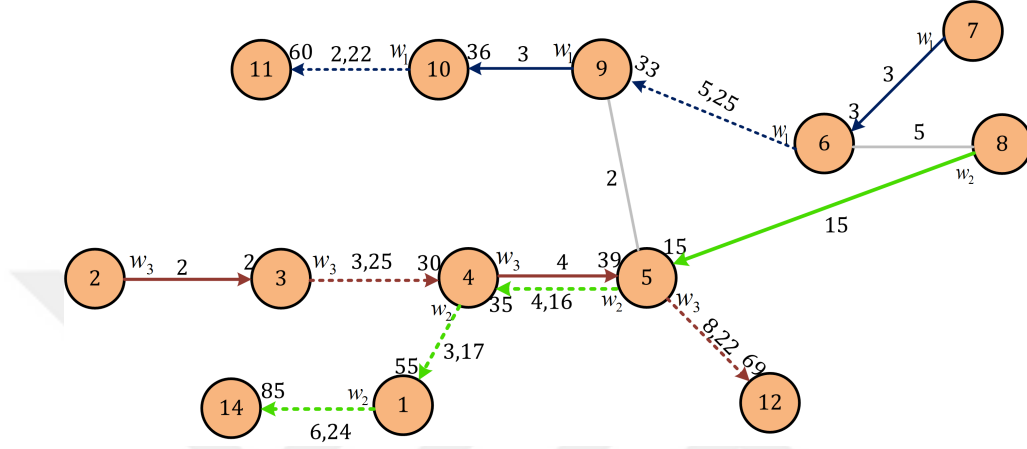
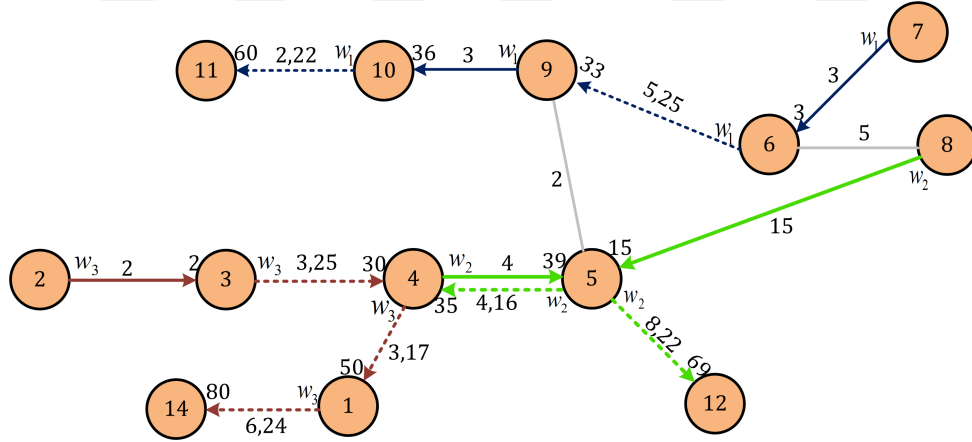


Figure 5.7: Procedure Change Route



routes of w_2 and w_3 are interchanged from node 4 onward (Figure 5.7). The final objective value which corresponds to a feasible solution derived in this example is 80, which is equal to the objective value of $R\text{-MIP-I}$. Hence, it is optimal to $K\text{-ARCP}$.

5.2 Solution Methods and Algorithms to $KPC\text{-ARCP}$

We presented $E\text{-MIP-II}$ formulation to solve $KPC\text{-ARCP}$ exactly. Since the formulation for $KPC\text{-ARCP}$ can be solved for only small-sized instances according to our computational tests, we propose a matheuristic method to solve larger instances in short time. We also introduced $R\text{-MIP-II}$ to derive upper bounds to $KPC\text{-ARCP}$.

Note that in *KPC-ARCP* we use different methods to tackle the problem. For instance, while in the heuristic algorithm to *K-ARCP*, *RBH*, we relied on the solution of *R-MIP-I*, in the heuristic procedure of solving *KPC-ARCP* we aim to find easier and faster methods to avoid solving *R-MIP-II* directly via its exact formulation.

5.2.1 Lagrangian Relaxation Method to Solve *R-MIP-II*

In the following we apply the Lagrangian Relaxation method to be able to solve larger instances of *R-MIP-II* in shorter time. While this method gives an upper bound to *R-MIP-II* and hence to *E-MIP-II*, we also obtain a lower bound to *R-MIP-II* by a heuristic algorithm. As a result, we can solve *R-MIP-II* within a proven optimality gap.

In the Lagrangian Relaxation method, a set of constraints which causes the most difficulties in the model is selected to be relaxed. In this method, a penalty cost is added to the objective function for violations of the relaxed constraints using Lagrange multipliers.

Considering that in the formulation of *R-MIP-II* only Constraints (4.73) and (4.36) relate the walks of the vehicles to each other, we relax these constraints. The Lagrange multipliers for constraints (4.73) and (4.36) are shown with $\alpha_q \geq 0, q \in Q$ and $\mu_{ij} \geq 0, (i, j) \in \bar{B}$, respectively. Let us show the Lagrangian model with $L(\mu, \alpha)$. The objective function of $L(\mu, \alpha)$ is given in (5.4).

$$Max \quad \sum_{q=1}^Q P_q y_q - \sum_{(i,j) \in \bar{B}} \mu_{ij} \sum_{k=1}^K (x_{ij}^k + x_{ji}^k - 2K(z_{ij}^k + z_{ji}^k)) - \sum_{q=2}^Q \alpha_q (y_q - \sum_{(i,j) \in \bar{B}: i \notin V_q, j \in V_q} \sum_{k=1}^K z_{ij}^k) \quad (5.4)$$

The decision variables and the constraints of $L(\mu, \alpha)$ are those of the given *R-MIP-II* model without Constraints (4.73) and (4.36).

The next step in the Lagrangian Relaxation algorithm is to choose the initial value of the Lagrange multipliers and to determine their search procedure. We use the subgradient optimization technique to update the Lagrange multipliers in each step. In the initial step of the Lagrangian Relaxation method, we solve the linear programming relaxation of the *R-MIP-II* model. We set the Lagrange multipliers, $\mu_{ij}^0, (i, j) \in \bar{B}$ and $\alpha_q^0, q \in Q$ of the initial step equal to their corresponding dual values derived from Constraints (4.73) and (4.36). Noting that s denotes the step number and we need to keep the Lagrange multipliers positive to impose the necessary penalties,

(5.5) and (5.6) define the updating of the Lagrange multipliers in each step.

$$\mu_{ij}^{s+1} = \max\{0, \mu_{ij}^s + \theta^s (\sum_{k=1}^K (x_{ij}^k(s) + x_{ji}^k(s) - 2K(z_{ij}^k(s) + z_{ji}^k(s))))\}, \forall (i, j) \in \bar{B} \quad (5.5)$$

$$\alpha_q^{s+1} = \max\{0, \alpha_q^s + \theta^s (y_q(s) - \sum_{(i,j) \in \bar{B}: i \notin V_q, j \in V_q} \sum_{k=1}^K z_{ij}^k(s))\}, \forall q \in Q \quad (5.6)$$

In (5.5) and (5.6), θ^s is the *step length* in the $(s+1)^{th}$ step of the subgradient optimization procedure. The value of θ^s can be calculated as given in (5.7).

$$\theta^s = \frac{LB_{R-MIP-II} - L^*(\mu_{ij}^s, \alpha_q^s)}{\|Ax^s - b\|^2} \quad (5.7)$$

We will explain each term in (5.7) next. $LB_{R-MIP-II}$ is a lower bound for $R-MIP-II$. We derive this lower bound using Algorithm 3 presented in the following. In Algorithm 3, $PC-ARCP-I(P_k, b_{ij})$ denotes the solution of the single vehicle case of $KPC-ARCP$ where only vehicle k is used. Note that in the single vehicle case, timing conflicts do not occur and consequently it is noticeably easier and faster to solve. It is worth mentioning that it is possible to obtain the set of cleared roads, E_k , the set of opened components by each vehicle k , O_k , and the optimal route of each vehicle, R_k , from the optimal solution of each single vehicle instance of $KPC-ARCP$ defined above.

In Algorithm 3, we solve the single vehicle problem repeatedly and update the network from the results obtained for each vehicle. These updates are related to clearing a blocked edge and connecting a component to the depot. Since we ignore timing conditions in $R-MIP-II$, the walks of the K vehicles collectively give a feasible solution to $R-MIP-II$. This means that $LB_{R-MIP-II}$ is a lower bound on the optimal objective value of $R-MIP-II$.

Proposition 5. *In the solution of Algorithm 3, if a vehicle contributes zero units of prize, then the solution of Algorithm 3 gives an upper bound to the optimal objective values of both R-MIP-II and KPC-ARCP at the same time. Hence the solution of Algorithm 3 is an optimal solution to R-MIP-II.*

Algorithm 3 $LB_{R-MIP-II}$ Algorithm

Input:
 $b_{ij}, (i, j) \in B$ $\triangleright$ Original costs of unblocking blocked edges.

 $P_q^1, q \in \tilde{Q}$ $\triangleright$ Original set of prizes, obtained by connecting each component $q \in \tilde{Q}$.

 K $\triangleright$ Number of vehicles

```

1:  $k \leftarrow 1$ 
2: while  $k \leq K$  do
3:   Solve  $PC-ARCP-I(P_k, b_{ij})$   $\triangleright PC-ARCP-I(P_k, b_{ij})$  is the single vehicle
   instance of  $KPC-ARCP$  using prize set  $P_k$  and unblocking costs  $b_{ij}$ 
4:   Extract  $E_k, O_k$  and  $R_k$  from the solution of  $PC-ARCP-I(P_k, b_{ij})$ .
    $\triangleright E_k$  is the set of cleared blocked edges by vehicle  $k$ 
    $\triangleright O_k$  is the set of components that are cleared by vehicle  $k$ 
    $\triangleright R_k$  is the optimal route for vehicle  $k$ 

5:   for  $(i, j) \in E_k$  do
6:      $b_{ij} \leftarrow 0$ 
7:   end for
8:   for  $q \in O_k$  do
9:      $P_q^{k+1} \leftarrow 0$ 
10:  end for
11:   $k \leftarrow k+1$ 
12: end while
13: return  $R_k, k = 1, \dots, K$ 

```

Proof. When a vehicle v contributes zero units of prize, it means that vehicle v does not reach any of the components that are still separated. Thus, all the components that can be reconnected in T_{max} are already connected by the routes of vehicles from 1 to $v - 1$. Let us furthermore assume that vehicle $\hat{v} \in \{1, \dots, v - 1\}$ connects the components in the set $C_{\hat{v}}$ in the solution output by Algorithm 3; where $C_{\hat{v}} = \{c \in \{2, \dots, Q\} : \sum_{(i,j) \in \bar{B}: j \in V_c, i \notin V_c} z_{ij}^{\bar{v}} \geq 1\}$. Then, $PC = \bigcup_{\hat{v} \in \{1, \dots, v-1\}} C_{\hat{v}}$ is the set of all components that are reachable in T_{max} . Since none of the components in the set $Q \setminus PC$ can be connected in T_{max} , reconnecting all the components in PC gives an upper bound on the optimal objective value of *KPC-ARCP* as well as the optimal objective value of *R-MIP-II*. $\square$

In (5.7), $L^*(\mu_{ij}^s, \alpha_q^s)$ is the optimal objective value of $L(\mu, \alpha)$ in the s^{th} step of the subgradient optimization procedure. Furthermore, $\|Ax^s - b\|$ in (5.7) denotes the Euclidean norm of the vector $Ax^s - b$ defined in (5.8). Note that $(i, j) \in \bar{B}$ and $q \in Q$, and

$$Ax^s - b = \begin{pmatrix} \sum_{k=1}^K (x_{ij}^{k,s} + x_{ji}^{k,s} - 2K(z_{ij}^{k,s} + z_{ji}^{k,s})) \\ y_q^s - \sum_{(i,j) \in \bar{B}: i \notin V_q, j \in V_q} \sum_{k=1}^K z_{ij}^{k,s} \end{pmatrix}. \quad (5.8)$$

Given the initial values of the Lagrange multipliers and the updating procedure for these multipliers, we implement the Lagrangian Relaxation method. First we solve $L(\mu, \alpha)$ with the initial multiplier values. Then in each step first we check if the current solution is optimal. If not, we update the Lagrange multipliers and then solve $L(\mu, \alpha)$ until a time limit is reached. For our tested instances we set the time limit to be one hour. In order to check the optimality of a solution $(\hat{x}, \hat{z}, \hat{y})$ and Lagrange multipliers $(\hat{\mu}_{ij}, \hat{\alpha}_q)$, we check the feasibility and the complementary slackness conditions given from (5.9) to (5.12). If a solution satisfies these conditions, then the Lagrangian Relaxation algorithm terminates. Otherwise, it terminates when

the time limit of one hour is reached.

$$\sum_{k=1}^K (\hat{x}_{ij}^k + \hat{x}_{ji}^k - 2K(\hat{z}_{ij}^k + \hat{z}_{ji}^k)) \leq 0, \forall (i, j) \in \bar{B} \quad (5.9)$$

$$\hat{\mu}_{ij} \left(\sum_{k=1}^K (\hat{x}_{ij}^k + \hat{x}_{ji}^k - 2K(\hat{z}_{ij}^k + \hat{z}_{ji}^k)) \right) = 0, \forall (i, j) \in B \quad (5.10)$$

$$\hat{y}_q - \sum_{(i,j) \in \bar{B}: i \notin V_q, j \in V_q} \sum_{k=1}^K \hat{z}_{ij}^k \leq 0, \forall q \in Q \quad (5.11)$$

$$\hat{\alpha}_q \left(\hat{y}_q - \sum_{(i,j) \in \bar{B}: i \notin V_q, j \in V_q} \sum_{k=1}^K \hat{z}_{ij}^k \right) = 0, \forall q \in Q \quad (5.12)$$

5.2.2 A Matheuristic for KPC-ARCP

In this section, we propose an easily implementable matheuristic algorithm to obtain near-optimal feasible solutions to large *KPC-ARCP* instances. In this algorithm, we solve a single vehicle problem repeatedly at each step and update the prize of the visited components to zero. With this procedure, we prevent multiple vehicles to connect the same components. We call this algorithm the *Successive Single Vehicle* algorithm and denote it as *SSV*. Note that since we use a single vehicle in each step, a timing conflict does not happen and the routes of the vehicles altogether give a feasible solution to *KPC-ARCP*. The reason will be explained later on when we compare Algorithms 3 and 4.

In Algorithm 4, namely *SSV*, $PC-ARCP-II(P_k)$ denotes the solution of the k^{th} single vehicle problem with the prize set P_k . Note that we can obtain both O_k and R_k from $PC-ARCP-II(P_k)$, where O_k is the set of connected components and R_k is the optimal route for vehicle k obtained by solving $PC-ARCP-II(P_k)$.

Proposition 6. *The output of Algorithm 4 gives a set of feasible walks to KPC-ARCP.*

Proof. The obtained walks from Algorithm 4 gives a set of edges and blocked edges to traverse for each vehicle. In these walks, traversing a blocked edge incurs the extra cost of unblocking it in all the cases. However, a blocked edge should be opened only

Algorithm 4 Successive Single Vehicle Algorithm (*SSV*)**Input:**

$P_q^1, q \in \tilde{Q}$ $\triangleright$ Original set of prizes, obtained by connecting each component $q \in \tilde{Q}$.
 K $\triangleright$ Number of vehicles

```

1:  $k \leftarrow 1$ 
2: while  $k \leq K$  do
3:   Solve  $PC\text{-}ARCP\text{-}II(P_k)$   $\triangleright PC\text{-}ARCP\text{-}II(P_k)$  is the single vehicle instance of
      $KPC\text{-}ARCP$  with prize set  $P_k$ .
4:   Extract  $O_k$  and  $R_k$  from the solution of  $PC\text{-}ARCP\text{-}II(P_k)$ .
      $\triangleright O_k$  is the set of components that are cleared by vehicle  $K$ 
      $\triangleright R_k$  is the optimal route for vehicle  $k$ 
5:   for  $q \in O_k$  do
6:      $P_q^{k+1} \leftarrow 0$ 
7:   end for
8:    $k \leftarrow k+1$ 
9: end while
10: return  $R_k, k = 1, \dots, K$ 

```

with one vehicle. Let us assume that multiple vehicles traverse a common blocked edge, $(i, j) \in B$. We can choose the opener of (i, j) using Definition 1 which implies that the vehicle that arrives to (i, j) earlier than other vehicles is the opener of (i, j) . Let us assume vehicle k is set to be the opener of (i, j) and it arrives to (i, j) at $T_{(i,j)k}$. Hence the clearing process of (i, j) will be finished at $T_{(i,j)k} + b_{ij} + c_{ij}$. It implies that other vehicles can finish traversing (i, j) at least at $T_{(i,j)k} + b_{ij} + c_{ij}$. Since the arrival time of other vehicles to (i, j) is larger than $T_{(i,j)k}$ and they should spend $c_{ij} + b_{ij}$ for traversing (i, j) , they can not finish traversing (i, j) sooner than $T_{(i,j)k} + b_{ij} + c_{ij}$, which is the time that clearing process of (i, j) is finished. Thus, timing conflicts does not occur and the walks are synchronized and feasible to $KPC\text{-}ARCP$. $\square$

The difference between Algorithms 3 and 4 is that, Algorithm 4 gives a lower bound to $KPC\text{-}ARCP$ but Algorithm 3 gives a lower bound to $R\text{-}MIP\text{-}II$. This is because in Algorithm 4 we do not update the opened blocked edges in each step. Hence, none of the vehicles traverses an edge unless its clearing process is finished or the same vehicle is responsible for clearing it. As a result, the collection of the routes obtained from Algorithm 4 gives a feasible solution to $KPC\text{-}ARCP$ and the feasible

solution to *KPC-ARCP* gives a lower bound on it.

Proposition 7. *If a vehicle contributes zero units of prize in the solution output by Algorithm 4, then the solution of Algorithm 4 is an optimal solution to KPC-ARCP.*

Proof. If a vehicle v connects none of the components in Algorithm 4, it means that it does not leave the depot. Since only when a vehicle does not leave the depot that it can not reach to a separated component with a positive prize within T_{max} , all the components that are reachable in T_{max} must have been connected by the vehicles before v . It means that a collection of the routes of the vehicles 1 to $v - 1$ connect all the components that can be connected within T_{max} and are also feasible to *KPC-ARCP* by Proposition 6. Hence, these routes give an optimal solution to *KPC-ARCP* according to Definition 3. $\square$

We utilize Propositions 5 and 7 in our computational results.

We demonstrate Algorithms 3 and 4 by an example in the following. Figure 5.8 is an example of *KPC-ARCP*. Four vehicles are originally positioned in the depot node denoted by D . These vehicles should be dispatched and restore the connectivity of components to maximize the total prize gained in a given time limit. The traversal time (c_{ij}) and the clearing time (b_{ij}) is given for this example in Table 5.2. These times are in minutes and the time limit is set to 5 hours ($T_{max} = 300$). The prize of each component is set to be equal to the number of nodes in that particular component.

Table 5.2: Traversing and Clearing Times of the Example

Intact Edges	c_{ij}	Intact Edges	c_{ij}	Blocked Edges	c_{ij}	b_{ij}	Blocked Edges	c_{ij}	b_{ij}
(D,2)	10	(11,13)	15	(D,7)	20	100	(5,14)	30	135
(D,3)	20	(12,13)	30	(D,11)	50	240	(5,20)	20	120
(1,3)	5	(15,16)	15	(D,5)	25	100	(8,9)	25	140
(3,4)	5	(15,17)	15	(1,2)	15	80	(14,20)	30	160
(6,7)	30	(17,18)	10	(1,15)	15	50	(15,19)	40	200
(6,8)	15	(19,20)	10	(3,6)	15	80	(16,17)	15	85
(7,8)	20	(21,22)	15	(4,16)	40	230	(18,24)	30	150
(9,10)	10	(21,23)	10	(4,22)	20	260	(21,24)	30	150
(11,12)	20			(5,12)	20	100	(22,23)	15	85

Figure 5.9 gives the solution obtained using Algorithm 3 for the given example. In

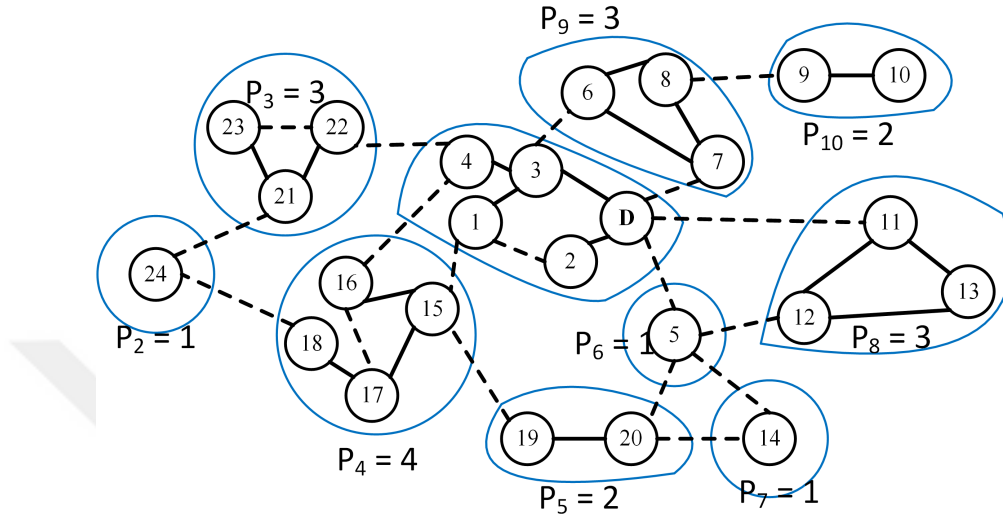


Figure 5.8: An Example to Illustrate Algorithms 3 and 4.

these graphs, if a vehicle opens an edge, it is shown by a dashed arrow and otherwise, it is shown by a solid arrow. The figure with caption "a) Vehicle 1" shows the walk of the first vehicle. It opens (3,6) and (8,9). The total traversal time for vehicle 1 is $30 (c_{6,7}) + 15 (c_{3,6}) + 80 (b_{3,6}) + 15 (c_{6,8}) + 25 (c_{8,9}) + 140 (b_{8,9}) = 295$ and since its less than 300, it does not violate the tour-length constraint. Vehicle 1 connects nodes 6,7,8,9 and 10 to the depot and obtains 5 units of prize. After the walk of the first vehicle is finished, the network will be updated and edges (6,8) and (8,9) are not blocked anymore. In "b) Vehicle 2", $D \rightarrow 3 \rightarrow 1 \rightarrow 15 \rightarrow 17 \rightarrow 18 \rightarrow 24$ is the path of the second vehicle. The second vehicle unblocks (1,15) and (18,24) and obtains 5 units of prize by connecting 15,16,17,18 and 24 to the depot node. The tour length of the second vehicle is 295 and is less than T_{max} . "c) Vehicle 3" shows the walk of the third vehicle. It unblocks (D,5) and (5,12) in 245 units of time and obtains 4 units of prize by connecting 5,11,12 and 13 to the depot node. Note that it can not open any other components in T_{max} hence, it finishes its walk after visiting node 12. In "d) Vehicle 4", the fourth vehicle traverses $D \rightarrow 3 \rightarrow 1 \rightarrow 15 \rightarrow 17 \rightarrow 18 \rightarrow 24 \rightarrow 21$. It only opens (24,21) since (1,15) and (18,24) are already opened by the second vehicle. The tour length of vehicle 4 is 275, which is less than T_{max} ; hence, we have a feasible

walk. Vehicle 4 obtains 3 units of prize by connecting 21, 22 and 23 to the depot.

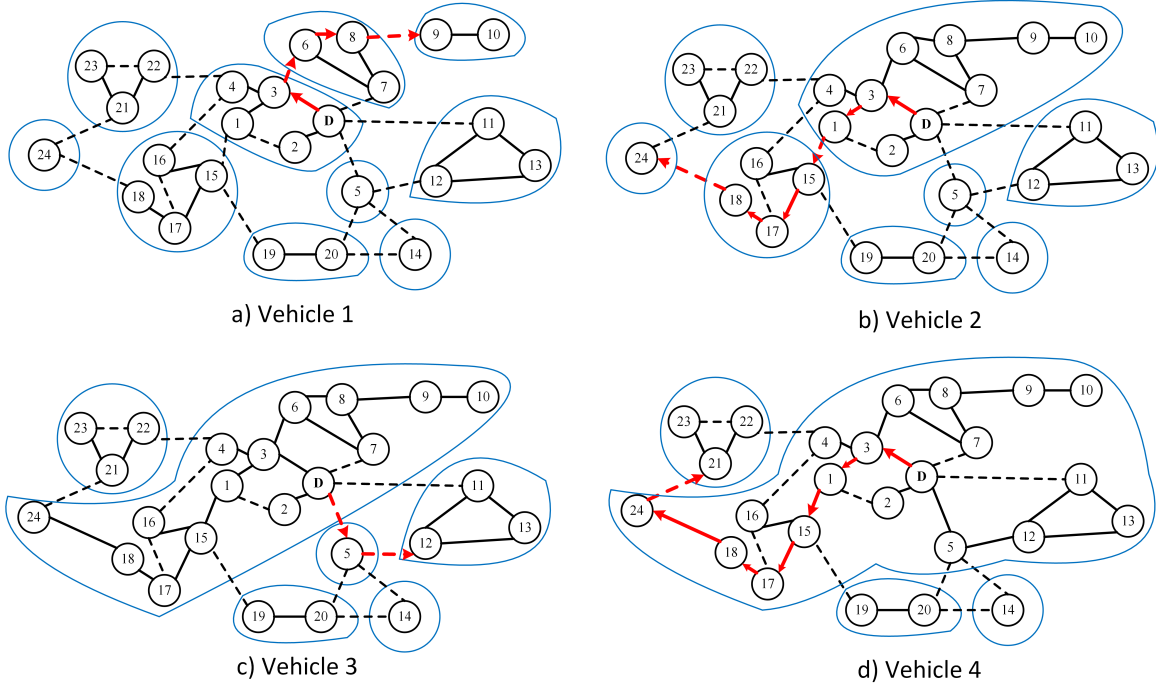


Figure 5.9: Demonstration of Algorithm 3

Figure 5.10 shows the obtained walks of all of the 4 vehicles using Algorithm 3 in the given example. The vehicles all together obtain 17 units of prize in 300 minutes. This corresponds to a feasible solution to *R-MIP-II*, in which time conflicts are allowed.

Figure 5.11 illustrates the implementation of Algorithm 4 on the same example. Different from Algorithm 3 in Figure 5.9, here the blocked edges will not be updated after they are traversed by a vehicle. For instance, vehicle 1 in "a) Vehicle 1" unblocks (3,6) and (8,9) but in the next step in "b) Vehicle 2" these edges are not assumed to be cleared but the prizes of their corresponding components are updated to zero to avoid connecting the same components in the walks of the next vehicles. Note that different from Figure 5.9, the fourth vehicle can not reach nodes 21 or 22 in T_{max} and it connects nodes 19 and 20 instead and obtains less total prize.

Figure 5.12 gives the obtained walks of all of the 4 vehicles using Algorithm 4 in

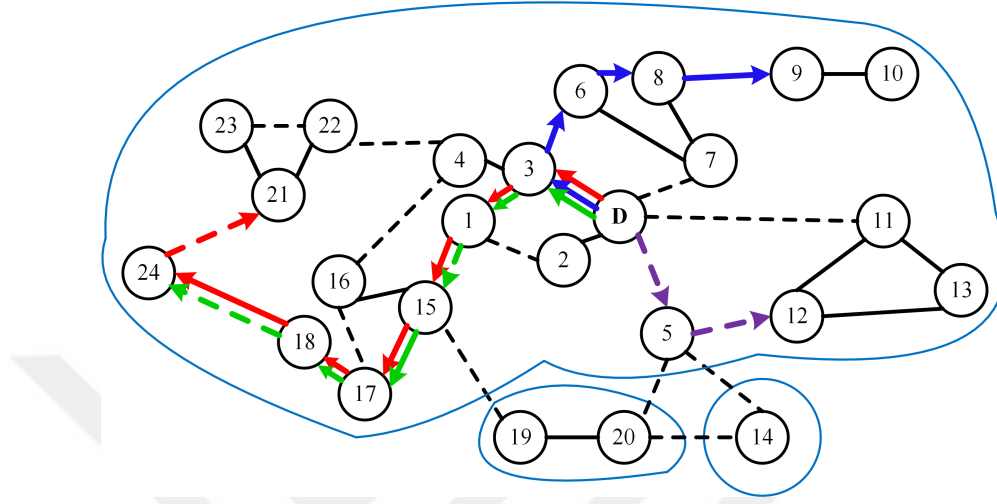


Figure 5.10: Solution Output by Algorithm 3

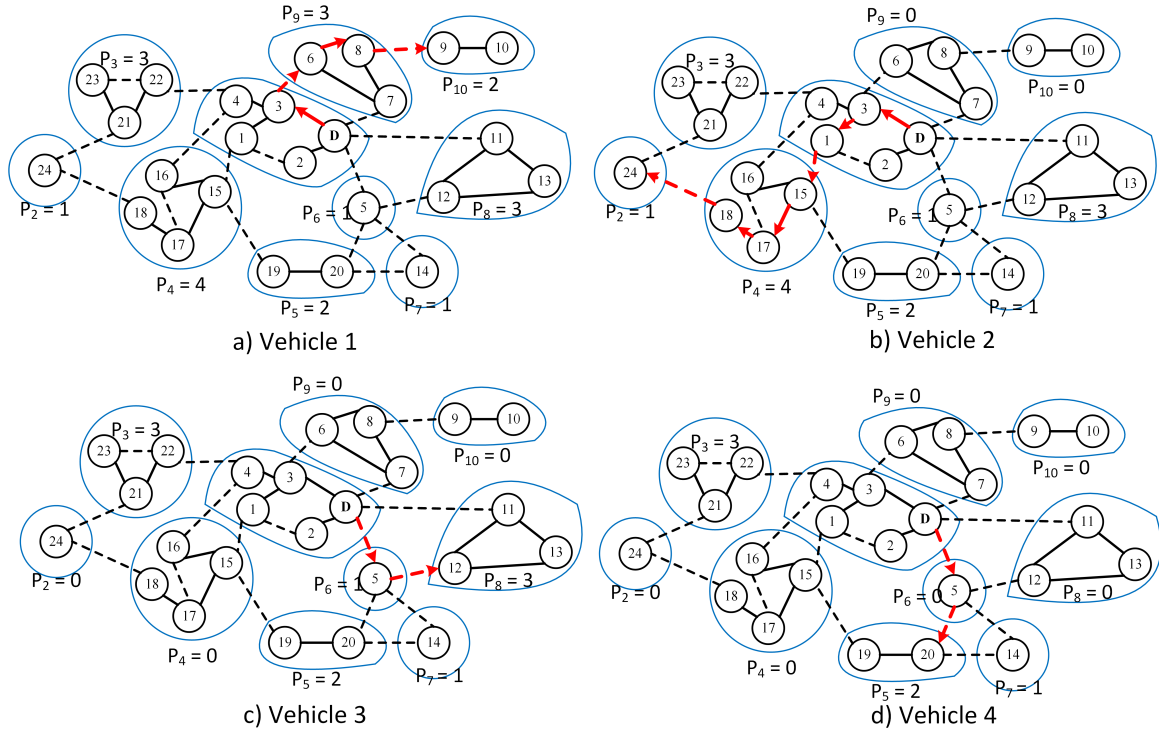


Figure 5.11: Demonstration of Algorithm 4.

the given example. The vehicles all together obtain 16 units of prize in 300 minutes. Vehicle 1 connects 6,7,8,9 and 10, vehicle 2 connects 15,16,17,18 and 24, vehicle 3

connects 5,11,12 and 13, and finally vehicle 4 connects 19 and 20 to the depot node. This is a feasible solution to *KPC-ARCP*.

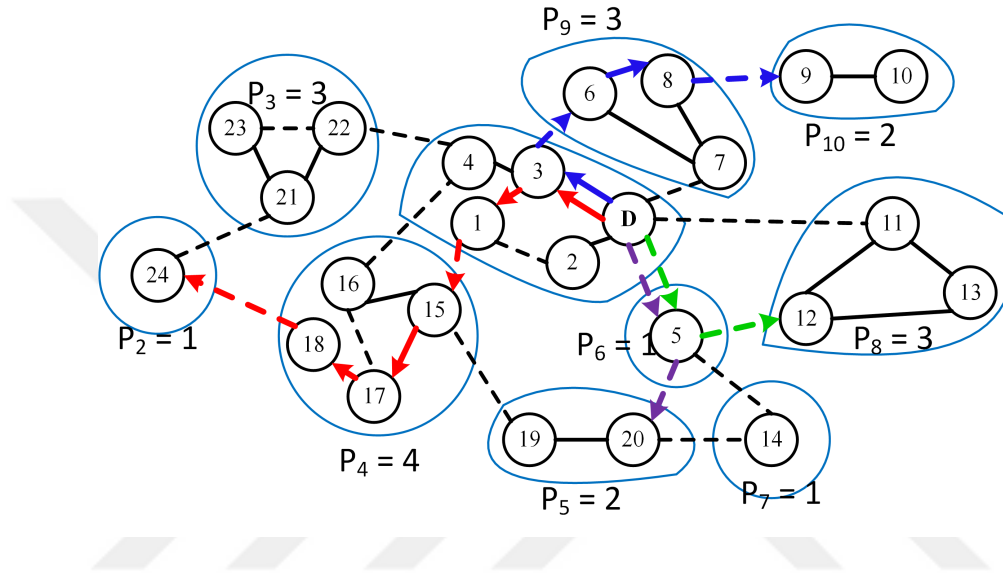


Figure 5.12: Solution Output by Algorithm 4.

Chapter 6

DESIGN OF COMPUTATIONAL EXPERIMENTS

In order to test the performance of the of the heuristic algorithms described in this thesis, we conducted a computational study. There are no benchmark instances in the literature for the problems discussed in this thesis. Hence, some real-life data from Istanbul city road network is generated and tested in this work. Moreover, we generated separate random instances to test the performance of the developed algorithms and mathematical models for *K-ARCP* and *KPC-ARCP*.

6.1 *Istanbul Networks*

Three different data sets are generated from the road network of Istanbul city. To specify a node set, using ArcGIS software, we identified strategically important locations such as province centers, disaster coordination centers, harbors, airports, bus terminals, fire stations, hospitals. These points are the potential supply points (yellow squares in Figure 6.1) and their connectivity to the demand points is important. In Figure 6.1, the blue points denote the junction points (an intersection point where several or different types of routes meet or link) and red points denote the demand points (district centers). The set of edges is composed of shortest routes among demand points, potential supply points and road junction points in the network. Distances are symmetric, that is, the travel times are the same for both directions for a pair of nodes.

As it is mentioned above, three different data sets are generated from the road network of Istanbul city. The first data set is based on a simplified Istanbul network with 74 nodes and 179 edges, considering the main highways and their connections to various demand centers (Figure 6.3). The second data set is based on a more

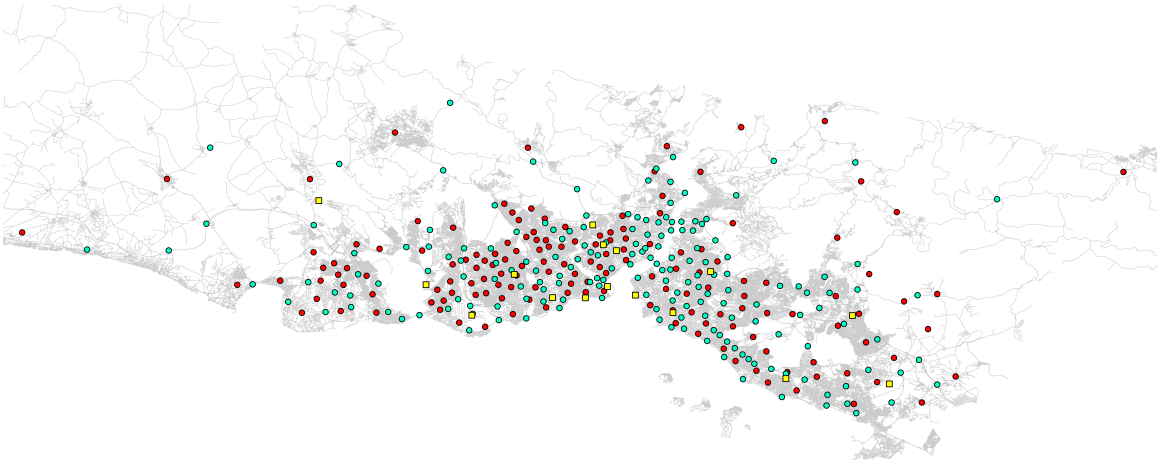


Figure 6.1: Istanbul network nodes

detailed Istanbul road network with 349 nodes and 689 edges (Figure 6.2). Finally, the third data set is based on a more detailed road network of the Southwestern region of Istanbul city with 250 nodes and 539 edges (Figure 6.4). Size of these data sets are given in Table 6.1. We note that these instances are much larger than those tested in related studies in the debris removal literature (e.g. [Özdamar et al., 2014], [Sahin et al., 2016]).

Table 6.1: Characteristics of the data sets

Number	Description	# of Nodes	# of Edges	# of Potential Depot Nodes
1	Simplified Istanbul	74	179	16
2	Detailed Istanbul	349	689	16
3	Southwestern Istanbul	250	539	5

Figure 6.2 shows the detailed Istanbul network and the active North Anatolian Fault line that is under the Marmara Sea that lies to the south of Istanbul. The edges are categorized into three different groups due to their proximity to the epicenter of the earthquake according to the scenarios predicted in the [JICA, 2002] report, as high, medium and low-risk edges. The probability that an edge from the low-risk

class is blocked after an earthquake is set to 0.1, while this probability is set to 0.2 and 0.3 for medium and high-risk edges, respectively. Edges located at the top of the higher horizontal line in Figure 6.2 lie in the group of low-risk edges; those located between the two horizontal lines are in the group of medium-risk edges; and those under the lower horizontal line are high-risk edges. If nodes of an edge are located in two different groups, we placed that particular edge in the more risky group.

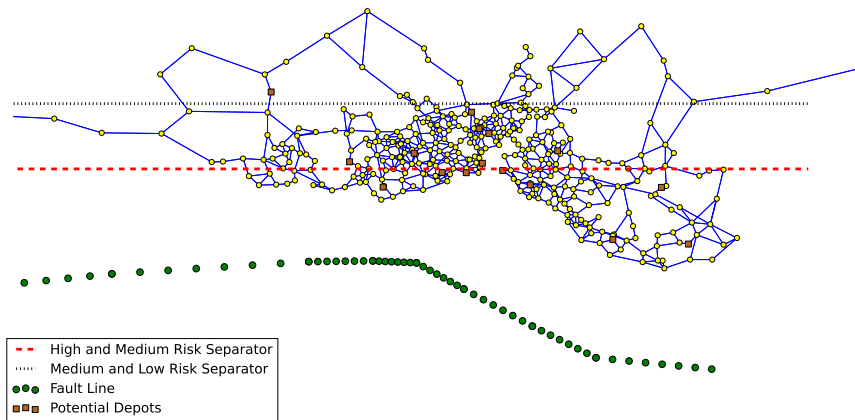


Figure 6.2: Detailed Istanbul Network and the Fault Line

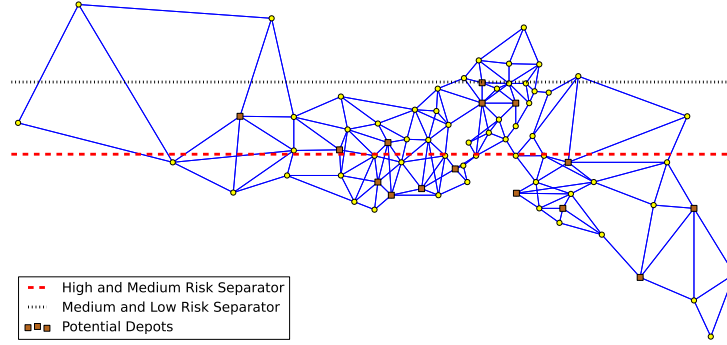


Figure 6.3: Simplified Istanbul Network

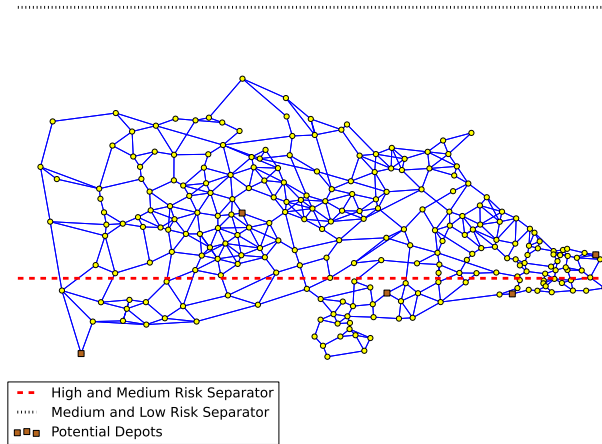


Figure 6.4: Southwest Istanbul Road Network

While in *K-ARCP* there might be multiple depots, in *KPC-ARCP*, the vehicles should be dispatched from one depot node. For the simplified and detailed Istanbul road networks, we selected 16 potential depot nodes as shown in Figure 6.2. Among these 16 nodes, 5 of them are located in the Southwestern part of the Istanbul road

network. In different instances, the depot of each of the vehicles is chosen randomly among the potential depots in *K-ARCP* and it remains the same for all the vehicles in *KPC-ARCP*.

For these three sets of data, the traversal cost c_{ij} on edge (i, j) is equal to the time it takes for a work troop to go from node i to node j . We assumed that the speed of vehicles is 50 *km/h* on the average. The distance between nodes i and j is calculated using ArcGIS. The opening cost, b_{ij} , on edge (i, j) is a random number generated according to: $b_{ij} = c_{ij} \times X$, where X has uniform distribution between 10 and 30.

Finally, in *KPC-ARCP* the value of T_{max} is determined by solving the single vehicle problem of minimizing the time to connect all the components according to the model provided in [Kasaei, 2015]. If we show this tour length by TL , then we set $T_{max} = 0.8 \times \frac{TL}{3}$. We chose this value to create a trade off since having T_{max} too small might prevent connecting any of the components or having it too large results in connecting all the components.

6.2 Random Data Sets

By generating random data, we pursue different purposes for *K-ARCP* and *KPC-ARCP*. Hence, we generated distinct sets of random network for each of them.

6.2.1 Random Data Sets for *K-ARCP*

As the first set of random networks, for *K-ARCP* we generated Euclidean random graphs with 20, 25, 30 and 40 nodes and 1 to 4 vehicles in order to test *E-MIP-I*. For each number of nodes and each number of vehicles selected, we tested 10 random instances. First, we assigned random coordinates in a $100 \cdot 100$ plane to every node. Costs, c_{ij} , are equal to the Euclidean distances. The extra cost, b_{ij} , is generated according to: $b_{ij} = c_{ij} \cdot X$ where X has uniform distribution between 10 and 30. In each case, an edge (i, j) exists if distance between nodes i and j is lower than 20 units. Since one of the primary assumptions is that the graph $G = (V, E)$ is connected, if the generated graph is not connected, we add random edges between

connected components to make G connected. These random edges are added with the same probability. Depots are also chosen randomly with equal probability. According to the problem definition, $G_B = (V, E \setminus B)$ is a disconnected graph consisting of Q connected components. We impose $Q \geq 2K$ in our instances in order to increase the possibility of assigning at least one unblocking task to every vehicle. At first we add the added random edges (the edges that has been added to make G a connected graph) to the set B . Then edges are randomly added to the set B with equal probabilities to separate G_B into the desired number of connected components.

In order to compare the performance of *RBH* with different number of connected components and vehicles, we generated another random set of 50 random networks with 50 nodes using the same method as described above. We solved each of these instances with 1 to 6 vehicles to test the performance of *RBH* with more vehicles.

6.2.2 Random Data Sets for KPC-ARCP

We generated random graphs with Euclidean distances having 200, 250, 300 and 350 nodes in order to test the proposed algorithms with different network topologies. The impacts of changing the density of the network and the number of blocked edges are also tested. For each number of nodes we tested 10 random instances and solved each of these instances with 1 to 5 vehicles and a fixed depot node. First, we assigned random coordinates to each node in a 1000×1000 plane. The costs, c_{ij} , are taken equal to the Euclidean distances. The extra cost, b_{ij} , is generated according to: $b_{ij} = c_{ij} \cdot X$ where X has uniform distribution between 10 and 30. In each case, an edge (i, j) exists if the distance between nodes i and j is at most a threshold value. For 200 nodes we set the distance threshold to be 85; for 250 nodes, 80; for 300 nodes, 75 and finally for 350 nodes it is set to 70. These random networks are denser than the Istanbul road networks. Since one of the primary assumptions is that the graph $G = (V, E)$ is connected, if the generated graph is not connected, we add random edges between connected components to make G connected. The depot is also chosen randomly in each instance. According to the problem definition, $G_B = (V, E \setminus B)$

is a disconnected graph consisting of Q connected components. Edges are randomly added to the set B with equal probabilities to separate G_B into the desired number of connected components. The numbers of edges and blocked edges do not vary much over the 10 instances except in a particular instance with 300 nodes where 240 edges are blocked.



Chapter 7

COMPUTATIONAL RESULTS

In order to evaluate the performance of the heuristic algorithms described in this thesis, we generated bounds by solving our proposed mathematical models with a run-time limit. In this chapter, we present the results of the tested instances on *K-ARCP* and *KPC-ARCP* and analyses on the solution characteristics.

7.1 Results of Tested Instances on *K-ARCP*

The results of the tested instances on the Istanbul roads networks for *K-ARCP* are given in Table 7.1.

7.1.1 Results for the Simplified Istanbul Network.

The figure of the simplified version of the Istanbul road network is given in Figure 6.3. For this network we tested 15 instances with the number of connected components varying between 3 to 8. We examined the performance of *RBH* with 1,2,3 and 4 vehicles for each of these 15 instances.

Left side of Table 7.1 shows the results for the first set of instances. While the first and the second columns in Table 7.1 show the instance number, number of connected components and the number of vehicles in the corresponding instance, the third column shows the elapsed time by the Gurobi solver to solve *R-MIP-I* optimally for each scenario. The time is in seconds and all the scenarios were tested on an Intel Xeon E5-2643 CPU @ 3.30GHz 3.30GHz (two processors) computer with 32 GB RAM.

Recall that the solution to *R-MIP-I* gives a lower bound on the optimal objective value of *K-ARCP*. We show this lower bound as Z_{R-MIP}^* . *RBH* is the solution obtained after local improvement steps, that gives an upper bound for *K-ARCP*. We show the

Table 7.1: Results for the Istanbul Roads Networks

Simplified Roads Network					Detailed Roads Network					Southwest Roads Network				
Number(Q)	K	Time	Gap	LSI	Number(Q)	K	Time	Gap	LSI	Number(Q)	K	Time	Gap	LSI
1(7)	1	32.2	0	-	16(4)	1	2002.00	0	-	31(4)	1	58.66	0	-
	2	252.87	0	-		2	23.84	0	-		2	17.77	0	-
	3	160.11	0	-		3	118.50	0.0218	22.32%		3	16.75	0	-
	4	15.81	0.0159	18.34%		4	-	-	-		4	-	-	-
2(5)	1	4.87	0	-	17(5)	1	692.31	0	-	32(6)	1	41.97	0	-
	2	194.25	0	-		2	946.50	0	-		2	36.25	0	-
	3	59.22	0	-		3	3500.96	0	-		3	69.18	0	-
	4	8.95	0	22.84%		4	107.04	0	47.12%		4	196.93	0	-
3(6)	1	0.29	0	-	18(6)	1	90.80	0	-	33(5)	1	23.87	0	-
	2	3.68	0	-		2	105.82	0	-		2	35.84	0	-
	3	10.24	0	-		3	102.40	0	-		3	38.12	0	-
	4	15.2	0	30.64%		4	151.18	0	-		4	66.27	0	-
4(4)	1	1.41	0	-	19(4)	1	13.25	0	-	34(4)	1	42.61	0	-
	2	11.18	0	-		2	403.58	0	-		2	12.05	0	-
	3	2.26	0	-		3	43.03	0	-		3	36.63	0	-
	4	-	-	-		4	-	-	-		4	-	-	-
5(6)	1	1.9	0	-	20(4)	1	3.93	0	-	35(4)	1	13.91	0	-
	2	3.56	0	-		2	133.79	0	-		2	608.71	0	-
	3	4.21	0	-		3	32.81	0	-		3	48.24	0	-
	4	4.45	0	-		4	-	-	-		4	-	-	-
6(7)	1	2.62	0	-	21(4)	1	95.15	0	-	36(5)	1	1020.74	0	-
	2	75.62	0.0111	5.10%		2	26.67	0.004	35.39%		2	259.02	0	-
	3	665.68	0	-		3	100.15	0	-		3	35.81	0	-
	4	166.21	0.002	28.09%		4	-	-	-		4	37.35	0	-
7(5)	1	5.84	0	-	22(6)	1	104.97	0	-	37(6)	1	91.81	0	-
	2	20.12	0	-		2	3796.17	0	-		2	419.75	0	-
	3	34.12	0	-		3	547.24	0	-		3	2003.33	0	-
	4	24.12	0	-		4	444.72	0	-		4	750.01	0	-
8(4)	1	4.54	0	-	23(6)	1	19.36	0	-	38(3)	1	376.29	0	-
	2	3.42	0	-		2	2772.21	0	-		2	37.02	0.0021	25.03%
	3	6.9	0.0044	5.39%		3	1572.22	0	-		3	-	-	-
	4	-	-	-		4	1117.83	0.0451	36.93%		4	-	-	-
9(3)	1	0.37	0	-	24(5)	1	8.00	0	-	39(4)	1	6.75	0	-
	2	3.75	0.0051	21.66%		2	25.71	0	-		2	16.18	0.0018	15.26%
	3	-	-	-		3	226.60	0	-		3	22.16	0	-
	4	-	-	-		4	85.11	0	-		4	-	-	-
10(8)	1	0.93	0	-	25(4)	1	11.02	0	-	40(4)	1	9.61	0	-
	2	5.37	0	-		2	16.46	0.0041	31.95%		2	19.58	0	-
	3	6.68	0	-		3	29.09	0	-		3	30.61	0	-
	4	123.48	0.1818	35.61%		4	-	-	-		4	-	-	-
11(8)	1	5.65	0	-	26(6)	1	35.85	0	-	41(6)	1	86.32	0	-
	2	6.74	0.0185	10.99%		2	4044.42	0	-		2	27.42	0	-
	3	4.78	0.0277	15.08%		3	1095.40	0	-		3	120.22	0	-
	4	8.95	0.0043	33.25%		4	1303.07	0.1044	36.30%		4	59.25	0	-
12(6)	1	7.44	0	-	27(5)	1	11.48	0	-	42(6)	1	3.69	0	-
	2	3.42	0	8.41%		2	161.29	0	-		2	24.85	0	-
	3	15.21	0.0085	43.99%		3	54.17	0	-		3	63.54	0	-
	4	21.27	0.0063	34.71%		4	145.68	0	-		4	179.61	0	-
13(6)	1	9.91	0	-	28(4)	1	28.74	0	-	43(4)	1	18.76	0	-
	2	92.67	0	-		2	72.87	0	-		2	38.27	0.0114	18.41%
	3	258.91	0	-		3	30.30	0	-		3	43.06	0	40.64%
	4	6.42	0	-		4	-	-	-		4	-	-	-
14(8)	1	4.32	0	-	29(6)	1	31.63	0	-	44(3)	1	3.08	0	-
	2	36.18	0.0204	20.24%		2	35.78	0	-		2	112.73	0	-
	3	10.79	0	-		3	450.62	0	-		3	-	-	-
	4	9.34	0	-		4	10748.72	0	-		4	-	-	-
15(4)	1	38.14	0	-	30(6)	1	16.15	0	-	45(5)	1	10.2	0	-
	2	61.12	0	-		2	6219.57	0	-		2	16.71	0	-
	3	17.49	0	-		3	600.47	0	26.56%		3	17.43	0	-
	4	-	-	-		4	685.17	0.0253	45.54%		4	34.48	0	-

objective values of the feasibility algorithm by Z_f and the upper bound by RBH . Then the fifth column in Table 7.1, denoted by LSI, denotes the Local Search Improvement and gives the percentage of the improvement made by the local search algorithm introduced in Section 5.1.2. This value is derived as follows:

$$LSI(\%) = \frac{Z_f - Z_{RBH}}{Z_{RBH}} \cdot 100 \quad (7.1)$$

Note that if the optimal solution of $R-MIP-I$ is feasible for $K-ARCP$, then we can not have any improvements. In such cases, this column is filled by '-' sign.

The fourth column in Table 7.1 shows the optimality gap for each scenario, which is a function of the upper and lower bounds as follows:

$$\text{Optimality Gap}(\%) = \frac{\text{Upper Bound} - \text{Lower Bound}}{\text{Lower Bound}} \cdot 100, \quad (7.2)$$

which is equivalent to $(Z_{RBH} - Z_{R-MIP})/(Z_{RBH})$.

In Table 7.1 we see that RBH obtained an optimal solution in 41 instances out of 55 of the Istanbul Simplified roads network. Note that the single vehicle problem is guaranteed to be solved optimally. Over all cases, this takes 8 seconds on the average. As the number of vehicles increases, solutions to certain instances yield optimality gaps. The optimal walks from $R-MIP-I$ in 15 instances out of 55 tested instances were not synchronized. Hence the local search procedure is used in these 15 instances. The local search procedure improved the result in all of these instances. The average improvement among these 15 instances is 22.28%. As a result, in those instances where the solution is not optimal, the average optimality gap is 2.13%. Except for instance 10 having 4 vehicles, over all other instances the gap is at most 2.77%.

As the number of connected components change from 3 to 8, we see that usually the problem gets more difficult. While for any of these scenarios with multiple vehicles, $E-MIP-I$ falls short to solve the problem optimally in two hours, we managed to derive good solutions using RBH in a very short time (12 minutes at most and 46.77 seconds on the average). We think that these times are reasonable in our context.

7.1.2 Detailed Istanbul Network.

As in the simplified Istanbul network case, we tested 15 different instances with 1,2,3 and 4 vehicles. The middle section of Table 7.1 gives the results for these instances. Columns are labelled as described in Section 7.1.1. Here we note that more than 4 vehicles may be used in post-disaster response. However, our purpose is to show the trend of performance of the algorithm. Therefore, we tested the algorithm with up to 4 vehicles.

In Table 7.1 we see that our method obtained an optimal solution in 48 instances out of 54 tested instances of the Detailed roads network. The optimal walks from *R-MIP-I* did not result in feasible walks in 8 instances out of the 54 tested cases. The local search improvement was 35.26 % in these 8 instances. Although in those instances where the solution is not optimal the average optimality gap is 3.41%, the average optimality gap over all instances is 0.37%. Similar to the results in Section 7.1.1, all instances with a single vehicle were solved optimally as timing conflict does not occur in them. As the number of vehicles changes from 2 to 4, we see that the optimality gap usually increases. While in instances with 2 or 3 vehicles, the largest gap is less than 0.5%, in instances with 4 vehicles, the average gap is 1.94%.

We see that even solving *R-MIP-I* is difficult with this size of the network (349 nodes and 689 edges), especially with more vehicles. The average run time to solve *R-MIP-I* is less than 15 minutes. Aside from instance 14 with 4 vehicles, which took almost 3 hours to solve, all other tested instances were solved in less than 2 hours. In cases where the run times are long, a reasonable approach would be to decompose the network.

7.1.3 Southwestern Istanbul Network.

We also tested 15 instances with 1,2,3 and 4 vehicles. The right side of Table 7.1 shows the results of the tested instances for the Southwest Istanbul road network. The columns of this part of the table are labeled as explained. In Table 7.1 for the tested instances of the southwest roads network we see that *RBH* derived an optimal

solution in 47 out of 50 tested instances. In four of the tested instances, the optimal solution to *R-MIP-I* did not correspond to feasible walks. In these four instances, applying local search improved the result by an average of 24.83%. In instances where the solution is not optimal, the average optimality gap is only 0.51%. The maximum optimality gap is 1.14% and the average gap over all instances is 0.01%. Note that since the number of vehicles changes for a specific instance number, the depot location is chosen randomly for each vehicle. These random assignment causes positive optimality gaps in instances 9 and 13 with 2 vehicles, while there is no gap in these instances when more vehicles exist.

Instances of this data set are mostly easier to solve than instances of the detailed Istanbul road network that has a larger size. The average run time to solve *R-MIP-I* over all instances of the regional network is less than 3 minutes. In general, as the number of vehicles increases from 1 to 4, *R-MIP-I* run-time increases. The maximum elapsed time to solve *R-MIP-I* over all tested instances is less than 35 minutes.

7.1.4 Result of Random Data Sets for *K-ARCP*

Table 7.2 shows the results comparing *E-MIP-I* formulation, *R-MIP-I* and *RBH* for the first set of random data with 20, 25, 30 and 40 nodes and 1 to 4 vehicles. While columns 1 to 6 show the properties of the generated instances for each set of nodes and vehicles, columns 7 to 10 give the results regarding *E-MIP-I* and *RBH*. In the first column, K shows the number of vehicles. In columns 3 and 4, $\bar{Q}$ and V show the average number of connected components and the number of nodes for every ten instances, respectively. $\bar{E}$ and $\bar{B}$ give the average number of edges and average number of blocked edges over ten instances, respectively. Note that when the number of nodes increases for a particular number of vehicles, the density of the graph increases considerably, as it can be seen in column $\bar{E}$. We set a time limit of 1 hour for all these 160 instances and tested whether *E-MIP-I* can solve them in this time limit or not. The # of Solved Instances by *E-MIP* (in 1 hour) column, reports this value among every ten instances with a particular number of vehicles and nodes. The

column *Average E-MIP-I Time* gives the average time spent by *E-MIP-I* to solve ten instances. The last two columns show the average elapsed time by *R-MIP-I* to find an initial solution and the average gap over ten instances using *RBH*, respectively.

As we can see in Table 7.2, all instances with a single vehicle were solved with *E-MIP-I* within 3 minutes on average. As expected, *RBH* solved them all optimally in much shorter time (in less than a second on average). The elapsed time to solve *E-MIP-I* is considerably more than *R-MIP-I*, especially when the number of vehicles increases. For the case with 40 nodes and one vehicle, *E-MIP-I* takes 168.79 seconds on the average, whereas *R-MIP-I* solves them in only 0.27 seconds on the average.

In cases with two vehicles, *E-MIP-I* solves all the instances with 20 and 25 nodes and solves 9 instances out of 10 with 30 nodes. Meanwhile, we can see that *E-MIP-I* falls short of solving instances with 40 nodes in 7 instances out of 10. While *E-MIP-I* solves these 40 instances with two vehicles in 1205.08 seconds on the average, they were solved by *R-MIP-I* in very short time (the over all average among these 40 instances is less than 8 seconds). *RBH* achieved near-optimal solutions in these tested instances. The average optimality gap of *RBH* is 1.12% among all instances with two vehicles.

All the instances with three vehicles and 20 nodes were still solved optimally by *E-MIP-I*. However, as the density of the graph and the number of nodes increase, *E-MIP-I* falls short of solving instances optimally within 1 hour. While 9 and 7 out of 10 instances were solved by *E-MIP-I* with 25 and 30 nodes, respectively, *E-MIP-I* could not solve any of the instances with 3 vehicles and 40 nodes optimally, indicating its computational limits. Again, the average run time of *E-MIP-I* over all instances with 3 vehicles (1906.04 seconds) is significantly higher than that of *R-MIP-I* (29.40 seconds). *RBH* gives very good upper bounds for instances with 3 vehicles. The average optimality gap of *RBH* is almost 1.10% on the average for all the instances with 3 vehicles.

In Table 7.2, we also see that none of the instances with 4 vehicles and 11 connected components could be solved to optimality by *E-MIP-I* in the given time limit of 1

hour. While *E-MIP-I* is able to solve instances with at most 3 vehicles and 30 nodes in a time limit of 1 hour, *R-MIP-I* solved all of the instances with 4 vehicles in an average time of 108.57 seconds. *RBH* still obtains very good upper bounds in instances with 4 vehicles. The average optimality gap of *RBH* is 1.11% over all instances with 4 vehicles.

Table 7.2: Results for the First Set of Random Data

K	Instance Number	$\bar{Q}$	V	$\bar{E}$	$\bar{B}$	No. of Solved Instances by E-MIP-I (in 1 hour)	Average E-MIP-I Time (seconds)	Average R-MIP-I Time (seconds)	Average RBH Optimality Gap
1	1 - 10	5	20	51.4	18.2	10	2.53	0.05	0%
	11 - 20	5	25	73.4	17.6	10	13.19	0.10	0%
	21 - 30	5	30	97.6	25	10	41.26	0.19	0%
	31 - 40	5	40	169	55.4	10	168.79	0.27	0%
2	41 - 50	8	20	45.4	19.4	10	88.23	0.36	0.8%
	51 - 60	8	25	71.4	31.6	10	389.17	0.87	1.1%
	61 - 70	8	30	95.6	41.8	9	1152.06	16.13	1.39%
	71 - 80	8	40	172.6	85.8	3	3190.90	13.86	1.22%
3	81 - 90	8.6	20	48.4	23.8	10	900.07	2.02	1.09%
	91 - 100	8	25	67.4	30	9	1101.43	1.42	0.14%
	101 - 110	8.6	30	90.2	35.4	7	2022.68	11.33	1.7%
	111 - 120	11	40	173.4	100.8	0	3600	102.86	1.5%
4	121 - 130	14	20	49	36.6	0	3600	18.58	2.0%
	131 - 140	14	25	77.5	53.2	0	3600	41.18	1.0%
	141 - 150	14	30	106	70.8	0	3600	70.99	0.84%
	151 - 160	14	40	158.2	96.4	0	3600	303.53	0.62%

In order to compare the performance of *RBH* with different number of connected components and vehicles, we generated another set of random networks. The results of these experiments is given in Table 7.3. The first row of Table 7.3 gives the number of connected components. Second and third rows give the average number of edges and blocked edges, respectively, over all of the 10 instances with each specific number of components. While K gives the number of vehicles in the tested instance, the "time" column gives the average spent time spent by solver to apply *RBH* on the 10 instances with the specific number of vehicles and the "gap" column gives the corresponding average percentage gap of *RBH* over all of these instances.

In Table 7.3, we can see that generally it takes more time to solve the same sized instances with more vehicles, but there are several cases where the problem is solved in

less time when more vehicle exists. We also observed that after a point, having more vehicles does not change the solution, especially with small number of components. When the number of connected components increases, on the average the time spent by solver increases. As a general pattern we can say that the number of components is an important indicator of running time. However, *RBH* tends to have smaller gaps when the number of components gets beyond 12. We think that timing conflicts become less observable as the number of connected components get larger.

Table 7.3: Results for the Second Set of Random Data (50 nodes)

Q	8		10		12		14		16	
$\bar{E}$	114.2		114.0		114.4		119.2		109.4	
$\bar{B}$	18.2		25.4		31.2		38.4		37.6	
K	time	gap	time	gap	time	gap	time	gap	time	gap
1	0.122	0.00%	0.134	0.00 %	0.304	0.00%	1.032	0.00%	4.463	0.00%
2	1.016	0.21%	0.784	0.00%	5.385	0.37%	6.715	0.18%	18.796	0.20%
3	1.560	0.52%	2.768	0.67%	41.757	1.29%	41.875	1.05%	35.232	0.00%
4	3.560	0.26%	16.150	2.07%	104.355	2.72%	136.066	0.01%	141.400	0.35%
5	12.564	0.26%	43.282	2.21%	63.366	1.17%	257.794	0.36%	397.281	1.12%
6	5.830	0.26%	11.141	3.25%	160.868	1.25%	192.522	1.01%	262.798	0.14%

7.2 Results of Tested Instances on KPC-ARCP

7.2.1 Results of the Detailed Istanbul Network Instances

Table 7.4 gives the results of the tested instances for the detailed Istanbul road network. For each tested instance, Q is the number of connected components, $|B|$ is the number of blocked edges and K is the number of vehicles. The columns under (1), titled *R-MIP-II Exact*, are related to the solution of *R-MIP-II* by the Gurobi solver. In these two columns, if an instance is solved optimally within the one hour time limit, the run time is given; otherwise, the run time is 3600 seconds and the optimality gap obtained within one hour is reported. Algorithm 3 provides a feasible solution and a lower bound to *R-MIP-II*, namely $LB_{R-MIP-II}$. For each particular instance the run time of Algorithm 3 is given in column (2). The Lagrangian Relaxation method is used to obtain an upper bound on *R-MIP-II* instances and the run

time of the procedure is given in column (3). Column (4) gives the optimality gap for each instance of $R-MIP-II$ calculated by the obtained lower and upper bounds. In column (5), the run time of the SSV algorithm (Algorithm 4) which gives a lower bound and a feasible solution to $KPC-ARCP$ is given. Column (6) presents the best optimality gap obtained for each individual instance of $KPC-ARCP$. The last two columns, P5 and P7 show if we used Proposition 2 and Proposition 7, respectively, in the results for each instance or not. For instance, Proposition 5 is used in run number 1 with four vehicles. It means that in the output of Algorithm 3 one of the vehicles remained unused and consequently, this algorithm gives the optimal solution to $R-MIP-II$. Hence, we did not solve the Lagrangian Relaxation of this run and the run time in column (3) is the same as the one in column (2). An example of using Proposition 7 is in run number 3 with 2 vehicles. It means that since in the solution to Algorithm 4 there exists a vehicle that clears none of the components, the solution to SSV gives the optimal solution to $KPC-ARCP$.

In Table 7.4, as the number of vehicles increases, the $R-MIP-II$ formulation can not be solved exactly within one hour. For instances with four and five vehicles, only one of the instances out of 10 was solved optimally within one hour and for three vehicles only two of the instances were solved in the given time limit.

In run number 2, $LB_{R-MIP-II}$ was found in 3300.48 seconds and 4939.27 seconds with four and five vehicles, respectively. Apart from run number 2, the maximum run time over other instances was 1190.56 seconds in run number 10 with 5 vehicles. The average run time of $LB_{R-MIP-II}$ increases from 22.37 seconds with a single vehicle to 827.47 with five vehicles over all of the 10 runs. The maximum run time of SSV algorithm is 734.99 in run number 9 with 5 vehicles. The average run time of SSV is 69 seconds over all instances with 1 to 5 vehicles. Column (6) shows that 48 instances out of the 50 tested instances were solved optimally. Despite the difficulty of solving even $R-MIP-II$ exactly, our solution method managed to solve $KPC-ARCP$ optimally in all but two instances. In these two instances, the obtained optimality gap for $R-MIP-II$ is not zero as well.

Table 7.4: Results for Detailed Istanbul Network Instances on KPC-ARCP

run number				R-MIP-II (Upper bound on KPC-ARCP)				Lower Bound on KPC-ARCP				P5		P7					
				(1)		(2)		(3)		(4)						(5)		(6)	
				<i>R-MIP-II</i> Exact Gap (%) Time (s)		Lower bound <i>LB_{R-MIP-II}</i> Time (s)		Upper bound Lagrangian Relaxation Time (s)		<i>R-MIP-II</i> Heuristic Gap (%)						SSV Time (s)		KPC-ARCP Gap (%)	
1	9	153	1	0	9.24	17.95	17.95	0	17.78	0	*								
			2	10.00	3600	24.53	173.28	0	23.93	0									
			3	9.09	3600	29.54	178.29	0	26.72	0									
			4	9.09	3600	82.83	82.83	0	48.3	0	*								
			5	9.09	3600	138.19	138.19	0	71.28	0	*								
2	9	139	1	0	4.18	6.03	6.03	0	6.12	0	*								
			2	0	437.28	20.38	22.1	0	15.98	0									
			3	45.45	3600	1663.34	1663	0	74.67	0	*								
			4	41.66	3600	3300.48	3300.48	0	134.86	0	*								
			5	41.66	3600	4939.27	4939.27	0	193.42	0	*								
3	10	150	1	0	0.88	0.42	0.42	0	0.43	0	*								
			2	0	3524.27	26.75	39.62	0.96	0.86	0		*							
			3	0	3545.4	92.6	112.7	0.32	1.29	0		*							
			4	7.88	3600	125.12	148.09	0	1.71	0		*							
			5	8.20	3600	186.51	186.51	0	2.12	0		*							
4	10	161	1	0	4.98	1.9	1.9	0	1.91	0	*								
			2	0	649.45	4.85	16.35	2.17	4.89	0									
			3	2.12	3600	20.59	39.67	2.13	9.67	2.13									
			4	4.16	3600	21.89	48.91	0	11.04	0									
			5	4.16	3600	65.23	65.23	0	29.97	0	*								
5	9	164	1	0	3.76	1.62	1.62	0	1.64	0	*								
			2	0	137.18	3.29	4.7	25.00	4.91	0									
			3	20.00	3600	6.09	8.45	0	8.27	0									
			4	30.00	3600	48.66	48.66	0	22.03	0	*								
			5	40.00	3600	90.35	90.35	0	34.98	0	*								
6	8	164	1	0	3.09	4.48	4.48	0	4.53	0	*								
			2	0	435.27	5.39	6.86	0	5.41	0									
			3	0	72.34	6.15	9.25	0	6.25	0									
			4	0	158.14	7.66	7.66	0	7.22	0	*								
			5	0	537.23	8.12	8.12	0	7.91	0	*								
7	8	157	1	0	6.98	6.38	6.38	0	6.4	0	*								
			2	0	3600	21.66	23.67	0	11.88	0									
			3	1.40	3600	92.77	92.77	0	20.65	0	*								
			4	1.40	3600	161.93	161.93	0	28.83	0	*								
			5	2.81	3600	229.54	229.54	0	37.32	0	*								
8	13	161	1	0	210.12	180.47	180.47	0	179.24	0	*								
			2	8.33	3600	188.46	190.01	0	192.76	0									
			3	15.38	3600	199.93	202.47	0	189.91	0									
			4	52.38	3600	208.06	378.72	0	205.61	0									
			5	113.33	3600	225.81	230.5	0	218.91	0									
9	13	153	1	0	12.18	11.14	11.14	0	10.81	0	*								
			2	5.26	3600	15.94	17.44	5.26	17.54	5.26									
			3	15.00	3600	44.01	61.46	0	22.73	0									
			4	23.80	3600	618.83	618.83	0	385.73	0	*								
			5	31.81	3600	1201.15	1201.15	0	734.99	0	*								
10	13	166	1	0	45.32	43.34	43.34	0	43.34	0	*								
			2	0	3600	51.85	53.79	0	51.72	0									
			3	14.08	3600	562.93	566.01	0	107.02	0									
			4	14.08	3600	904.07	908.15	0	113.65	0									
			5	0	3600	1190.56	1194.66	0	114.74	0									

7.2.2 Results of the Southwestern Istanbul Network Instances

Table 7.5 gives the results of the conducted tests of *KPC-ARCP* on Southwestern road network.

From Table 7.5 we can see that as the number of vehicles and the number of components increase, *R-MIP-II* becomes more difficult to solve. While all the runs with a single vehicle were solved in 20.53 seconds on the average, none of the runs with five vehicles could be solved optimally in one hour. With four vehicles, *R-MIP-*

Table 7.5: Result for Southwestern Istanbul Road Network Instances on KPC-ARCP

				R-MIP-II (Upper bound on KPC-ARCP)				Lower Bound on KPC-ARCP									
				(1)		(2)		(3)		(4)		(5)		(6)			
run number	Q	B	K	<i>R-MIP-II</i> Exact		Lower bound		Upper bound		<i>R-MIP-II</i>		<i>SSV</i>		KPC-ARCP		P5	P7
				Gap (%)	Time (s)	<i>LB_{R-MIP-II}</i> Time (s)	Lagrangian Relaxation Time (s)	Heuristic Gap (%)	Time (s)	Gap (%)							
1	9	175	1	0	4.46	2.55	2.55	0	2.62	0	2.62	0					
			2	0	366.27	4.74	5.8	7.14	5.02	0							
			3	46.66	3600	25.78	27.32	0	10.4	0							
			4	33.33	3600	78.68	78.68	0	23.79	0	*						
			5	13.63	3600	130.68	130.68	0	37.36	0	*						
2	10	165	1	0	0.56	0.34	0.34	0	0.34	0	0.34	0					
			2	0	9.79	2.35	2.35	0	1.82	0	*						
			3	0	461.55	4.13	4.13	0	3.3	0	*						
			4	33.33	3600	6.05	6.05	0	4.77	0	*						
			5	42.85	3600	8.01	8.01	0	6.27	0	*						
3	10	183	1	0	2.13	1.45	1.45	0	1.41	0	1.41	0					
			2	0	15.05	2.3	19.75	0	3.19	0							
			3	0.41	3600	2.91	59.44	0	4.64	0	*						
			4	0	470	4.15	88.99	0	6.28	0	*						
			5	0.82	3600	5.38	5.38	0	7.64	0	*	*					
4	10	170	1	0	1.8	1.1	1.1	0	1.07	0	1.07	0					
			2	0	80.1	1.51	2.49	0	1.52	0							
			3	0	294.59	2.21	9.79	0	2.68	0							
			4	9.09	3600	2.71	5.62	9.09	3.15	9.09							
			5	8.33	3600	3.25	6.99	0	3.57	0							
5	17	182	1	0	0.63	0.42	0.42	0	0.4	0	0.4	0		*			
			2	0	933.08	52.3	53.61	2.59	0.84	0		*					
			3	1.69	3600	119.52	122.28	2.54	1.2	0		*					
			4	2.52	3600	134.67	137.25	2.10	2.78	0		*					
			5	2.07	3600	1378.75	1382.82	3.75	2.51	0		*					
6	10	179	1	0	5.47	3.71	3.71	0	3.8	0	3.8	0					
			2	0	220.77	6.97	8.07	0	7.17	0							
			3	300.00	3600	22.74	22.74	0	20.8	0	*						
			4	325.00	3600	39.36	39.36	0	34.53	0	*						
			5	325.00	3600	55.78	55.78	0	48.84	0	*						
7	11	167	1	0	91.89	25.68	25.68	0	20.54	0	20.54	0					
			2	6.66	3600	24.06	25.2	13.33	25.37	6.66							
			3	18.75	3600	26.23	28.11	25	26.67	6.25							
			4	18.75	3600	27.58	30.66	17.65	28.01	0							
			5	11.76	3600	31.17	34.86	5.26	30.23	0	*						
8	13	166	1	0	9.29	8.37	8.37	0	8.52	0	8.52	0					
			2	1.69	3600	74.04	3600	1.27	13.54	1.70							
			3	1.66	3600	158.27	163.2	4.22	20.07	0							
			4	2.48	3600	253.04	514.49	2.53	110	0	*						
			5	1.22	3600	278.63	323.5	2.06	219.16	0	*						
9	11	169	1	0	86.03	44.78	44.78	0	43.73	0	43.73	0					
			2	9.09	3600	90.02	91.25	30.00	70.55	9.09							
			3	23.07	3600	100.13	140.69	8.33	79.73	8.33							
			4	12.50	3600	127.13	130.4	0	87.61	0							
			5	12.50	3600	657.15	661.46	0	176.68	0	*						
10	11	177	1	0	3.05	4.62	5.54	0	4.62	0	4.62	0					
			2	0	28.31	6.17	16.62	0	6.16	0							
			3	3.70	3600	9.36	15.36	7.41	8.6	0							
			4	14.28	3600	25.46	3600	3.57	29.4	3.57							
			5	0.34	3600	229.83	3600	0	64.8	0							

II was solved optimally within the one hour time limit only for one instance of the 10 tested runs. With two and three vehicles, seven and two instances were solved optimally within the time limit, respectively. These results attest to the difficulty of solving *R-MIP-II* directly by the Gurobi solver.

While *R-MIP-II* can not be solved for larger instances having more vehicles, Algorithm 3 runs in much shorter time. The largest run time of Algorithm 3 is in run number 5, where the number of components is the largest and 5 vehicles exist. The

average Algorithm 3 run time over 10 instances for a single vehicle is 9.30 seconds and this value increases to 277.86 with five vehicles. We show in column (4) that many of the Algorithm 3 solutions are in fact optimal solutions to $R-MIP-II$ by means of Proposition 5.

The average run time of the Lagrangian Relaxation procedure over all of the instances with 1 to 5 vehicles is 307.06 seconds. Among all of the instances, in three of them, the Lagrangian Relaxation procedure stopped by reaching the time limit of one hour.

In column (5) we see that the SSV algorithm was solved in a very short time for each of the instances. The maximum run time of this algorithm was 219.16 seconds in run number 8 with 5 vehicles. The average SSV run time is 8.44 seconds with a single vehicle and it increases to only 59.71 seconds with 5 vehicles.

Finally, we are able to show that an optimal solution to $KPC-ARCP$ was found in 43 instances out of the 50 tested instances in very short times. The average optimality gap over all of the tested instances with 1 to 5 vehicles is 0.89%. In those instances that the optimal solution to $R-MIP-II$ is not found the optimality gap to $KPC-ARCP$ can be positive.

7.2.3 Results of the Simplified Istanbul Network Instances

Table 7.6 gives the results of the conducted tests of $KPC-ARCP$ on the Simplified Istanbul road network.

In Table 7.6, we can see that as the number of vehicles increases for the same instance, on the average, the problem becomes more difficult. While the average run time of solving $R-MIP-II$ formulation exactly with the Gurobi solver over all 10 instances with a single vehicle is 1.51 seconds, the average $R-MIP-II$ run time increases to 1994.30 seconds over all instances. Although the run time of solving $R-MIP-II$ exactly increases significantly with more vehicles, the run times of Algorithms 3 and 4 remain small in each of the instances and these times are less than 9 seconds for each individual instance. In the Lagrangian Relaxation procedure, in general, small

Table 7.6: Results for Simplified Istanbul Network Instances

				R-MIP-II (Upper bound on KPC-ARCP)				Lower Bound on KPC-ARCP				
				(1)		(2)	(3)	(4)	(5)	(6)		
				<i>R-MIP-II</i> Exact		Lower bound	Upper bound	<i>R-MIP-II</i>	SSV	KPC-ARCP		
run number	Q	B	K	Gap (%)	Time (s)	<i>LB_{R-MIP-II}</i> Time (s)	Lagrangian Relaxation Time (s)	Heuristic Gap (%)	Time (s)	Gap (%)	P5	P7
1	13	102	1	0	2.23	1.42	1.42	0	1.46	0		
			2	0	268.59	2.51	2.87	0	2.39	0		
			3	10.52	3600	3.05	55.03	10.53	2.96	5.00		
			4	10.00	3600	3.44	7.9	5.00	3.17	0		
			5	4.76	3600	3.7	9.07	0	4.06	0	*	
2	8	89	1	0	1.45	0.67	0.8	0	0.7	0		
			2	0	23.01	1.56	8.2	20.00	1.19	0		
			3	0	288.08	2.1	4.96	9.09	2.87	0	*	
			4	8.33	3600	2.38	6.19	0	4.51	0	*	
			5	8.33	3600	2.86	2.86	0	6.31	0	*	
3	7	78	1	0	0.66	1.01	1.01	0	1.04	0		
			2	0	15.48	1.14	2.35	0	1.61	0		
			3	0	10.28	1.43	1.43	0	1.67	0	*	*
			4	0	5.97	1.73	1.73	0	2.1	0	*	
			5	0	36.43	1.96	1.96	0	2.57	0	*	
4	7	84	1	0	0.26	0.2	0.2	0	0.2	0		
			2	0	16.59	0.37	4.91	0	0.42	0		
			3	0	120.01	0.57	2.89	0	0.61	0		
			4	0	15.27	0.99	10.28	0	1.34	0		
			5	0	266.19	1.11	1.11	0	1.32	0	*	
5	8	89	1	0	3.62	2.59	2.59	0	2.52	0		
			2	0	2944.16	5.68	5.93	3.33	3.59	1.75		
			3	0	390.21	6.48	6.87	1.64	5.38	0		
			4	1.61	3600	7.4	7.91	0	6.31	0	*	*
			5	1.61	3600	8.78	9.41	0	7.11	0	*	*
6	8	79	1	0	1.36	0.68	0.68	0	0.72	0		
			2	0	29.78	1.38	1.63	0	1.22	0		
			3	0	58.18	1.59	1.99	0	2.42	0		*
			4	0	337.54	1.7	1.7	0	3.5	0	*	
			5	0	956.35	1.83	1.83	0	4.73	0	*	
7	9	90	1	0	1.4	0.89	0.89	0	0.9	0		
			2	0	38.44	1.66	1.95	6.67	1.69	6.67		
			3	5.88	3600	2.6	2.97	0	2.77	0		
			4	11.11	3600	3.82	3.82	0	3.52	0	*	
			5	5.26	3600	5.08	5.08	0	4.3	0		
8	8	83	1	0	2.96	1.04	1.04	0	1.04	0		
			2	0	98.28	2.04	2.33	0	1.99	0		
			3	0	112.12	2.11	2.11	0	2.06	0	*	
			4	0	125.31	2.23	2.23	0	2.22	0	*	
			5	0	129.56	2.32	2.32	0	2.36	0	*	
9	9	97	1	0	0.91	0.85	0.85	0	0.83	0		
			2	0	12.34	1.22	2.26	0	1.22	0		
			3	0	112.76	1.32	4.51	0	1.45	0		
			4	0	537.24	1.52	1.79	0	1.68	0		*
			5	0	554.45	1.77	1.77	0	1.98	0		*
10	9	88	1	0	0.28	0.28	0.28	0	0.27	0		
			2	0	121.89	0.43	14.8	0	0.42	0		
			3	0	981.91	0.92	5.47	0	0.59	0		
			4	1.81	3600	1.51	1.51	0	0.78	0	*	
			5	1.81	3600	1.97	1.97	0	0.92	0	*	

run times of at most 55 seconds are obtained. The largest run times are associated with run number 1 due to larger number of connected components. *SSV* gives an optimal solution to *KPC-ARCP* in all but three instances. We utilize Propositions 5 and 7 to verify optimality in many instances.

In Tables 7.4, 7.5 and 7.6 we can see that the proposed *SSV* algorithm yields an optimal solution to *KPC-ARCP* in most of the instances. The average gap over all tested instances for the detailed road network is 0.15%, for the southwest region it is

0.89%, and for the simplified road network this average gap is 0.14%.

Table 7.7 shows a summary of the results of Istanbul instances presented in Tables 7.4, 7.5 and 7.6. While the exact mathematical model for *KPC-ARCP* falls short of solving even instances of the Simplified Istanbul road network in reasonable time, the developed heuristic algorithm gives an optimal or near-optimal solution in less than a minute in all of the tested instances. Again only in the instances that the optimality gap to *R-MIP-II* is positive, the optimality gap of *KPC-ARCP* is positive.

Table 7.7: Summary of the Results of KPC-ARCP (Istanbul Networks)

Category	K	R-MIP-II (Upper bound on KPC-ARCP)					Lower Bound on KPC-ARCP	
		<i>R-MIP-II</i> Exact		Lower bound <i>LB_{R-MIP-II}</i>	Upper bound Lagrangian Relaxation	<i>R-MIP-II</i> Heuristic	<i>SSV</i>	<i>KPC-ARCP</i>
		Avg. Gap (%)	Avg. Time (s)	Avg. Time (s)	Avg. Time (s)	Avg. Gap (%)	Avg. Time (s)	Avg. Gap (%)
Detailed	1	0	30.07	27.37	27.37	0	27.22	0
	2	2.36	2318.35	36.31	54.78	3.34	32.99	0.53
	3	12.25	3241.77	271.80	293.41	0.24	46.72	0.21
	4	18.45	3255.81	547.95	570.43	0	95.90	0
	5	25.11	3293.72	827.47	828.35	0	144.56	0
Southwest	1	0	20.53	9.30	9.39	0	8.44	0
	2	1.74	1245.34	26.45	382.51	5.43	13.52	1.75
	3	39.55	2955.61	47.13	59.31	4.75	17.81	1.46
	4	45.13	3287.00	69.88	463.15	3.49	32.56	1.27
	5	41.77	3600	277.86	620.95	1.11	59.71	0.00
Simplified	1	0	1.51	0.96	0.98	0	0.97	0
	2	0	356.86	1.80	4.72	3.00	1.57	0
	3	1.64	927.36	2.22	8.82	2.13	2.28	0.50
	4	3.29	1902.13	2.33	4.51	0.50	2.60	0
	5	2.18	1994.30	3.14	3.74	0.00	3.57	0

7.2.4 Result of Random Data Sets for KPC-ARCP

Since we tested the performance of *E-MIP-I* in Section 7.1.4, and this formulation is similar to *E-MIP-II*, we decided to investigate another perspective of the problem and to test the impacts of changing the density of the network and the number of blocked edges on the quality of the obtained solutions.

Table 7.8 gives the results of the tested random instances on *KPC-ARCP*.

The average degree of the nodes in the generated Istanbul networks are 3.95, 4.31 and 4.84 for the Detailed, Southwest and Simplified networks respectively. The average degree of the nodes for random instances with 200 to 350 nodes is 4.22, 4.62, 5.10 and 5.17 on the average over the 10 tested instances. While the density of the networks increases from 200 nodes to 350 nodes, the difficulty of the problem does not increase monotonically. In particular, while both the Southwest network generated in Section 7.2.2 and the second set of random networks has 250 nodes, and the average

Table 7.8: Random Data results

V	E	B	Q	K	R-MIP (Upper bound on KPC-ARCP)					Lower Bound on KPC-ARCP	
					<i>R-MIP-II</i> Exact		Upper bound		<i>R-MIP-II</i> Heuristic Avg. Gap (%)	<i>SSV</i> Avg. Time (s)	<i>KPC-ARCP</i> Avg. Gap (%)
					Avg. Gap (%)	Avg. Time (s)	Lower bound <i>LB_{R-MIP-II}</i> Avg. Time (s)	Lagrangian Relaxation Avg. Time (s)			
200	422.40	12.9	13	1	0	10.01	5.85	5.87	0	4.61	0
				2	0	208.16	6.94	7.58	1.21	7.00	1.31
				3	0.14	578.35	8.65	10.07	2.91	14.11	1.96
				4	1.28	3600	9.10	12.90	1.32	14.60	2.96
				5	0	19.37	9.35	10.73	0	15.85	1.60
250	578.11	14.9	10	1	0	8.76	6.01	6.01	0	5.96	0
				2	2.22	471.11	7.26	8.51	0.48	9.87	0.46
				3	2.91	1249.34	97.13	99.09	1.75	10.99	1.08
				4	0.49	946.43	47.13	50.03	1.69	11.58	1.32
				5	0	9.05	87.41	88.58	0	8.16	0
300	766.37	26.8	10.25	1	0	3.88	3.06	1.31	0	2.76	0
				2	0.64	1383.12	11.61	3.72	5.94	6.33	0.31
				3	2.00	1084.53	15.27	5.89	3.06	8.04	2.28
				4	0.63	2252.94	15.95	8.07	0.92	8.81	0.63
				5	0.08	469.68	16.53	5.26	0.29	9.27	0.10
350	950	12.6	10.5	1	0	5.39	4.41	4.41	0	4.31	0
				2	0	188.03	19.55	25.19	3.09	8.67	0.37
				3	0.12	853.42	27.76	37.34	1.73	10.64	0.83
				4	0.72	2083.92	35.72	43.90	2.17	11.61	1.67
				5	0	29.14	43.24	47.77	1.07	13.03	1.12

density of the random networks is larger than the Southwest network, the Southwest instances are harder in terms of being solved by *R-MIP-II*. For instance, *R-MIP-II* was solved for none of the Southwest instances with 5 vehicles in one hour but all the instances of the random network with 250 nodes were solved in 9.05 seconds on the average. Since the Detailed Istanbul road network has 349 nodes, random instances with 350 nodes are comparable with them in terms of size. The average degree of the nodes of the random instances with 350 nodes is 5.17 and its higher than the average degree of the nodes for the Detailed Istanbul network which is 3.95. The difficulty of the Detailed Istanbul road network is significantly higher than the randomly generated Euclidean networks with 350 nodes. In general it can be seen in Table 7.8 that Euclidean networks become disconnected more easily and by blocking fewer number of edges. Since the average number of connected components in both of these sets are almost similar, it appears that the number of blocked edges can significantly impact the hardness of the problem.

The results of the experiments of the proposed algorithm on the Euclidean random networks which has been tested over 200 distinct topologies with 200, 250, 300 and 350 nodes with 1 to 5 vehicles in one hour time limit verify the good performance of the proposed methods and algorithms in terms of both run times and optimality gaps.

Chapter 8

CONCLUSIONS AND REMARKS

Within the context of the disaster response logistics, we studied two new arc routing problems in which we plan for the logistics of a road clearing operation after a disaster. In these problems, we optimize the routes of K vehicles that traverse edges and open a subset of the blocked ones to reconnect the post-disaster road network in a synchronized way by accounting for necessary waiting times. The first problem aims to reconnect the whole network in the shortest time. We call this problem *The Multi-vehicle Arc Routing for Connectivity Problem* (K-ARCP). The second problem is a prize collecting arc routing problem that aims to collect the most prize by connecting components within specified time limit, T_{max} . This problem is called *Multi-vehicle Prize Collecting Arc Routing for Connectivity Problem* (KPC-ARCP).

These problems have applications in clearing the roads and restoring the connectivity of the road network damaged by a disaster in different situations. *K-ARCP* can be employed when fairness policy is a must; meaning that we have to restore connectivity of all components. On the other hand, *KPC-ARCP* can be deployed in a situation where assigning prizes and prioritizing different segments of a city with regard to specific criteria can be achieved easily. In a real-life situation, we would expect to use *KPC-ARCP* first in a time-limit to connect the most important components (the components with the most prizes) in the immediate post-disaster phase. Then the network will be updated and *KPC-ARCP* can be solved again in a new time horizon, i.e. T_{max} . This can be repeated until all components are connected. Here the time frame would be the several days following the incident. On the other hand, *K-ARCP* can be used for planning the unblocking and repair operations in the recovery phase, over a longer time horizon to ensure that all components get connected.

For *K-ARCP*, we developed an exact mixed integer programming formulation called *E-MIP-I* that solves small sized instances, i.e. up to 30 nodes, 10 connected components and 3 vehicles within an hour. To solve larger instances, we developed a Relaxation-Based Heuristic (*RBH*) that solves a relaxed version of the main problem and implements a feasibility procedure followed by a neighbourhood search algorithm. We proved that the optimality gap of the relaxation is bounded by the number of vehicles. We also proved the feasible solution obtained from (*RBH*) is bounded by K times the optimal value. Additionally, we devised new and problem-specific local improvement moves.

We tested *E-MIP-I* and *RBH* on both randomly generated and Istanbul road network instances. The largest tested Istanbul instance has 349 nodes, 689 edges, 6 connected components and 4 vehicles. *RBH* showed very good performance in tested instances resulting in either an optimal or near-optimal solution in all of the tested instances. We could derive the optimal solution in more than 85% of the tested cases using *RBH* and obtained tight upper bounds in the others.

For *KPC-ARCP*, we developed an exact MIP formulation called *E-MIP-II* using which only small sized instances can be solved. To solve larger instances, we developed a relaxed MIP formulation called *R-MIP-II*. Even *R-MIP-II* can not be solved for large instances, with more than 300 nodes and 3 vehicles in a reasonable time, i.e. less than an hour. In order to obtain an upper bound on *KPC-ARCP*, we solved the Lagrangian Relaxation of *R-MIP-II* and developed a matheuristic that gives a lower bound to *R-MIP-II* (Algorithm 3). Using these bounds on *R-MIP-II*, we can obtain an optimality gap to *R-MIP-II*. Since *R-MIP-II* is a relaxation of *E-MIP-II*, the results also give an upper bound to *KPC-ARCP*. In addition, we developed a matheuristic method to obtain a feasible solution to *KPC-ARCP*. This method is called the Successive Single Vehicle Algorithm (Algorithm 4). In this algorithm, we solve single vehicle problems while updating the obtained prize of the connected components to zero at every iteration.

We tested *R-MIP-II*, the Lagrangian Relaxation of *R-MIP-II*, Algorithm 3 and

Algorithm 4 on both randomly generated Euclidean and Istanbul road network instances. We used three Istanbul road networks and tested 10 instances for each of them with 1 to 5 vehicles. We also tested with random networks having 200, 250, 300 and 350 nodes and 1 to 5 vehicles. The largest tested Istanbul instance has 349 nodes, 689 edges, 13 connected components and 5 vehicles. Our methods to derive an optimality gap on *KPC-ARCP* showed very good performance in tested instances resulting in either an optimal or near-optimal solution in all of the tested instances. We found an optimal solution in more than 90% of the tested instances.

The work in this thesis can be extended to the case where vehicles are heterogeneous with respect to their specialties and each vehicle can be assigned to a suitable subset of blocked roads. For instance, damages on road sections can be categorized into three groups as; light, medium and severe. Similarly, vehicles can be categorized into three groups based on their equipment to fix either of these damage types. The less equipped vehicles are faster but their ability to open blocked edges is limited to only light damages. On the other hand, the more equipped vehicles (the vehicles that are equipped enough to fix severe damages) are slower but capable of fixing any type of damage on road segments.

The case with multiple depots can also be analyzed for *KPC-ARCP*. However, in this case how the connectivity prizes will be determined should be clarified. For instance, the prizes can be collected if a component is connected to any of the depots. In another variation, we can assume that the depot and the facility (supply point) are located in two distinct nodes and the prizes can be collected only if the components are connected to the facility location and not the vehicle depots.

Another variation of these problems which can be considered as combination of them can be studied as well. Earlier we mentioned that in a real-life situation, we would use *KPC-ARCP* first and then *K-ARCP* to restore the connectivity of all the components. However, we can consider a more dynamic approach by considering multiple periods and incurring a penalty of not opening a component in each of the periods. The objective in this case can be defined as maximizing the prize of opened

components minus the penalties from the components that are not opened in that particular period. Another variation of this type of problems can be studied by considering fixing the required percentage of prizes for each period and then minimizing the period lengths. For instance, we aim to collect at least 60% of prizes in the first period in the shortest possible time and a cumulative of 85% in the second period and finally open all the components in the third period. We minimize the length of these periods and then pick the largest among them as the period length. The final goal is to minimize this period length.

We can solve our problems consecutively in multiple periods as new information arrives. With multiple periods, we can use the same solution approach and update the condition of the network at the end of each period, and input it to the problem for the next period. Cases with uncertain opening times or uncertain set of blocked edges can be studied as well. If the opening times are uncertain, a scenario-based stochastic programming or a robust optimization approach can be used to tackle the problem. In such cases, we still need to solve the deterministic problem efficiently as a first step. The problem of choosing the depot prior to the disaster, considering a set of damage scenarios, can also be interesting. The latter case falls into the group of location-routing problems.

BIBLIOGRAPHY

- [Akbari and Salman, 2016] Akbari, V. and Salman, F. S. (2016). Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity, "under review".
- [Akoudad and Jawab, 2013] Akoudad, K. and Jawab, F. (2013). Recent survey on bases routing problems: Cpp, rpp and carp. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications.
- [Aksu and Özdamar, 2014] Aksu, D. T. and Özdamar, L. (2014). A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61:56 – 67.
- [Aráoz et al., 2009] Aráoz, J., Fernández, E., and Meza, O. (2009). Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3):886–896.
- [Aráoz et al., 2006] Aráoz, J., Fernández, E., and Zoltan, C. (2006). Privatized rural postman problems. *Computers and Operations Research*, 33(12):3432–3449.
- [Archetti et al., 2013] Archetti, C., Speranza, M. G., Corberán, Á., Sanchis, J. M., and Plana, I. (2013). The team orienteering arc routing problem. *Transportation Science*, 48(3):442–457.
- [Asaly, 2013] Asaly, A. N. (2013). Logistics planning for restoration of network connectivity after a disaster, thesis no: 397223. Master's thesis, Koc University.
- [Assad et al., 1987] Assad, A. A., Pearn, W.-L., and Golden, B. L. (1987). The

- capacitated chinese postman problem: Lower bounds and solvable cases. *American Journal of Mathematical and Management Sciences*, 7(1-2):63–88.
- [Balas, 1989] Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6):621–636.
- [Benavent et al., 2009] Benavent, E., Corberan, A., Plana, I., and Sanchis, J. M. (2009). Min-max k-vehicles windy rural postman problem. *Networks*, 54(4):216–226.
- [Berktaş et al., 2016] Berktaş, N., Kara, B. Y., and Karaşan, O. E. (2016). Solution methodologies for debris removal in disaster response. *EURO Journal on Computational Optimization*, pages 1–43.
- [Black et al., 2013] Black, D., Eglese, R., and Wøhlk, S. (2013). The time-dependent prize-collecting arc routing problem. *Computers and Operations Research*, 40(2):526–535.
- [Bodin and Kursh, 1978] Bodin, L. D. and Kursh, S. J. (1978). A computer-assisted system for the routing and scheduling of street sweepers. *Operations Research*, 26(4):525–537.
- [Bouliane and Laporte, 1992] Bouliane, J. and Laporte, G. (1992). Locating postal relay boxes using a set covering algorithm. *American Journal of Mathematical and Management Sciences*, 12(1):65–74.
- [Brucker, 1981] Brucker, P. (1981). The chinese postman problem for mixed graphs. In *Graphtheoretic Concepts in Computer Science*, pages 354–366. Springer.
- [Campbell and Langevin, 2000] Campbell, J. F. and Langevin, A. (2000). *Roadway Snow and Ice Control*, pages 389–418. Springer US.

- [Çelik et al., 2015] Çelik, M., Ergun, O., and Kesknocak, P. (2015). The post-disaster debris clearance problem under incomplete information. *Operations Research*, 63(1):65–85.
- [Chang and Nojima, 2001] Chang, S. E. and Nojima, N. (2001). Measuring post-disaster transportation system performance: the 1995 kobe earthquake in comparative perspective. *Transportation Research Part A: Policy and Practice*, 35(6):475–494.
- [Chen and Tzeng, 1999] Chen, Y.-W. and Tzeng, G.-H. (1999). A fuzzy multi-objective model for reconstructing the post-quake road-network by genetic algorithm. *International Journal of Fuzzy Systems*, 1(2):85–95.
- [Christofides et al., 1986a] Christofides, N., Campos, V., Corberán, Á., and Mota, E. (1986a). An algorithm for the rural postman problem on a directed graph. *Mathematical Programming Study*, 26:155–166.
- [Christofides et al., 1986b] Christofides, N., Campos, V., Corberán, A., and Mota, E. (1986b). An algorithm for the rural postman problem on a directed graph. *Mathematical Programming*, 26:155–166.
- [Coleman et al., 2009] Coleman, D. J., Georgiadou, Y., and Labonte, J. (2009). Volunteered geographic information: The nature and motivation of producers. *International Journal of Spatial Data Infrastructures Research*, 4(1):332–358.
- [Corberán et al., 2014] Corberán, Á., Plana, I., and Sanchis, J. (2014). The rural postman problem on directed, mixed, and windy graphs. in *Arc Routing Problems, Methods and Applications* (Á. Corberán and G. Laporte, eds.). *MOS-SIAM Series on Optimization*. Philadelphia, USA.
- [Corberán and Sanchis, 1994] Corberán, A. and Sanchis, J. (1994). A polyhedral

- approach to the rural postman problem. *European Journal of Operational Research*, 79(1):95–114.
- [Duque and Sörensen, 2011] Duque, P. M. and Sörensen, K. (2011). A GRASP meta-heuristic to improve accessibility after a disaster. *OR Spectrum*, 33:525–542.
- [Eiselt et al., 1995a] Eiselt, H. A., Gendreau, M., and Laporte, G. (1995a). Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242.
- [Eiselt et al., 1995b] Eiselt, H. A., Gendreau, M., and Laporte, G. (1995b). Arc routing problems, part ii: The rural postman problem. *Operations Research*, 43(3):399–414.
- [Faturechi and Miller-Hooks, 2015] Faturechi, R. and Miller-Hooks, E. (2015). Measuring the performance of transportation infrastructure systems in disasters: A comprehensive review. *Journal of Infrastructure Systems*, 21(1):04014025 1–15.
- [Feillet et al., 2005] Feillet, D., Dejax, P., and Gendreau, M. (2005). The profitable arc tour problem: solution with a branch-and-price algorithm. *Transportation Science*, 39(4):539–552.
- [Frederickson, 1979] Frederickson, G. N. (1979). Approximation algorithms for some postman problems. *Journal of the ACM (JACM)*, 26(3):538–554.
- [Gavalas et al., 2015] Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., and Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers and Operations Research*, 62:36 – 50.
- [Ghiani and Laporte, 2000] Ghiani, G. and Laporte, G. (2000). A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87(3):467–481.

- [Ghiani and Laporte, 2014] Ghiani, G. and Laporte, G. (2014). The undirected rural postman problem. in *Arc Routing Problems, Methods and Applications* (Á. Corberán and G. Laporte, eds.). *MOS-SIAM Series on Optimization*. Philadelphia, USA.
- [Grötschel and Win, 1992] Grötschel, M. and Win, Z. (1992). A cutting plane algorithm for the windy postman problem. *Mathematical Programming*, 55(1-3):339–358.
- [Hertz et al., 1999] Hertz, A., Laporte, G., and Hugo, P. N. (1999). Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing*, 11(1):53–62.
- [Hertz et al., 2000] Hertz, A., Laporte, G., and Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135.
- [JICA, 2002] JICA (2002). The study on a disaster prevention/mitigation basic plan in istanbul including seismic micronization in the republic of Turkey. Technical report, Japan International Cooperation Agency.
- [Kasaei, 2015] Kasaei, M. (2015). Arc routing problems to restore connectivity of a road network, thesis no: 332214. Master’s thesis, Koc University.
- [Kim et al., 2006] Kim, B.-I., Kim, S., and Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers and Operations Research*, 33(12):3624 – 3642.
- [Lenstra and Kan, 1976] Lenstra, J. K. and Kan, A. (1976). On general routing problems. *Networks*, 6(3):273–280.
- [Letchford, 1997] Letchford, A. N. (1997). New inequalities for the general routing problem. *European Journal of Operational Research*, 96(2):317–322.

- [Liberatore et al., 2014] Liberatore, F., Ortuño, M. T., Tirado, G., Vitoriano, B., and Scaparra, M. P. (2014). A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. *Computers and Operations Research*, 42:3–13.
- [Martinez, 2003] Martinez, F. J. Z. (2003). *Postman Problems on Mixed Graphs*. PhD thesis, University of Waterloo.
- [Minieka, 1979] Minieka, E. (1979). The chinese postman problem for mixed networks. *Management Science*, 25(7):643–648.
- [Orloff, 1974] Orloff, C. S. (1974). A fundamental problem in vehicle routing. *Networks*, 4(1):35–64.
- [Özdamar et al., 2014] Özdamar, L., Aksu, D. T., and Ergünes, B. (2014). Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, 48(4):249 – 262.
- [Papadimitriou, 1976] Papadimitriou, C. H. (1976). On the complexity of edge traversing. *Journal of the ACM (JACM)*, 23(3):544–554.
- [Pearn, 1994] Pearn, W. L. (1994). Solvable cases of the k-person Chinese postman problem. *Operations Research Letters*, 16:241–244.
- [Raghavachari and Veerasamy, 1999] Raghavachari, B. and Veerasamy, J. (1999). Approximation algorithms for the asymmetric postman problem. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 734–741. Society for Industrial and Applied Mathematics.
- [Richard W. Eglese, 1991] Richard W. Eglese, H. M. (1991). Routeing road sweepers in a rural area. *The Journal of the Operational Research Society*, 42(4):281–288.

- [Sahin et al., 2016] Sahin, H., Kara, B. Y., and Karasan, O. (2016). Debris removal during disaster response: A case for Turkey. *Socio-Economic Planning Sciences*, 53:49 – 59.
- [Salazar-Aguilar et al., 2012] Salazar-Aguilar, M. A., Langevin, A., and Laporte, G. (2012). Synchronized arc routing for snow plowing operations. *Computers and Operations Research*, 39(7):1432 – 1440.
- [Salazar-Aguilar et al., 2013] Salazar-Aguilar, M. A., Langevin, A., and Laporte, G. (2013). The synchronized arc and node routing problem: Application to road marking. *Computers and Operations Research*, 40(7):1708 – 1715.
- [Sohn, 2006] Sohn, J. (2006). Evaluating the significance of highway network links under the flood damage: An accessibility approach. *Transportation Research Part A: Policy and Practice*, 40(6):491 – 506.
- [Win, 1987] Win, Z. (1987). *Contributions to Routing Problems*. PhD thesis, University of Augsburg.
- [Wunderlich et al., 1992] Wunderlich, J., Collette, M., Levy, L., and Bodin, L. (1992). Scheduling meter readers for southern california gas company. *Interfaces*, 22(3):22–30.
- [Yan and Shih, 2009] Yan, S. and Shih, Y. L. (2009). Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers and Operations Research*, 36:2049–2065.