Arc Routing Problems to Restore Connectivity of a Road Network

by

Maziar Kasaei Roodsari

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

Master of Science

in

Industrial Engineering

Koç University

July, 2015

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a M.Sc. thesis by

Maziar Kasaei Roodsari

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. Ceyda Oğuz

_____

Asst. Prof. Dilek Günneç

Date: _____

To my family

ABSTRACT

Natural disasters cause destructive effects. Road networks can be damaged or blocked by debris; bridges and viaducts may collapse. This may enforce closing of some road sections and even render some parts of the road network disconnected. In the immediate disaster response phase, to facilitate emergency transportation, the problematic roads should be cleared or repaired and the road clearing team(s) should be dispatched to the affected areas in the most efficient way. We study emergency road restoration with the aim to reconnect a disconnected road network in the shortest time after a disaster. We address three arc routing problems that find the route of a work-troop dispatched to clear blocked roads and achieve connectivity. In the first problem, we minimize the time to reconnect the network while in the second problem, we maximize the total prize from components that are reconnected within a time limit. In the third problem, we reconnect critical supply and demand points. We prove that these problems are NP-hard. For each problem, we develop an MIP formulation. For the first two problems, we propose two heuristic algorithms based on Variable Neighborhood Descent (VND) framework. We compare the performance of the heuristics with MIP solutions on a case of Istanbul and instances adapted from the literature with up to 400 nodes and 600 edges, and analyze the solution characteristics. We achieve near-optimal or optimal solutions much faster than Cplex in most tested instances; thereby justifying the use of the heuristics in the post-disaster stage. We discuss how these problems can be utilized in a post-disaster decision support system and compare solutions obtained by solving the second problem consecutively over a time frame. This analysis shows the benefit of using the second problem in terms of total waiting time of people until they get access.

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my advisor Dr. F. Sibel Salman for her kind and useful comments, remarks and engagement through the learning process of this Master thesis. Without her support, I would not be able to do this work which was my first research.

I would like to thank my thesis committee members: Dr. Ceyda Oğuz and Dr. Dilek Günneç for their time and support by reading my thesis, listening to my work, and giving great feedback on it. I believe that their insights made my work perfect and flawless.

Many people have supported me during my graduate study at Koç University and made this experience wonderful for me. I will be forever grateful to Seifollah Kasaei, Tahere Haji Kermani, Ehsan Sarayloo, Mohammad Nasr, Marwa Ghattousi, Kave Kasaei, Babak Kasaei, Aida Kasaei, my long-time friend and companion Vahid Akbari. My special thanks goes to Anja Tratnjek for her endless support. I would also like to thank all my 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. I also thank Kaan Telciler and Çağan Ürküp for helping me prepare the data for this thesis.

NOMENCLATURE

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

ARCNCP : Arc Routing for Critical Nodes Connectivity Problem

ARCP : Arc Routing for Connectivity Problem

Cplex : An optimization software package

CPU : Central processing unit

D-node : A disaster-affected demand node

FEMA : Federal Emergency Management Agency

GAMS : General Algebraic Modeling System

H1 : Heuristic 1

H2 : Heuristic 2

JICA : Japan International Cooperation Agency

LB : Lower bound

MIP : Mixed Integer Programming

MST : Minimum Spanning Tree

NP-hard : Non-deterministic Polynomial-time hard

PC-ARCP : Prize Collecting Arc Routing for Connectivity Problem

PGA : Peak Ground Acceleration

R-edge : An edge which has to be unblocked in a solution to obtain a connected graph

RPP : Rural Postman Problem

S-node : A critical supply point

SPL : Shortest Path Length

U-edge : An edge which has to be unblocked in a solution

UB : Upper bound

VND : Variable Neighborhood Descent

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Transportation network disruptions have happened after many of the previous natural disasters. These incidents caused the road networks to be disconnected due to having blocked roads, impeding accessibility to hospitals and critical supply locations. Kocaeli Earthquake (1999), Indonesian Tsunami and Earthquake (2004), Haiti Earthquake (2010), the "Triple Disaster" in Japan (2011), and more recently, the Nepal earthquake (2015) best exemplify such situations and remind the authorities of the need to prepare their cities. In Kocaeli Earthquake, some highway sections were severely damaged and many roads were blocked by building debris. After Japan's earthquake, the debris accumulated in the downtown of Kamaishi City caused the community to be isolated from search-and-rescue efforts. Around 76 percent of the highways in the area were closed. In Nepal earthquake, many roads near the epicenter of the earthquake had become impassable, hence isolating many villages for days.

Disaster response efforts vary depending on the nature of a disaster but typically consist of deploying search-and-rescue teams in the field, evacuation of the casualties, transportation of the casualties for emergency medical response, and the shipment of loads of disaster relief supplies. Hence, a connected and functional road network would assist the disaster manager to provide such services effectively and efficiently.

In the immediate disaster response phase, to facilitate emergency transportation, the problematic roads should be either rapidly cleared or repaired; if not possible, bypassed by an alternative way, such as the use of helicopters. Road clearing teams, consisting of heavy machinery (such as bulldozers, lighting vehicles, drainage pump vehicles, and satellite communication vehicles) together with their operators and other

required personnel, should be dispatched to the affected areas in the most efficient way to restore accessibility in the shortest time.

The main purpose of this research is to provide an efficient solution method that helps the disaster managers decide which problematic roads to open and in which order they should be opened so that in the first critical hours or days and the proceeding time frame, people can use the road network to evacuate the region and the emergency relief requirements can arrive at the affected areas.

The first problem we study is called *Arc Routing for Connectivity Problem* (ARCP) which is the problem of optimizing the route of a team by minimizing the completion time of the last unblocking task resulting in a connected network. The problem is to find which blocked edges to open to ensure network connectivity and a walk for the vehicle starting at the depot (that does not need to go back to the depot). 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 this open walk. The time to clear certain road segments may take long depending on the damage level, and victims would be waiting to get connected as the team is working.

The humanitarian response objective of reaching the most number of people in shortest time may be achieved by modeling the road clearing problem with the objective of maximizing the number of people whose connectivity can be restored within an imposed time constraint. Among the alternative solutions, one that minimizes the time at which the connectivity of all chosen components are restored is selected. Priorities can be assigned to the components in the form of prizes with regard to the number of casualties and amount of emergency requirements within the disaster zones, as well as the existence of critical supply locations such as airports, ports, hospitals, relief coordination points and the depots. Within this context, we propose a new *prize collecting arc routing problem*, where the objective is to collect the maximum amount of prizes in the given time limit by connecting the components. We name it *Prize Collecting Arc Routing for Connectivity Problem (PC-ARCP)*. When applied to support disaster response operations, this optimization problem aims to provide

accessibility to as many people as possible in the very short time available after a disaster occurs.

The third problem discussed in this research is called *Arc Routing for Critical Nodes Connectivity Problem* (ARCNCP) and is addressed for the first time. By solving this problem, we ensure the connectivity between the set of supply points and demand points using the shortest paths. Some components may remain disconnected from the rest of the network provided they contain no supply or demand points. We are only concerned with accessibility of the nodes to each other, and relief supply distribution is not considered in this work.

In order to solve each problem exactly, mathematical models are proposed. Moreover, for the first two problems, a VND algorithm is developed to obtain near-optimal solutions in short time. We provide computational tests using RPP instances from the literature and a case of Istanbul city road network under various disaster scenarios and damage levels.

The thesis will continue as follows. First, we give a literature review related to our problems in Chapter 2 followed by Chapter 3 that describes the problems and proves their NP-hardness. Then, the mathematical models to solve each of the problems exactly are presented in Chapter 4. After describing the proposed heuristic algorithms in Chapter 5, we discuss data generation in Chapter 6. We compare the MIP and heuristic solutions for each of the first two problems in Chapter 7, and discuss the use of the problems in post-disaster decision support by analyzing their solutions in Chapter 8. The thesis is closed by the concluding remarks in Chapter 9.

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). Next, some problems focusing on road maintenance and upgrading in disastrous or non-disastrous situations are introduced. We end this chapter by presenting some more similar problems in the literature to our problems.

2.1 Arc Routing Problems

The problems discussed in this thesis fall into the class of *Arc Routing Problems* (ARP) that seek a least-cost traversal of all of the arcs, as in the *Chinese Postman Problem* (CPP) [32]), or a specified subset of the arcs (as in the Rural Postman Problem [59]) of a graph starting and ending at a given node depot. A comprehensive survey on such problems is presented in Dror [29], [46] and more recently, in [25].

The *Chinese Postman Problem* (CPP) is a basic graph theory problem which falls into the class of ARP. CPP is to find the minimum cost closed walk on all arcs of a given graph. This problem is NP-hard on a mixed graph [63] or a windy graph [14], but on the undirected and directed graphs, CPP can be solved by a polynomial time algorithm. A mixed graph has both directed and undirected links and a windy graph is defined on an undirected graph but traversing in each of the two directions of an edge may cost differently. If all nodes of a graph have even degrees, CPP can be solved by a polynomial time algorithm on a windy graph [81]. The windy version of CPP, called the *Windy Postman Problem* (WPP), was first introduced by Minieka [58]. There exists a 3/2 - approximation algorithm for WPP [71]. Moreover, there exists a 4/3 - approximation algorithm for mixed CPP [57]. However, for the mixed and windy

versions of CPP, there are promising exact approaches such as the branch-and-cut algorithms proposed by [44]. In many practical contexts, it is not necessary to traverse all arcs of a graph, hence, CPP arises mainly as a sub-problem.

A version of CPP that considers priorities among different arcs of a graph is called Hierarchical CPP which is NP-hard [30]. In this problem, the arcs are partitioned into clusters, and there is a precedence relation defined on these clusters. As in CPP, the aim is to find the minimum length tour such that each arc is traversed exactly once while higher prioritized arcs are traversed before the lower prioritized ones. There are heuristic methods proposed in the literature for Hierarchical CPP such as [3]. Hierarchical CPP can be solved polynomially if each of the k subgraphs is connected and the order relation is complete [30].

In most arc routing applications, it is not necessary to service all arcs of the graph and they need to be modeled differently. A well-studied version of ARP is the *Rural Postman Problem* (RPP) proposed by Orlof [59]. RPP is the problem of finding the minimum cost cycle traversing only a specified subset of the arcs of a graph (required arcs) at least once. This problem is NP-hard on an undirected or directed graph [53]. Exact algorithms were developed by Christofides et al. [22], Corberan and Sanchis [26], Letchford [54] and Ghiani and Laporte [42]. In [42], they used branch and cut method and could solve to optimality instances having up to 350 vertices. A mathematical programming formulation that differs from the others was introduced by Garfinkel and Webb [40]. By modifying this formulation and using cutting planes, Fernandez et al. [36] obtained tight lower bounds.

Frederickson [38] gave a 3/2-approximation algorithm for the case of metric costs. This algorithm is a variation of Christofides [21] algorithm for the TSP. Pearn and Wu [65] present a similar constructive algorithm. Ghiani et al. [41] propose a constructive heuristic together with a post-optimization. Improvement procedures have been described by Cook et al. [23] and Hertz et al. [47]. The method used in [47] is based on the ideas of 2-opt move for TSP. De Cordoba et al. [27] employed a heuristic based on Monte Carlo principles. A Local Search based on previous works on TSP is presented

by [45].

Some studies use metaheuristics to tackle RPP, such as Perez [66] and Rodrigues and Ferreira [72]. In [66], they develop an algorithm based on Artificial Ant Colonies while in [72], they introduce a Memetic algorithm and solve RPP instances up to 196 nodes and 238 required edges. Another metaheuristic based on Ant Colony Optimization is proposed in Lagana et al. [51] in which they solve instances up to 300 nodes and 146 required edges.

Various types of RPP can be classified from different perspectives. Similar to CPP, undirected, directed, mixed and windy versions exist. 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 [2] provide a recent survey which presents some variations and applications of RPP while a detailed review on previous works can be found in [32].

The main differences of our problems from RPP are as follows: We search for an open walk starting from a node specified as the depot, while in RPP, a closed walk is desired without any starting or ending points. The required edges in RPP are predetermined; however in our problems, the set of the edges that have to be unblocked is not specified and determining it is a crucial decision. Also, the elapsed time for traversal and serving of an edge in RPP can be assumed to be the same, but in our case, the first traversal of a blocked edge involves unblocking it and charges the work-troop more cost/time. Moreover, the edge is ready for further use only after being unblocked and traversing a blocked edge before unblocking it, is infeasible.

Another type of ARP which resembles RPP is the *Stacker Crane Problem* (SCP). It is defined on a mixed graph $G = (V, A \cup E)$, where A is a set of directed arcs, and E is a set of undirected edges. The problem is to determine a least cost closed walk including each arc of A at least once. There exists a 9/5 - approximation algorithm for this NP-hard problem [38].

The *Capacitated Arc Routing Problem* (CARP) is to specify a minimum cost traversal of all arcs with positive demand by the fleet of vehicles so that the total

served demand by each vehicle does not exceed its capacity. Each arc has a nonnegative demand and must be serviced by exactly one vehicle to fulfill the demand. CARP is NP-hard [43] and a branch-and-cut algorithm is developed by [11] for this problem.

Variations of CARP have been studied in the literature one of which is the *Capacitated Chinese Postman Problem* (CCPP). In CCPP, all arcs have positive demand and have to be serviced exactly once. There exists a $(3/2-\varepsilon)$ - approximation algorithm for this problem [43]. If the arcs have the same demand and vehicles have the same capacity on a linear graph [15], or if the arc demands are identical and the graph is cyclic [9], there exists a polynomial time algorithm for CCPP. Another variant of CARP is the one that has time-dependent service costs [77]. In this problem, there is a subset of arcs that must be serviced at a cost that is dependent on the time of beginning of service. This cost is a piecewise linear function of time. The objective is to minimize the sum of service and traversal times.

2.2 Road Maintenance Problems

There have been many articles published in the past 20 years on disaster management. In [4], which is a recent review paper in this context, the authors focus on the response phase. They claim that the emergency repair of the damages on a road network is the least popular subject. More recently in [61], the authors provide a comprehensive survey on the problems arising in this context.

Feng and Wang [35] consider the highway emergency rehabilitation problem within the critical 72-hour-period after an earthquake. They develop a multi-objective scheduling model to solve this problem. Their objectives are maximizing the length of accessible roads, maximizing the number of life savings and minimizing risks of working in sensitive areas. The main risk of emergency rehabilitation is working in land sliding or falling rock areas. Manpower, machines and vehicles are limited but the budget is not constrained and the decision is how to dispatch the manpower, vehicles and machines.

Fiedrich et al. [37] also consider the initial search-and-rescue (SAR) period after a

disaster. In their problem, they have limited resources that should be assigned to the affected areas with the objective of minimizing the total number of fatalities. They divide the areas into some clusters with different priorities. A dynamic optimization model is introduced because of the parameters that change in time like survival rate that decreases in time and the probability of secondary disasters. They use the expected value of fatalities caused by different causes of death.

Futura et al. [39] address the problem of post-disaster restoration of life-line systems in an uncertain environment. The decisions include assigning groups to disaster regions and finding the best restoration order. Sources of uncertainty are liquefaction, delay in moves, being unable to restore a life-line and increase of damages because of aftershocks. They improved an existing Genetic Algorithm for this problem but still they could not have robustness in all solutions.

The problem defined in [80] is a network design problem in the disaster context. Their objective is to make the connectivity between origin-destination pairs and cover as much population as they can by selecting critical routes to retrofit in preparation for earthquake response, while minimizing the routing costs. An edge can be used in a route only if it is retrofitted. Cost of retrofitting cannot exceed the budget. They solve this model with branch-and-cut module of CPLEX on a case study.

More recently, Liberatore et al. [55] 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 [20], 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 [18] 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.

Moreover, several studies have modeled upgrading the road network or improving accessibility after a disaster. They focus on selection of road segments that need to be upgraded or repaired without considering the routing aspect. One such study is by [31]. In their case, there is a budget constraint and there are a number of roads that need to be repaired while minimizing the weighted sum of time to travel from each rural town to its closest regional center. Depending on the importance of the rural towns, they assign weights to them. Campbell et al. [16] use heuristic methods to solve the problem of deciding on the number of edges to be upgraded before a disaster happens. Their objective is to minimize the maximum travel time between any source-terminal pair.

Some road upgrading studies have focused on non-disastrous situations. One such study is addressed in [75] for which they consider a disconnected undirected graph in a rural region. The objective is to optimize the weighted combination of traffic flow volume served by upgraded roads and weighted traveled distance associated with

those traffic volumes. The problem is subject to a fixed budget for upgrading the roads while ensuring the connectivity of the graph. They propose a metaheuristic to solve this problem. Another work on upgrading arcs has been done in [28] which contains network design and routing aspects as well. The problem is a variation of the classical Minimum Cost Flow Problem. The unit flow cost on each arc can be lowered by investing money and upgrading it. It is defined on a directed graph with capacity and costs associated with each arc. The goal is minimizing the total cost of routing the data on the given graph and upgrading the arcs with regard to an upgrade budget. They proposed a polynomial time approximation algorithm to attack this problem.

Snow removal is another application of the road maintenance problems. There is a four part survey by Perrier et al. [67–70] in which they consider the different decision making problems of winter road maintenance operations including spreading of chemicals and abrasives and snow plowing or disposal. The last two parts of this survey provides solution methods on vehicle routing, depot location and fleet sizing problems. Different types of cost are assigned to the roads: servicing, deadheading a serviced road, deadheading a non-serviced road. The objective is minimizing the total length traveled.

2.3 Other Related Problems

Similar to our work, some recent works have considered emergency road repair problems in terms of scheduling and selecting the order of unblocking or repairing the roads such as Stilp et al. [76] in which they formulate a mixed-integer programming model and propose efficient heuristics to solve the debris management problem in multiple periods. Yan and Shih [82] proposed a time-space network model to find optimal routes for emergency road repairs. Later, Yan and Shih [83], addressed emergency repair and relief distribution simultaneously. There are supply points from which relief distribution is started. The demand points have to receive a minimum required amount of relief items. Relief distribution cannot continue through an arc if that arc is not repaired. A vehicle scheduling model is proposed, minimizing the time of

repairing all damaged points and relief distribution. The *Multi-Vehicle Arc Routing for Connectivity Problem* (K-ARCP) has been addressed recently by Akbari and Salman [1] who provided a math-heuristic for the problem minimizing the maximum tour length of the routes. They use a relaxed version of the model described in [50] and make it feasible with some heuristic moves. However, in our research, we focus on the single vehicle version of this problem namely ARCP, and propose heuristics to solve the problem efficiently, and in addition, we propose the PC-ARCP and ARCNCP for the first time. A similar multi-vehicle problem is also studied in [74] considering the same objective of minimizing the makespan as in [1] and [50]. They propose a mathematical model and heuristic to solve the problem for instances up to 300 vertices and 25 vehicles. Most recently, Aksu and Ozdamar [78] considered the same problem but with a different objective of maximizing the total weighted earliness of all paths' restoration times subject to a time limit. The proposed dynamic path-based model decides on the order of blocked arcs to be restored. In another similar work, Ozdamar et al. [60] propose a rule-based heuristic that decides on the order of blocked arcs to be restored for a limited number of equipments. They also develop a mathematical model that maximizes network accessibility during the debris removal operations and minimizes makespan but fails to solve the problem due to its recursive nature. Their accessibility measure is based on sum of the lengths of shortest paths between all pairs of nodes in the network. Sahin et al. [73] prioritize different affected regions of the network and aim at visiting some critical regions as quickly as possible and if needed, the solution may include unblocking some blocked roads. They propose a multiperiod MIP model to solve the problem.

There are some online/stochastic network problems having similar objectives as ARCP, PC-ARCP or ARCNCP. The *Canadian Traveler Problem* (CTP) is the problem of connecting two nodes using the shortest path on an uncertain network where some of the links might get blocked suddenly and the traveler will know it only when he is in an adjacent node. It was first defined in [64] and they showed the #P-hardness. The foremost difference of the CTP from the problems defined in this thesis is that

CTP is an online problem. Hence, the choice of required edges that is a part of our problems, in CTP is meaningless, since at any time there can be a new blockage. The offline version of CTP differs from our problems in the sense that it reduces to the simple Shortest Path Problem from node s to node t which is only a sub-problem in our problems. The assumption in CTP that says G is connected even when the blocked edges are removed implies a difference to our problems in which there has to be a disconnected graph after omitting the blocked edges. Also, in *Stochastic Debris Clearance Problem* (SDCP) defined by Celik et al. [17], uncertainty (edge lengths) is revealed only when the agent reaches the edge. SDCP is defined over a fixed planning horizon which they divide into discrete time periods. The main decision in SDCP is to determine a sequence of roads to clear in each period such that the benefit by satisfying relief demand is maximized. In [17], they develop a partially observable Markov decision process model and a heuristic. Our problems are similar to their problem in the sense that we both propose quantitative tools to manage the post-disaster road clearing operations with similar decisions, however, our objectives are different. Moreover, their problem is dynamic and they propose a heuristic policy for large scale problems.

In the literature, there are a couple of related Arc Routing Problems to PC-ARCP but they are not concerned with connectivity. For instance, the *Prize Collecting Rural Postman Problem* (PCRPP) that was first defined in [7] with the name *Privatized Rural Postman Problem* (PRPP). A tour starting and ending at the depot, maximizing the difference between the profits of serving required arcs and total traversing costs is found. The ILP model for this problem has an exponential number of constraints and in [6], they propose an algorithm that relaxes the model and iteratively solves it with smaller number of constraints and a heuristic to generate feasible solutions. The *Profitable Arc Tour Problem* (PATP) imposes restrictions on the maximum length of the tour and the number of times a benefit (prize) can be obtained [34]. They could solve instances up to 65 vertices using the branch-and-price algorithm. Variants of this problem; for instance, the *Clustered Prize Collecting Arc Routing Problem* [5] and

the windy version of it [24] have also been studied. More recently, Black et al. [12] studied the *Time Dependent Prize Collecting Arc Routing Problem* (TD-PARP) that has a tour length limit and time-dependent travel times. They solved the problem with up to 600 prize arcs using Variable Neighborhood Search and Tabu Search.

With similar objective to PC-ARCP but different nature, there are analogous node routing problems such as the *Prize Collecting Traveling Salesman Problem* (PCTSP) [10], 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. There is yet another more similar node routing problem to PC-ARCP called the *Orienteering Problem* (OP). The objective is to obtain as much score possible as the tour length limit allows. Many versions of OP are presented [79]. An extension of OP is the *Team Orienteering Problem* (TOP) proposed by [19]. In TOP, they specify start and end points for all team members. Given a fixed amount of time for each of the members, the goal is to determine M paths from the start point to the end point through a subset of nodes in order to maximize the total score. They propose fast heuristics and test it on 353 instances. More recently, Archetti et al. [8] address an arc routing version of TOP, *Team Orienteering Arc Routing Problem* (TOARP). TOARP finds tours for a number of vehicles each having a tour length constraint while collecting the prizes from the arcs of a graph. In [8], they propose a metaheuristic that uses an ILP formulation within a Tabu Search framework and it finds an optimal solution in almost all benchmark instances they used having up to 27 nodes and 296 arcs. TOARP differs from PC-ARCP in the sense that in PC-ARCP, the tour can end in any desired node and the prizes in PC-ARCP are related to each component of the network. Making the component connected allows us to collect the prize. Moreover, the shortest path calculations are not as easy as in TOARP, since after unblocking each blocked edge, there can be potentially shorter paths to other nodes.

This study focuses on ensuring functionality of transportation networks such as highway and road networks that may be affected by a natural disaster. To the best of

our knowledge, the restoration of the roads after a disaster to ensure connectivity of a network with arc routing and prize collecting perspectives has not been addressed in the literature. We define two new network optimization problems considering road restoration. One has the objective of providing connectivity of as many components of a disconnected network as possible within a time limit and the other one provides connectivity between supply and demand points using shortest paths. We also address another similar arc routing problem to ensure connectivity of the whole disconnected network. These problems combine arc routing, network design and scheduling concepts. We propose mathematical models to solve the problems exactly and algorithms that take advantage of the problem structure to find good feasible solutions in short time. Proving the NP-hardness of each problem is another contribution of our work. Moreover, generating and using real-life data from Istanbul road network, and adapting some other instances from literature, we empirically tested the performance of our algorithms. It is worth mentioning that solving the multi-vehicle problems is computationally challenging due to the coordination of vehicles. Since the road clearance problem should be solved after a disaster, it is critical to find a solution quickly. Spreading the work-troops around a city and locating each single work-troop within every small region of the city, and dispatching it immediately after the disaster is both practical and beneficial. This approach carries less risk of having many work-troops unfunctional in a possible isolation of the depot. Therefore, partitioning the network and solving single vehicle problems is practical and a promising time-efficient solution method to the multi-vehicle case of this problem that lacks efficient solution methods.

Chapter 3

PROBLEM DESCRIPTION AND HARDNESS

3.1 Arc Routing for Connectivity Problem (ARCP)

Let $G = (V, E)$ be an undirected, simple graph, where V is the vertex set, E is the edge set, $B \subseteq E$ is the set of blocked edges such that the graph $G^- = (V, E \setminus B)$ is disconnected. Q denotes the set of components of the graph G^- where we assume $|Q| \geq 2$. V_q denotes the set of vertices in component $q \in Q$ and q_d is the index of the component containing the depot. c_{ij} is the traversal time on an open edge $(i, j) \in E$. When a blocked edge $(i, j) \in B$ is traversed for the first time, it is unblocked with additional time of unblocking $b_{ij} \geq 0$ and becomes open for later use. Thus, traversal time on an edge which was not blocked initially or a blocked edge after it has been unblocked is equal to c_{ij}. The objective is to minimize the tour length while restoring the connectivity of the whole network.

Definition 3.1.1. *For any subset $S \subset V$, we define cut-set $\delta(S)$ as the set of all edges $(i, j) \in E$ such that $i \in S$, $j \in V \setminus S$ and $\lambda(S)$ as the set of all edges $(i, j) \in E$ such that $i, j \in S$.*

Proposition 3.1.1. *In an optimal solution to ARCP, unblocking any blocked edge inside of a component i.e. $(i, j) \in B \cap \lambda(V_q), \forall q \in Q$, may be necessary.*

Proof. Calculate the shortest path length from i to j, $SPL(i, j)$ in G. In the case that $SPL(i, j) \geq b_{ij} + c_{ij}$, unblocking edge (i, j) will end in less or the same total cost. Therefore, (i, j) will be unblocked in an optimal solution. □

Definition 3.1.2. *Let W be the walk of a feasible solution to ARCP:*

1. *Let U be the set of all blocked edges that are to be unblocked in W. We refer to the edges in U as U-edges.*

2. *Let R be the set of edges in W that are to be unblocked and required to obtain a connected network. This set is not pre-specified and $R \subset \delta(V_q) \cup U, \forall q \in Q$. We refer to the edges in R as R-edges.*

3. *Let O be the order in which the blocked edges in R are to be unblocked. In other words, O is an ordered list of the set R.*

Proposition 3.1.2. *In any optimal solution to ARCP, the following property holds. Let $O : e_1 - e_2 - ... - e_{|Q|-1}$ be the order in which the R-edges are to be unblocked. Then, the walk includes the shortest paths to connect the consecutive edges in this ordered list, and to connect the depot to e_1.*

Proof. Dominance Relation 3 in [42] which is for RPP states that between any two connected components of the R-induced graph, among the edges that have exactly one end-node in one component and one end-node in the other component, only the one with minimum cost will be used to connect those two components. When the set U is known, ARCP reduces to RPP. If we assume each of the U-edges in ARCP to be one component in the R-induced graph in RPP, then we can conclude that shortest paths are used to traverse in between the U-edges in ARCP. This holds for the set U in every optimal solution to ARCP. □

Next, we prove that ARCP is NP-hard:

Theorem 3.1.3. *ARCP is NP-hard. In addition, the special case of ARCP in which traversal costs/times are zero, that is selecting the edges to be unblocked and sequencing the unblocking tasks, is also NP-hard.*

Proof. (By reduction from the Steiner Tree Problem): In the *Steiner Tree Problem* (STP), we are given a graph $G_S = (V_S, E_S)$, a positive weight w_{ij} for each edge (i, j) and a subset of nodes, R, that should be connected with minimum total weight. For a given STP instance, we construct an ARCP instance on a graph $G_A = (V_A, E_A)$ as follows:

All edges in E_S are included in E_A as blocked edges. We add a dummy node s to G_A and connect every node in V_S to s by an open edge with a high traversal cost/time. As a result, all nodes in V_S are in one component. Arbitrarily, we let a node $d \in R$ as the depot node. For each node $i \in R$, we add a new node i' and a blocked edge (i, i') to G_A with unblocking time $b_{ii'} = 1/|V|$. All traversal times in G_A are zero and unblocking time of each edge $(i, j) \in E_A$ is set to its weight, i.e., $b_{ij} = w_{ij}$. Note that G_A consists of $|R| + 1$ components. Take a feasible solution to ARCP defined on G_A. It should connect every node $i \in R$ since it has to unblock (i, i'). For each edge that is unblocked to reach the required nodes, we incur w_{ij} and if those edges are traversed multiple times, we do not incur any additional cost/time.

If we consider the union of edges in the walk, say T, it gives us a tree with total weight: $\sum_{(i,j)\in T} w_{ij} + |R|/|V|$. Then T is a feasible solution to STP problem with total weight: $\sum_{(i,j)\in T} w_{ij}$.

Now suppose there is a feasible solution to STP with edges T. Then, starting from d, we traverse all the edges in T in a walk such that the first time we reach a node $i \in R$, we add i, i', i to the walk. Clearly, we connect all components of G_A by this walk with total cost/time $\sum_{(i,j)\in T} w_{ij} + |R|/|V|$. Hence, it provides a feasible solution to ARCP.

Moreover, any optimal solution to STP on G_S gives us an optimal solution to ARCP on G_A as described above since all the edges (i, i') must be unblocked. With the same reasoning an optimal solution to ARCP on G_A gives an optimal solution to STP on G_S. □

3.2 Prize Collecting Arc Routing for Connectivity Problem (PC-ARCP)

Similar to ARCP, we model the road network as an undirected connected graph $G = (V, E)$ and costs/times b and c apply to all of the edges of the graph G. The edges that are not blocked form a graph $G^- = (V, E \setminus B)$ with $|Q|$ components where Q is the set of components. Let V_q be the node set of component $q \in Q$. The non-negative prize p_q is associated with each component q.

PC-ARCP aims at finding a walk in G collecting the maximum total prize while limiting the length (time) of the walk by the given time limit t_{max}. We define the objective of PC-ARCP to include the length of the walk with a small coefficient to break ties among alternative solutions.

The definitions of U-edges, R-edges and the ordered list O in a feasible walk W for ARCP (Definition 3.1.2) apply to PC-ARCP as well. Likewise, both Propositions 3.1.1 and 3.1.2 are valid for PC-ARCP.

Theorem 3.2.1. *PC-ARCP is NP-hard.*

Proof. According to the following lemma, ARCP is a special case of PC-ARCP. Hence, PC-ARCP should be NP-hard since ARCP is NP-hard. □

Lemma 3.2.2. *PC-ARCP generalizes ARCP.*

Proof. Assume a PC-ARCP instance that we design by specifying the input t_{max} as a large enough number (enough to collect all prizes). All prizes will be collected for sure and among the solutions that have collected all prizes, the ones with minimum tour length will be chosen. □

3.3 Arc Routing for Critical Nodes Connectivity Problem (ARCNCP)

Similar to ARCP and PC-ARCP, we model the road network as an undirected connected graph $G = (V, E)$ and costs/times b and c apply to all of the edges of the graph G. The edges that are not blocked form a graph $G^- = (V, E \setminus B)$ with $|Q|$ components where Q is the set of components. Let S be the set of pre-specified critical supply locations referred to as S-nodes and D the set of disaster-struck locations (demand nodes) referred to as D-nodes. Note that the definition of U-edges for ARCP (Definition 3.1.2) and both Propositions 3.1.1 and 3.1.2 are valid for ARCNCP as well.

The objective of this problem is to connect the S-nodes to the D-nodes in shortest time. The time at which the last critical node (S-node or D-node) is connected, is minimized. In real life, S-nodes can be considered as hospitals, fire stations, airports,

harbors, disaster coordination centers or other supply locations. The problem is to find the set of U-edges to be unblocked and a route for the vehicle, starting at the depot node, unblocking some blocked edges such that all S-nodes and D-nodes are connected to each other for subsequent supply distribution from S-nodes to D-nodes. The vehicle does not need to go back to the depot.

Since there are possible blockages inside of the components, making connectivity between the nodes themselves makes more sense than making connectivity between the components. Finding a direct walk between S-nodes and D-nodes and unblocking the U-edges (the required blocked edges inside of or between the components), we ensure existence of an open path between S-nodes and D-nodes for the consequent relief distribution efforts. Note that we just consider connectivity of the critical nodes, and transportation of relief supply is not our concern in this work.

Theorem 3.3.1. *ARCNCP is NP-hard.*

Proof. Let us imagine an ARCNCP instance with three nodes that has two connected components such that one of them contains the depot node and a D-node, and the other component contains a single S-node. The only blocked edge lies in the cut-set of the two components, linking the S-node to the D-node. Then, ARCNCP becomes to connect these two components in shortest time. In other words, it transforms into ARCP which is NP-hard. Thus, we can conclude that ARCNCP is NP-hard, too. □

Chapter 4

MATHEMATICAL MODELS

4.1 MIP Model for ARCP

In this section, we present an MIP model to solve ARCP exactly. We first provide a dominance property.

Proposition 4.1.1. *There exists an optimal solution such that an edge will not be traversed in the same direction more than once.*

Proof. A similar dominance relation was proven in [22] for the RPP. When the edges to be unblocked, U, are selected, ARCP reduces to RPP except that the walk in ARCP is open and the first traversal of a blocked edge incurs the additional cost b_{ij}. Having an open walk will not cause extra traversals compared to a closed walk. Note that the total unblocking cost/time of all selected blocked edges, U, should be added to the objective function, which is a constant when the set U is known. Therefore, only the traversal costs c_{ij} will be used in ARCP. Hence, the cost structure is the same as in RPP. As a result, the dominance property holds. □

We add the dominance relation imposing an edge to be traversed at most once in each direction according to Proposition 4.1.1. This dominance relation makes the solution space smaller by cutting off some feasible solutions. We write our formulation to find only solutions that satisfy this property.

The connectivity of the walk is ensured by sending flow. As a result, no sub-tour or subgraph of components will be in the optimal solution. The depot node has supply equal to the number of nodes visited by the vehicle. The net flow into a node is set to the number of times the node is visited. The flow variables, F_{ij} and F_{ji}, are defined as real numbers; however, they take integer values. In the formulation presented below,

since the walk is open, we define a sink node with index $n+1$ to end the walk.

Decision Variables:

X_{ij} : Binary variable which equals 1 if edge (i,j) is traversed in the direction from node i to node j, and 0 otherwise, $\forall (i,j) \in E$. A corresponding variable X_{ji} is defined to indicate that edge (i,j) is traversed from node j to i.

Z_{ij} : Binary variable which equals 1 if blocked edge (i,j) is unblocked, and 0 otherwise, $\forall (i,j) \in B$.

Y_i : Number of times node i is visited, $\forall i \in V$.

F_{ij} : Flow amount on edge (i,j) in the direction from node i to node $j, \forall (i,j) \in E$. Similarly, F_{ji} is defined to represent the flow in the direction from node j to i on edge (i,j).

MIP Model for ARCP:

$$\text{Minimize} \sum_{(i,j)\in E} c_{ij}(X_{ij} + X_{ji}) + \sum_{(i,j)\in B} b_{ij} Z_{ij}$$

subject to:

$$\sum_{\substack{j\in V:\\(j,i)\in E}} X_{ji} - \sum_{\substack{j\in V:\\(i,j)\in E}} X_{ij} = 0 \qquad \forall i \in V \setminus \{d\} \tag{4.1}$$

$$\sum_{\substack{i\in V:\\(d,i)\in E}} X_{di} - \sum_{\substack{i\in V:\\(i,d)\in E}} X_{id} = 1 \tag{4.2}$$

$$\sum_{j\in V} X_{(n+1)j} = 0 \tag{4.3}$$

$$\sum_{i \in V} X_{i(n+1)} = 1 \tag{4.4}$$

$$\sum_{i \in V_q} Y_i \geq 1 \qquad \forall q \in Q \setminus \{q_d\} \tag{4.5}$$

$$X_{ij} + X_{ji} \geq Z_{ij} \qquad \forall (i,j) \in B, i < j \tag{4.6}$$

$$X_{ij} \leq Z_{ij} \qquad \forall (i,j) \in B, i < j \tag{4.7}$$

$$X_{ji} \leq Z_{ij} \qquad \forall (i,j) \in B, i < j \tag{4.8}$$

$$\sum_{\substack{j \in V: \\ (j,i) \in E}} X_{ji} = Y_i \qquad \forall i \in V \cup \{n+1\} \setminus \{d\} \tag{4.9}$$

$$\sum_{\substack{j \in V: \\ (j,i) \in E}} F_{ji} - \sum_{\substack{j \in V: \\ (i,j) \in E}} F_{ij} = Y_i \qquad \forall i \in V \setminus \{d\} \tag{4.10}$$

$$\sum_{\substack{j \in V: \\ (d,j) \in E}} F_{dj} - \sum_{\substack{j \in V: \\ (j,d) \in E}} F_{jd} = \sum_{i \in V \cup (n+1) \setminus \{d\}} Y_i \tag{4.11}$$

$$\sum_{i \in V} F_{i(n+1)} = 1 \tag{4.12}$$

$$\sum_{j \in V} F_{(n+1)j} = 0 \tag{4.13}$$

$$F_{ij} \leq (2|V| - 1) X_{ij} \qquad \forall (i,j) \in E \tag{4.14}$$

$$F_{ji} \leq (2|V| - 1) X_{ji} \qquad \forall (i,j) \in E \tag{4.15}$$

$$F_{ij} \geq X_{ij} \qquad \forall (i,j) \in E \tag{4.16}$$

$$F_{ji} \geq X_{ji} \qquad \forall (i,j) \in E \tag{4.17}$$

$$X_{ij}, X_{ji} \in \{0,1\} \qquad \forall (i,j) \in E \tag{4.18}$$

$$F_{ij}, F_{ji} \in \mathbb{R}^+ \cup \{0\} \qquad \forall (i,j) \in E \tag{4.19}$$

$$Z_{ij} \in \{0,1\} \qquad \forall (i,j) \in B \tag{4.20}$$

$$Y_i \in \mathbb{Z}^+ \cup \{0\} \qquad \forall i \in V \cup \{n+1\} \setminus \{d\} \tag{4.21}$$

The objective function is to minimize the summation of the total unblocking time and traversal time. The trade-off between these two parts of the objective function gives us a heuristic idea which will be described in Chapter 5.

Constraints (4.1)-(4.4) are the vehicle balance equations. To ensure connectivity of

the graph, instead of using cut-set constraints for every subset of connected components that will be exponential in terms of the number of components, we made use of the number of times node i is visited in different components. In Constraints (4.5), visiting at least one of the nodes in each of the components is imposed to ensure connectivity of the graph. Constraints (4.6) ensure that an unblocked edge should be traversed. On the other hand, in Constraints (4.7) and (4.8), we prevent our walk from traversing a blocked edge if it is not unblocked.

Constraints (4.9) define the total number of times node i is visited in the walk. The flow balance equations are Constraints (4.10) to (4.13). In Constraints (4.11), the net flow into the depot node should equal the total number of visits to all other nodes except the depot. For the other nodes, the net flow equals the number of visits to the corresponding node as seen in Constraints (4.10). In other words, a vehicle leaves one unit of flow each time it visits a node.

Constraints (4.14) and (4.15) do not allow flow on an edge unless it is traversed. It can be shown that no more than $2|V| - 1$ units of flow is needed on an edge knowing that one unit of flow is left at a node each time it is visited.

Constraints (4.16) and (4.17) show that if an edge is traversed, then there must be a positive amount of flow passing through it. Finally, Constraints (4.18), (4.19), (4.20) and (4.21) are variable definition constraints. Although in this problem we assume the graph to be undirected, we define the X_{ij} and X_{ji} distinctly, since the walk has direction.

Proposition 4.1.2. *In an optimal solution to ARCP, at least $|Q| - 1$ blocked edges should be unblocked.*

Proof. Unblocking $|Q| - 1$ blocked edges that are in the cut-sets of the connected components is needed to reconnect G^-. □

Remark 4.1.3. *Note that there are cases where we need to unblock more blocked edges, one of which was discussed in Proposition 3.1.1 (unblocking blocked edges inside of the components). The other case happens where we need to unblock more than $|Q| - 1$*

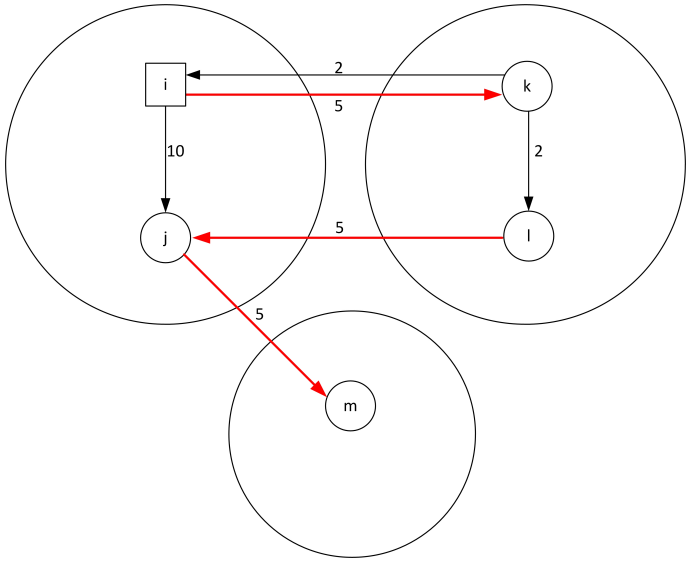


Figure 4.1: An example of unblocking more than $|Q| - 1$ blocked edges from the cut-sets

edges in the cut-sets of the components. The latter case happens simply because of the same reason as the previous case, being more cost/time efficient. For example, in Figure 4.1 we see that the path (i, k, l, j, m) *costs less than* (i, k, i, j, m)*. Hence, edges* (i, k)*,* (l, j) *and* (j, m) *are chosen to be unblocked. In this example, all traversal costs of edges* (c) *equal 2 and unblocking costs* (b) *equal 3 except for edge* (i, j) *where* $b_{ij} = 8$*.*

As a result of Proposition 4.1.2, we add the valid inequality $\sum_{(i,j)\in B} Z_{ij} \geq |Q| - 1$ to the MIP Model for ARCP.

4.2 MIP Model for PC-ARCP

In this section, a maximization bi-criteria model is proposed. The two objectives are the collected prize to be maximized and the tour length objective which needs to be minimized. The second objective is brought as a constraint with a limit imposed. The ε-constraint approach is used to solve the model. The dominance property resulting from Proposition 4.1.1 applies to PC-ARCP as well. We write our formulation to find only solutions that satisfy the dominance property.

In addition to the sets, parameters, indices and decision variables that were defined for the MIP Model for ARCP, we need to define some additional ones as follows:

Additional Parameters:

p_q : Prize of connecting component q to q_d, $\forall q \in Q$

t_{max} : Tour length limit

ε : Multiplier of the tour length in the objective function

Additional Decision Variables:

W_q : Binary variable which equals 1 if component q is connected to q_d and 0 otherwise, $\forall q \in Q \setminus q_d$

MIP Model for PC-ARCP:

$$\text{Maximize} \sum_{q \in Q \setminus \{q_d\}} p_q W_q - \varepsilon \Big(\sum_{(i,j) \in E} c_{ij}(X_{ij} + X_{ji}) + \sum_{(i,j) \in B} b_{ij} Z_{ij} \Big)$$

subject to:

$$\sum_{i \in V_q} Y_i \geq W_q \qquad \forall q \in Q \setminus \{q_d\} \tag{4.22}$$

$$\sum_{(i,j) \in E} c_{ij}(X_{ij} + X_{ji}) + \sum_{(i,j) \in B} b_{ij} Z_{ij} \leq t_{max} \tag{4.23}$$

$$W_q \in \{0, 1\} \qquad \forall q \in Q \tag{4.24}$$

Constraints (4.1)-(4.4)

Constraints (4.6)-(4.21)

In the above formulation, the first equation gives the objective function which is maximizing the difference between sum of the prizes collected by connecting each of the components to the component containing the depot and the tour length multiplied by ε. Note that numerically, ε should be small enough not to influence collecting as much prize as we can in the time limit. In our tests, setting it equal to 0.01 was

sufficient in all cases. Constraints (4.22) ensure that if W_q is 1, then at least a node in component q must be visited. Constraint (4.23) limits the tour length to t_{max}. Finally, (4.24) defines W_q as binary variables.

4.3 MIP Model for ARCNCP

In this section, a minimization MIP model is proposed to solve ARCNCP exactly. Note that the dominance property resulting from Proposition 4.1.1 applies to ARCNCP as well. We write our formulation to find only solutions that satisfy the dominance property. In addition to the decision variables, objective function and constraints that were defined for the MIP Model for ARCP, we need to define some additional ones as follows:

Additional Decision Variables:

h_{ij}^k : The amount of flow related to supply/demand node k on edge (i,j) in the direction from node i to node j, $\forall k \in S \cup D$, $(i,j) \in E$. Similarly, h_{ji}^k is defined to represent the flow related to the supply/demand node k in the direction from node j to i on edge (i,j).

Additional Constraints:

$$h_{ij}^k \leq X_{ij} \qquad \forall (i,j) \in E, \forall k \in S \cup D \tag{4.25}$$

$$h_{ji}^k \leq X_{ji} \qquad \forall (i,j) \in E, \forall k \in S \cup D \tag{4.26}$$

$$\sum_{\substack{j \in V: \\ (d,j) \in E}} h_{dj}^k - \sum_{\substack{j \in V: \\ (j,d) \in E}} h_{jd}^k = 1 \qquad \forall k \in S \cup D \tag{4.27}$$

$$\sum_{\substack{j \in V: \\ (i,j) \in E}} h_{ij}^k - \sum_{\substack{j \in V: \\ (j,i) \in E}} h_{ji}^k = 0 \qquad \forall i \in V \setminus \{d,k\}, \forall k \in S \cup D \tag{4.28}$$

$$\sum_{\substack{j\in V:\\(k,j)\in E}} h_{kj}^k - \sum_{\substack{j\in V:\\(j,k)\in E}} h_{jk}^k = -1 \qquad \forall k \in S \cup D \tag{4.29}$$

$$h_{ij}^k, h_{ji}^k \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in S \cup D \tag{4.30}$$

Constraints (4.25) and (4.26) do not allow flow on an edge unless it is traversed. The flow balance equations are Constraints (4.27)-(4.29). Finally, (4.30) defines h_{ij}^k and h_{ji}^k as binary variables.

Chapter 5

HEURISTIC APPROACH

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. The heuristic approach discussed in this chapter works with a solution representation (R, O) for which R and O are defined in Definition 3.1.2. Two types of search are conducted regarding each of the two elements R and O. The first type of search keeps the R part constant and tries to find a better order of unblocking the links, O; whereas, the second type of search seeks for a better R. In the latter one, alternative ways of connecting the components are considered. In this chapter, we describe our heuristic solution method. We use the same heuristic idea for both ARCP and PC-ARCP, however, they differ in the way we construct the initial solutions and the number of neighborhoods. In the heuristic for ARCP, tour length does not increase after implementing the moves, but for PC-ARCP it is possible that by increasing the tour length, we get better solutions. The computational complexity of both heuristics is proportional to $|Q|$ since the neighborhood sizes depend on $|R| = |Q| - 1$.

5.1 Variable Neighborhood Descent

We develop a metaheuristic algorithm which is based on VND, a deterministic version of VNS [13], and constructive heuristics to obtain high quality initial solutions. The neighborhood structures are consecutively repeated within the VND framework. The following pseudo-code gives an outline of the VND algorithm in which K is the number of neighborhood structures used in the algorithm. These neighborhoods are described in Section 7.1.

The neighborhoods used to solve ARCP are as follows ($K = 2 + \lceil \frac{|R|}{2} \rceil$):

- N_1 : Combined *Pair-wise Edge Exchange* and *Deletion/Reconnection* (modifying R or O)
- N_2 : *Postpone* (modifying O)
- N_3 : *Drop/Add* (modifying O)
- N_{i+2} : *Block moves of size* i; $1 < i < \lceil \frac{|R|}{2} \rceil$ (modifying O)

Likewise, the neighborhoods used to solve PC-ARCP are as follows ($K = 10$):

- N_1 : Combined *Pair-wise Edge Exchange* and *Deletion/Reconnection* (modifying R or O)
- N_2 : *Postpone* (modifying O)
- N_3 : *Add* (modifying O)
- N_4 : *Drop* (modifying O)
- N_5 : *Drop/Add* (modifying O)
- N_{i+4} : *Block moves of size* i; $1 < i < 7$ (modifying O)

The reason why we used the two moves called *Add* and *Drop* in PC-ARCP lies in the problem structure that reconnecting the whole network is not a must. Also, the underlying tour length constraint in PC-ARCP imposes an extra computational burden for checking feasibility throughout running the algorithm. Hence, to make it more efficient, we reduced the number of block moves.

In the pseudo-code, the objective value that is minimized is represented by *Obj-Value*. Thus, it equals the tour length in ARCP and the difference between sum of prizes collected and the tour length which is multiplied by $\varepsilon = 0.01$, $\varepsilon(\sum_{(i,j)\in E} c_{ij}X_{ij} + \sum_{(i,j)\in B} b_{ij}Z_{ij}) - \sum_{q\in Q\setminus\{q_d\}} p_q W_q$, in PC-ARCP. The function FEASIBILITY() is described later in Section 5.4.

Variable Neighborhood Descent

Inputs:
W ▷ initial walk
K ▷ number of neighborhoods

```
1: function VND(W,K)
2:     k ← 1
3:     while k < K do
4:         W' ← Best Solution in N_k(W)
5:         FEASIBILITY(W')
6:         if ObjValue(W') < ObjValue(W) then
7:             W ← W'.
8:             k ← 1.
9:         else
10:            k ← k + 1.
11: return W
```

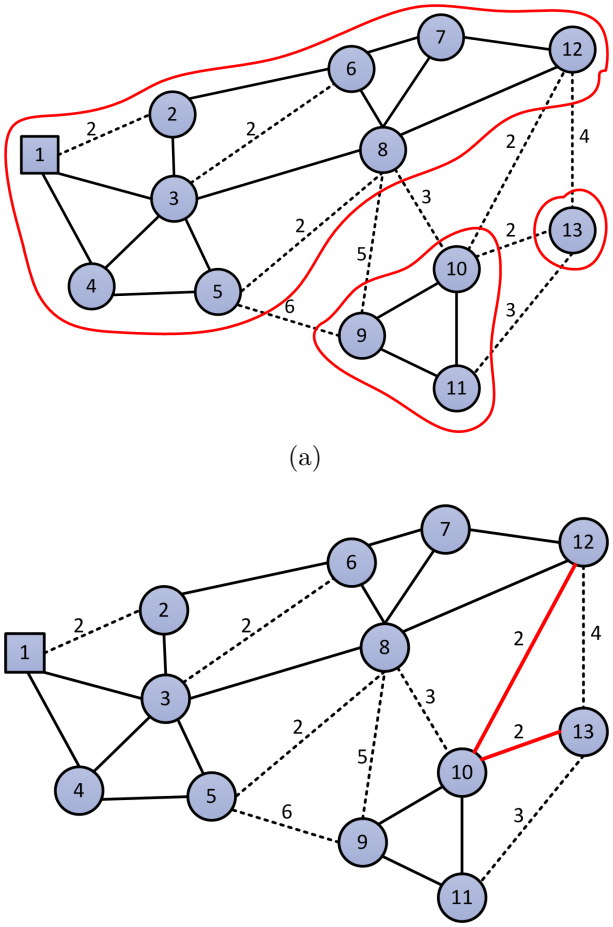
5.2 Initial Solution

For the sake of finding a good initial solution, (R_0, O_0), to ARCP, we first find R_0 and then O_0. We construct a Minimum Spanning Tree (MST) on a transformed graph to identify R_0 and then we use the well-known constructive heuristic for the Rural Postman Problem, which is referred to as Frederickson Heuristic [38] to find O_0. For PC-ARCP, we modify the MST to satisfy the time limit. To find O_0, a constructive heuristic based on Edge Insertion is used.

5.2.1 Finding R_0 for ARCP and PC-ARCP

Deciding on which blocked edges to open is one of the crucial decisions that affects the quality of a solution for both ARCP and PC-ARCP. We seek to find a subset of blocked edges to unblock so that we ensure the connectivity of a network.

For both problems, after omitting the blocked edges and obtaining the graph G^- which is disconnected into $|Q|$ components ($|Q| > 1$), we build an auxiliary graph called $G'' = (V'', E'')$, where E'' is a subset of the set B. We include all blocked edges whose end-nodes are in two different components of G''. As a result, G'' may contain parallel edges. Then, we contract the nodes in a component to a single node



(a)

(b)

Figure 5.1: MST example

representing the component. V'' consists of these newly defined nodes. We construct a Minimum Spanning Tree in G'', for instance using Prim's algorithm. Then for each edge in MST, we take its equivalent edge in G^- and R_0 is fixed in this way for ARCP. For PC-ARCP, we should take care of the time limit constraint in determining R_0. We order the edges in MST according to the Edge Insertion algorithm described in Section 5.2.3. R_0 includes only those edges from MST that form a walk with a tour length less than or equal to t_{max}.

In Figure 5.1, we show a simple example of choosing R_0 using MST. The dashed lines represent the blocked edges and the numbers on top of them show the unblocking costs/times. All traversal costs/times are equal to 1. Figure 5.1 part (a) shows the

input disconnected network with three components. In Figure 5.1 part (b), according to MST, edges $(10, 12)$ and $(10, 13)$ are chosen to be in R_0.

5.2.2 Implementation of Frederickson Heuristic for ARCP

Frederickson Constructive Heuristic [38] is a well-known heuristic for RPP and is adapted commonly to solve several variations of RPP. As described in [65], the heuristic consists of three phases. Phase 1 transforms the R-induced graph (G_R) containing only the required edges (R) into a complete network. Phase 2 applies the MST algorithm to make G_R connected. Phase 3 applies the minimum-cost matching algorithm to the odd-degree nodes to obtain an Eulerian graph. The postman tour can be constructed from the resulting Eulerian graph.

We implemented a quite different version of the algorithm. In the third phase, we implemented a greedy algorithm instead of the exact matching algorithm. Still, the algorithm makes the best choice for each odd-degree node to make it even, thus make the graph unicursal. The idea of this algorithm is simple. At each iteration, it looks at one of the odd-degree nodes of the graph and matches it with one of the remaining odd-degree nodes with a minimum cost and both nodes will get even degrees. It goes on as such until the stopping criterion, that is having even degrees for all nodes, is met. Moreover, in the first and second phases, Floyd-Warshall shortest path algorithm, Prim's minimum spanning tree algorithm and Fleury's algorithm [56] were used in the implementation of this constructive heuristic. Note that as we need an open walk, after unblocking the final required blocked edge on the list, we end the walk. We determine O_0 from this walk. Also, the design of the walk is in a way that it starts always at the depot node.

5.2.3 Edge Insertion Constructive Heuristic for PC-ARCP

The time constraint in PC-ARCP allows us to leave the graph disconnected and unblock less than $|Q| - 1$ blocked edges in $\delta(V_q)$ $\forall q \in Q$. This characteristic lets us add or drop edges from the solution. In our computational tests, we observed better

results than having a random initial solution for PC-ARCP.

In this method, we start with the depot node and insert the edges to the solution iteratively until the tour length exceeds the time limit t_{max}. Let $S \subseteq R$ be the set of R-edges in the solution, $S' \subseteq R$ the set of R-edges that are not inserted into the solution yet and O_S be the order in which the edges in S are to be unblocked. We take each edge $e = (u, v) \in S'$ and find the shortest path from u and v to each end-node of the edges in S. We connect e tentatively by the shortest path among these. Let ΔT_e be the length of this path, and ΔPC_e be the additional prize obtained by including e in the solution. The edge e giving us the maximum $(\frac{\Delta PC_e}{\Delta T_e})$ is included in S. Next, we decide on the best position to insert e to O_S. Let PC be the total prizes collected in our current solution and T the total length (time) of the solution. We calculate $(\frac{PC}{T})$ upon inserting e in each of the positions in O_S and choose the position that maximizes this ratio.

In this method, we most likely cross the feasible boundary of tour length, and enter the infeasible space by some depth, $infeasible\_depth$. Here, instead of removing the last edge on the list and making the solution feasible, we use a strategy to search around the feasible boundary not to miss any good solutions. We turn back to the feasible boundaries by using *Pair-wise Edge Exchange* move subject to a new tour length limit $t_{max} + infeasible\_depth$ instead of t_{max}. No crossing from this new tour length limit is allowed. This way, at every iteration of this search, we aggressively reduce the amount of $infeasible\_depth$ by accepting the solutions only when the depth is reduced. This search will iterate until $infeasible\_depth = 0$ which implies feasibility for the problem.

5.3 Neighborhood Structures

In the local search, each neighborhood is explored with a best improvement approach. These neighborhoods will be described in the following sections. Although it was possible to separate the two moves *Pair-wise Edge Exchange* and *Deletion/Reconnection* and put them in two consecutive neighborhoods in our VND, we combined them into

one neighborhood due to computational superiority of the combined version. In our preliminary tests on some instances from Istanbul and RPP data for the separate version, the R set changed steeply and the frequency of sub-optimal solutions turned out higher.

5.3.1 Combined Pair-wise Edge Exchange and Deletion/Reconnection

This is the only neighborhood entailing a move for the R part of the solution. In this neighborhood, we implement a move to the R part or to the O part or none of them, provided they both have objective values worse than the best solution found so far.

Pair-wise Edge Exchange. We consider swapping the positions of two edges on the list O to reduce the length of the walk. To implement a move of this type, the resulting objective value should be better than the best solution found so far and the objective value of *Deletion/Reconnection* move. This move is illustrated in Figure 5.2. In this figure, C_1, C_2 and C_3 represent the components of the B-induced graph, and the indices of R-edges show their order of being unblocked on the list O. The dashed links show the blocked edges in the cut-sets and the bold links represent the blocked edges that are chosen to be unblocked from the cut-sets. This move consists of $(|Q|-1) \times (|Q|-2)/2$ reachable solutions (size of neighborhood), so the frequency of this move is $O(|Q|^2)$.

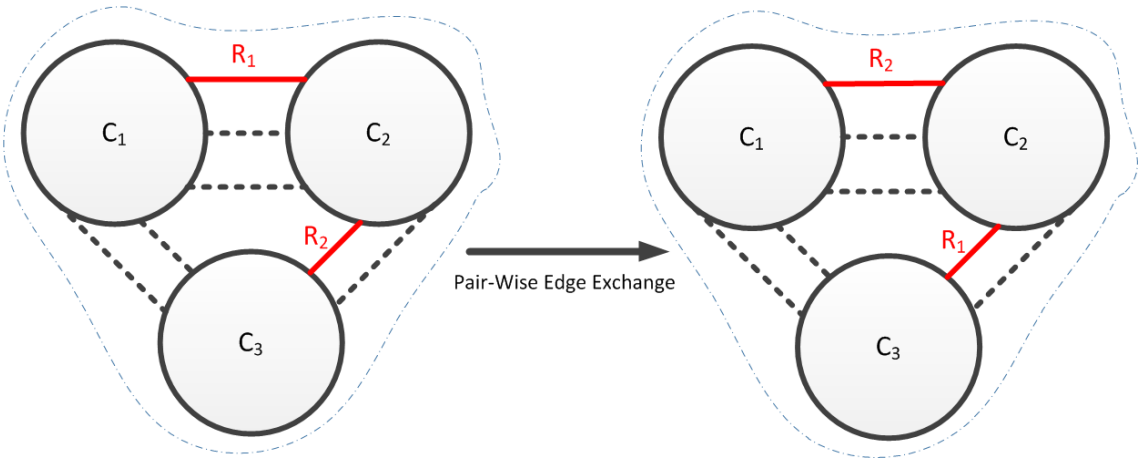


Figure 5.2: Pair-wise Edge Exchange

Deletion/Reconnection. In this move, we try to find alternative ways of connecting the components with regard to the current O. A move consists of removing an R-edge from the set R and reconnecting the resulting two components. Let C be one of the components resulting from removing the R-edge in this move. We reconnect the two components using the blocked edges in $\delta(C)$. As a result of implementing this move, our fixed part of the total tour length which is $\sum_{(i,j)\in R} c_{ij} + b_{ij}$ may increase or stay the same. Our criterion for comparing different solutions to the current solution is the value of objective function with regard to the current order of unblocking O, while the original list of the edges to be unblocked, R, is changing. We consider implementing this move, if its resulting objective value is better than the best solution found so far and the objective value from the *Pair-wise Edge Exchange* move. In the worst case, we have $O(|R| \times |B|)$ moves. This move is illustrated in Figure 5.3. The symbols used in this figure are the same as in Figure 5.2. Note that upon substituting R_1' for R_1, the index (the position on the list O) did not change.

Checking through all the candidates from the set $\delta(C)$ to replace this edge is a time-consuming task. Hence, we assigned a probability of acceptance, $P_{acc}(\Delta_{ij})$, to each edge $(i,j) \in \delta(C)$. The idea of having this probability function is to assign higher probabilities of acceptance to the edges having lesser b_{ij} than average, and at the same time, making it possible to consider some edges with higher b_{ij}'s. We assign less probablities of acceptance to the edges having higher than average costs. By average cost, we mean the average cost of the edges in the cut-set of the resulting two components. This could be any reasonable decreasing function, but we derived by experimenting a non-linear function (see Appendix 10.2) that depends on the difference Δ_{ij} between its total unblocking and traversal time and the average total unblocking and traversal time of the edges, γ, in $\delta(C)$. Δ_{ij} is defined in Equation (5.1) and parameter γ is defined in Equation (5.2) in which parameter n is the total number of blocked edges in $\delta(C)$.

$$\Delta_{ij} = c_{ij} + b_{ij} - \gamma \qquad (i,j) \in \delta(C) \tag{5.1}$$

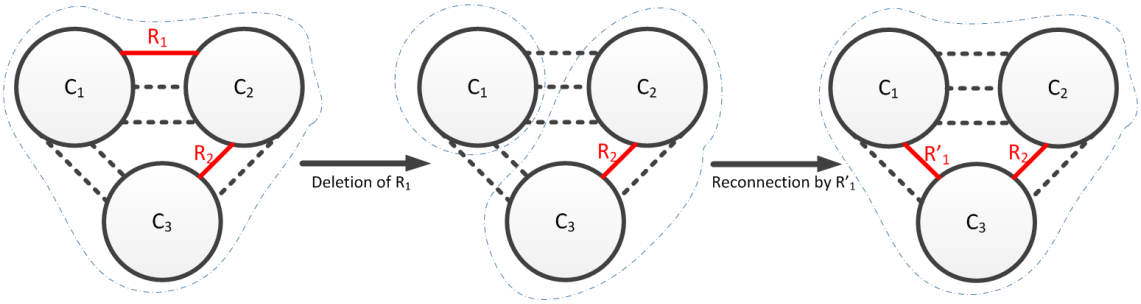


Figure 5.3: Deletion/Reconnection

$$\gamma = \frac{\sum_{(i,j)\in\delta(C)} c_{ij} + b_{ij}}{n} \tag{5.2}$$

5.3.2 Drop/Add and Block Moves

In the *Drop/Add* move, we examine dropping each of the edges on the list O and adding it to other positions on the resulting list. For PC-ARCP, this move works with the edges positioned before the one hitting t_{max}. In the next neighborhoods called Block Moves, sequences of consecutive edges in O called blocks are identified and dropped from their current positions and then added to other positions. In the worst case, it is $O(|Q|^3)$.

5.3.3 Drop Move

In the *Drop* move, we consider dropping each of the edges on the list O positioned before the one hitting t_{max} and adding it to the end of the list. In the worst case, it is $O(|Q|)$.

5.3.4 Add Move

In the *Add* move, we consider dropping each of the edges on the list O positioned after the one hitting t_{max} and adding it to the best place before t_{max}. In the worst case, it is $O(|Q|^2)$.

5.3.5 Postpone

In a feasible solution to ARCP or PC-ARCP, it may be necessary to traverse an unblocked edge once or more than once. These repetitions give us an opportunity to check for a possible shorter path. In the *Postpone* move, each R-edge is checked to see if it is traversed more than once in the walk. The move considers postponing the unblocking of that R-edge to one of the later positions to detect a possible improvement on the tour length. In the worst case, the neighborhood size is $O(|Q|^2)$. This move is similar to the procedure mentioned in [46]. This move is illustrated using a simple example in Figure 5.4, where bold arrows represent unblocking of an edge and the dashed ones show traversal. In this example, all traversal costs/times and unblocking costs/times equal 1. The length of the partial walk is 12 in part (a). As we see, edge (i_2, i_3) is traversed once after it is unblocked. In parts (b) and (c) of this figure, one can see that by postponing the unblocking of this edge to the place where it is traversed later, the length of the partial walk decreases to 10, since a shorter path from i_1 to i_4 is found with length 1 that does not include edge (i_2, i_3).

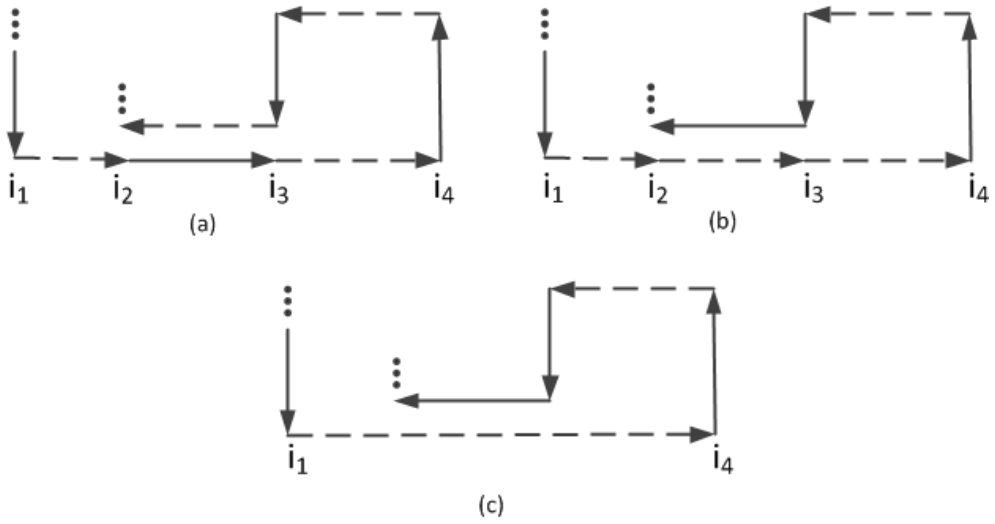


Figure 5.4: An example to the *Postpone* move

5.4 Feasibility Algorithm

Feasibility is checked after implementing any move on the current solution, so that the vehicle does not pass a blocked edge before unblocking it. Given any order of the blocked edges to unblock, the algorithm detects any infeasibility and makes the solution feasible and takes it as the current feasible solution. For PC-ARCP, in addition to the one mentioned above, we check to avoid exceeding the time limit t_{max} on the tour length by removing the last R-edges on O, the one hitting t_{max} and the ones after that. This algorithm is described in the following pseudo-code in which T_W is the length of the walk.

Feasibility Algorithm

```
1: function FEASIBILITY(W)
2:     for (i, j) ∈ B do                          ▷ For both ARCP and PC-ARCP
3:         if (i, j) is repeated in W then
4:             Unblock it before traversing it
5:     if T_W > t_max then                                ▷ Only for PC-ARCP
6:         Remove the last edges on the list O_s, the one hitting t_max and the ones
   after that
7: return W
```

Let us see an example of implementing this algorithm in which all traversal and unblocking costs/times equal 1. In Figure 5.5, the R-edges are to be unblocked in the order $O = \{e_1, e_2, e_3, e_4\}$. However, as seen is part (a) of this figure, this solution is infeasible since edge e_3 is traversed in the path from e_1 to e_2 before it is unblocked. The order of unblocking should be $\{e_1, e_3, e_2, e_4\}$ to make this solution feasible (see part (b)). Regarding this feasible order, the algorithm chooses the shortest path from e_2 to e_4 that may not necessarily include e_3. The shortest path length from e_2 to e_4 that was calculated at the beginning of the algorithm is 2 (red dashed links in part(c)).

Note that if passing through unblocked e_3, with cost/time c_{e_3} instead of $c_{e_3} + b_{e_3}$, for getting from e_2 to e_4 could lead to a shorter path, it has already been considered in our initial shortest path calculation. As seen in part (c), the green dashed links' cost/time amount to 3 which implies a longer path.

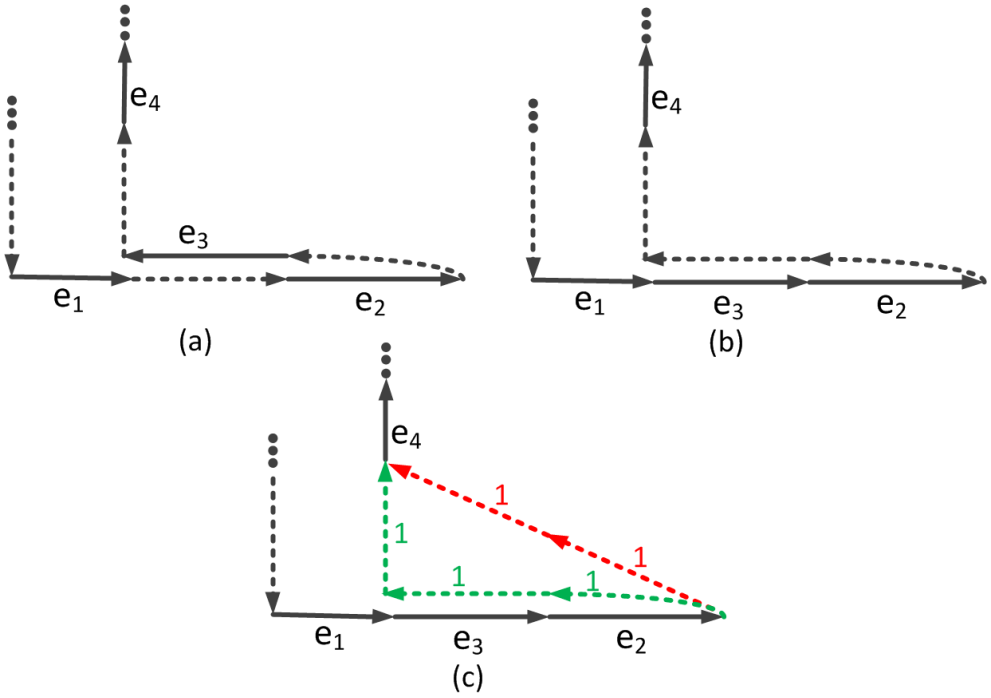


Figure 5.5: An example to implementing the Feasibility Algorithm

Chapter 6

DESIGN OF COMPUTATIONAL EXPERIMENTS

There are no benchmark instances in the literature for the problems discussed in this thesis. However, due to similarity of our problems to RPP, we adapted RPP instances available in the literature. Moreover, some real-life data from Istanbul city road network is generated and tested in this work. We first describe the data of the Istanbul instances.

6.1 Istanbul Network

We constructed the Istanbul road network by considering province centers and road distances. In this section, we describe how these instances were generated.

Nodes. 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). We chose Arnavutkoy center as the depot, since it is among the potential depot locations.

Edges. 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. Travel times are determined according to road distances which are the shortest path lengths calculated by ArcGIS. We have assumed on average 40km/h fixed speed for vehicles and converted road distances to time measure by dividing distance by speed.

Blocked Edges. We selected the blocked edges randomly such that the probability of failure for each edge is set according to the risk maps and classification of risky highway components given in JICA report [49] (for Model C). The JICA report provides four highly probable earthquake scenarios and Model C is the worst-case scenario in the Marmara fault zone. This fault zone lies about 20 km south of the city and poses a high seismic hazard risk to Istanbul [52]. By generating random numbers between zero and one, and comparing them to the probability of failure for that specific edge, we consider that edge as blocked or open.

The probability of failure for each link (i, j), $P_{ij}(failure)$, is calculated by considering three criteria: the seismic zone factor based on where edge (i, j) is located (β_{ij}), its distance from the fault line of the earthquake (x_{ij}) and the measure of disaster intensity or magnitude (μ).

$$P_{ij}(failure) = \beta_{ij}\mathrm{PGA}_{ij}, \tag{6.1}$$

where PGA_{ij} is the peak ground acceleration at link (i, j). It is formulated in [62] as:

$$\mathrm{PGA}_{ij} = \alpha\frac{e^{0.8\mu}}{(x_{ij}+40)^2}, \tag{6.2}$$

where α is a parameter used to determine the range of survival probabilities and we set it equal to 3.2. Seismic zone factors represent the damage risk of the region in an earthquake and according to Model C in JICA report for Istanbul city, it can be one of the numbers from the following set: $\{0.95, 0.85, 0.75, 0.65\}$ with 0.95 representing the damage risk of the riskiest region among the four different zones in Istanbul. We assign them depending on the position of the edge on the map (see Appendix 10.3).

Prizes. In order to generate prizes for PC-ARCP instances, we calculated the population of the area represented by each node, according to population data of the districts of Istanbul (Turkish Statistical Institute). Although the junction points did not have any population according to our data, we assigned a small population to them so that the solution methods used to solve PC-ARCP connects the components

consisting of merely junction points as the least priority tasks. The population assigned to a component containing only junction points should not exceed that of any other component that contains demand points. Thus, we assigned a population equal to the minimum population of a node among all nodes in the set V divided by the maximum number of junction points contained in one component among all components in the set Q over all of our instances. For Istanbul data, this amounts to 1077. We then set the prize of a component as the total population of the nodes in it divided by 1000.

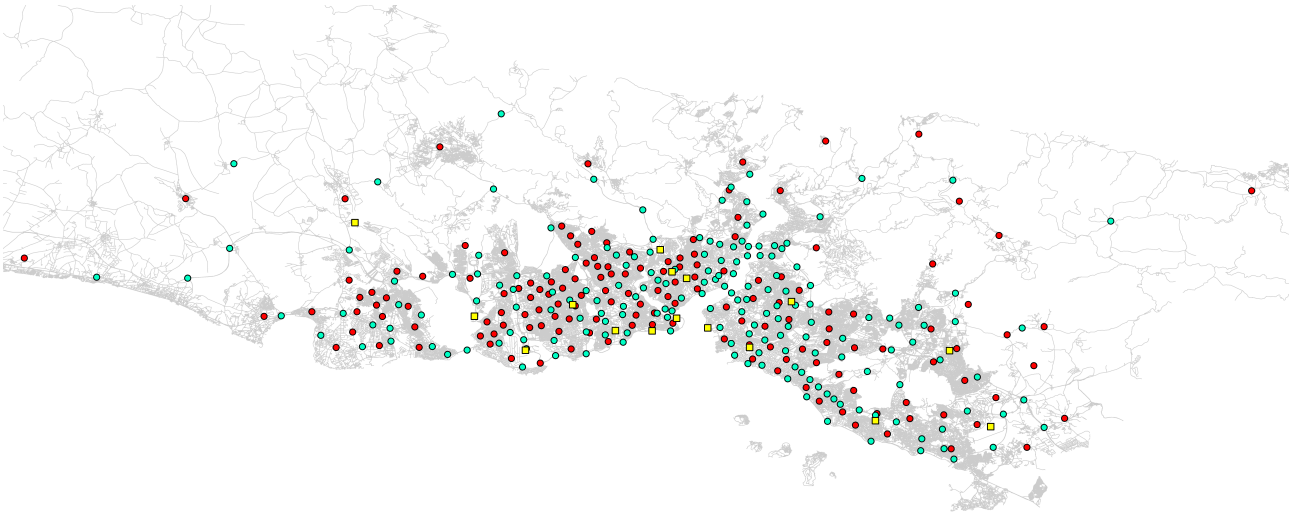


Figure 6.1: Istanbul network nodes

Table 6.1 lists the characteristics of the instances. $|V|$ is the number of nodes in a network, $|E|$ is the number of edges, $|B|$ is the number of blocked edges and $|Q|$ is the number of connected components of a network. The last column in Table 6.1 shows the disaster magnitude, which affects the calculations related to probability of failure of an edge (see equation (6.2)) and the corresponding traversal and unblocking time of that edge (see Section 10.1). Note that in Low, Medium and High disaster magnitudes, we expect μ in equation (6.2) to be 6.6, 7 and 7.4, respectively, according to the earthquake scenarios in the JICA report. For each magnitude level, we generated four instances randomly as listed in the first column of Table 6.1.

Table 6.1: Characteristics of the Istanbul network instances

Disaster Scenarios	**|V|**	**|E|**	**|B|**	**|Q|**	**Disaster Magnitude**
L1	349	647	172	12	Low
L2	349	647	192	14	Low
L3	349	647	200	15	Low
L4	349	647	204	17	Low
M1	349	647	240	26	Medium
M2	349	647	252	29	Medium
M3	349	647	226	30	Medium
M4	349	647	250	35	Medium
H1	349	647	310	38	High
H2	349	647	284	39	High
H3	349	647	311	44	High
H4	349	647	320	44	High

6.2 RPP Instances

We found several sets of instances used by other authors. We chose some city-like networks from the instances given in `(www.mai.liu.se/~kajho48/problemdata/rup)`. The instances whose names start with ALBA or I are based on the city Albaida and the ones starting with MADR are based on Madriguera. Required edges in these instances were selected in various random ways [48] and we set the required edges in these networks as blocked edges in our problem. The number of nodes in each component $q \in Q$ in the B-induced graph (G^-) is simply set as the prize of that component for PC-ARCP. We can see the information about this set of instances in Table 6.2. We use the same notation with Table 6.1.

Note that in the calculations of traversal and unblocking times of the edges in these instances, we assumed the disaster magnitude to be low and used the method in Section 6.3. Then, we changed the number units to hours assuming the speed of a vehicle being equal to 40 km/hr.

Table 6.2: Characteristics of the RPP instances

Instances	|V|	|E|	|B|	|Q|	Instances	|V|	|E|	|B|	|Q|
I20	50	98	63	32	MADR31	196	316	86	9
I21	49	110	67	26	MADR32	196	316	108	17
I22	50	184	74	20	MADR55	196	316	147	36
I23	50	158	78	23	MADR34	196	316	101	12
I24	41	125	55	20	MADR35	196	316	95	6
ALBA31	116	174	51	12	dv1	320	480	91	27
ALBA52	116	174	92	38	dv2	320	480	105	36
ALBA33	116	174	44	12	dv3	320	480	136	46
ALBA34	116	174	49	14	dv4	320	480	172	57
ALBA35	116	174	57	15	B451	465	1055	377	20

6.3 Traversal and Unblocking Times

We developed a method to generate the traversal and unblocking time of a road. This method incorporates several factors related to how the roads may be blocked and assigns probabilities with respect to these factors. For all instances in this thesis, we use this method to generate the traversal and unblocking times.

In our settings, we assume that the damage level can be one of the three types: 1) light debris or destruction, 2) bypassable heavy debris or destruction and 3) non-bypassable heavy debris or destruction. Note that the Rapid Earthquake Response Systems such as the ones used in Istanbul can estimate damage in quasi-real time after an earthquake according to output data from pre-installed sensors [33]. In the second condition, we can find an alternative way to bypass the debris or the damaged road. In the third condition, there is no other way of doing so, such as collapsed viaducts. The heavy bypassable condition can be cleared in a longer time than the light debris condition. The third condition might take days due to activities such as repairing a viaduct, or exploding rocks, or rebuilding the infrastructures. We assigned condition $k \in \{1, 2, 3\}$ to $P_k\%$ of the roads. The percentages vary according to disaster magnitude as given in Table 10.3 in Appendix 10.1. We assume that an edge is partially under conditions 1, 2 and 3, and the percentage of condition k on edge (i, j)

is p_{ij}^k. We choose p_{ij}^k randomly with $U(0, P_k)$.

We used three factors that affect b_{ij}'s and c_{ij}'s. These factors are set according to Table 10.2 in Appendix 10.1 depending on whether the road is narrow, in a residential area or under flooding or liquefaction. The factor for flooding or liquefaction is assigned after the disaster. See Appendix 10.1 for an explanation of how these parameters are generated.

Other methods can be used to estimate the traversal and unblocking times either for planning purposes or for post-disaster estimation. For instance, FEMA's Hazus risk assessment program (http://www.fema.gov/hazus) provides estimates of losses from disasters such as floods, hurricanes and earthquakes.

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. However, in some smaller-sized instances, the Cplex could find the optimal values, so we were able to compare the heuristics with the optimal values. In this chapter, we present the results and analyses on the solution characteristics.

7.1 Settings

The experiments were conducted on a workstation with Intel Xeon E5-2643 3.30 GHz Quad-Core processor, 32 GB RAM and 64-bit Windows 7 Professional operating system. The heuristic algorithms are coded in C# and run on Visual Studio 2012. In all of the following results, we have the exact model solved using GAMS 24.0 (Cplex 12.5) and the results are given by the Cplex UB or Cplex LB columns in the tables. The heuristic algorithms are under the names H1 for the one that works with a fixed R set ($P_{acc}(\Delta_{ij}) = 0$ for all $(i,j) \in E$), and H2 with $P_{acc}(\Delta_{ij})$ given by the function described in equation (10.6) for all $(i,j) \in E$. In both H1 and H2 for ARCP, the number of neighborhoods that we used, K is $2 + \lceil \frac{|R|}{2} \rceil$, and for PC-ARCP, $K = 10$. The order in which the neighborhoods are searched in the VND is described in Section

.

For PC-ARCP, the maximum number of edges in a "block" for the block moves is 6. However, for ARCP, the maximum number of edges in a block differs with respect to $|R|$. Starting from two, each block move is implemented if it contributes to the objective function value and it continues until blocks of size $|R|/2$ are tested.

We replicated H2 five times since there is randomness in acceptance of different

blocked edges to be substituted although we did not observe much difference in the quality of the solutions in different replications. The reported elapsed time and percentage gap are the average elapsed time and average percentage gap over all replications. Two-hour, three-hour or four-hour time limits were imposed to Cplex to report bounds with regard to the instance sizes.

The depot location could change in the computational experiments. In Istanbul case, we consider the depot location to be in Arnavutkoy center and in RPP instances, we consider it to be node 0. However, after testing different instances with different depot locations, we found out that the location of the depot does not affect the difficulty of the problems. Hence we considered a fixed location for the depot in our experiments and analyses.

7.1.1 Choosing and Ordering the neighborhoods for the VND

In some arc routing problems in the literature similar to ARCP and PC-ARCP, analogous neighborhood structures with quite the same order were used and the results were satisfactory (for example, see [12] and [77]). The neighborhoods and the order in which they are searched within the VND were set according to our preliminary tests done on a subset of the instances.

In Tables 7.1 and 7.2, we show different settings of the VND for ARCP and PC-ARCP respectively with regard to the order and the choice of neighborhoods used in our preliminary tests. Note that N_k represents the neighborhood k explained in Chapter 5. In Tables 7.3 and 7.4 the results of running algorithm H2 for ARCP and H2 for PC-ARCP respectively on a test bed of 10 RPP instances are listed. The second to the 10th columns of Table 7.3 depict the tour length of the solution obtained by solving H2 for ARCP having the order and choice of the neighborhoods specific to the corresponding setting. The second to the 10th columns of Table 7.4 depict the total collected prize of the solution obtained by solving H2 for PC-ARCP having the order and choice of the neighborhoods specific to the corresponding setting. The 12th row presents the average percentage gap with regard to each setting compared to

the best tour length (prize) given by Cplex, shown in the 11th column. According to these tables, we conclude that the Setting 9 is the best choice for both ARCP and PC-ARCP due to having the least average percentage gaps.

Table 7.1: Settings of the neighborhoods used for the VND for ARCP

Setting 1	$N_2 \rightarrow N_3 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil)$
Setting 2	$N_1 \rightarrow N_2 \rightarrow N_3$
Setting 3	$N_1 \rightarrow N_3 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil)$
Setting 4	$N_1 \rightarrow N_3 \rightarrow N_2 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil)$
Setting 5	$N_3 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil) \rightarrow N_1 \rightarrow N_2$
Setting 6	$N_3 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil) \rightarrow N_2 \rightarrow N_1$
Setting 7	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_{i+2}(1 < i < 6)$
Setting 8	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_{i+2}(1 < i < \rightarrow N_{i+2}(1 < i < \frac{\lvert R \rvert}{3}\})$
Setting 9	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_{i+2}(1 < i < \lceil \frac{\lvert R \rvert}{2} \rceil)$

Table 7.2: Settings of the neighborhoods used for the VND for PC-ARCP

Setting 1	$N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7)$
Setting 2	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4$
Setting 3	$N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7)$
Setting 4	$N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow N_2 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7)$
Setting 5	$N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7) \rightarrow N_1 \rightarrow N_2$
Setting 6	$N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7) \rightarrow N_2 \rightarrow N_1$
Setting 7	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 5)$
Setting 8	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 6)$
Setting 9	$N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow N_5 \rightarrow N_{i+4}(1 < i < 7)$

7.1.2 Deciding on $P_{acc}(\Delta_{ij})$ for H2

In this section, we show our preliminary tests on the algorithm H2 to decide on $P_{acc}(\Delta_{ij})$ for both ARCP and PC-ARCP. To this end, we formed a test bed of eight instances from the literature having random settings for b_{ij}'s. We test 6 different settings and test each of the settings on all seven instances and compare the tour lengths with the results obtained by Cplex. For each test, five replications are used. We present the results in Table 7.5 in which the first five columns list the instances' characteristics and the rest show the heuristics solutions' percentage gaps with respect

Table 7.3: The results of the tests done on the heuristics for ARCP to decide on the order and choice of the neighborhoods

Instance	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5	Setting 6	Setting 7	Setting 8	Setting 9	Best
I20	23.382	23.381	23.381	23.382	23.381	23.382	23.384	23.382	23.353	23.35
I21	0.900	0.903	0.900	0.903	0.900	0.900	0.903	0.900	0.901	0.89
I22	0.956	0.955	0.955	0.955	0.955	0.955	0.956	0.955	0.953	0.95
I23	0.970	0.970	0.970	0.970	0.970	0.970	0.970	0.970	0.967	0.95
I24	0.900	0.900	0.890	0.890	0.900	0.900	0.890	0.900	0.889	0.88
ALBA31	12.440	12.100	12.100	12.100	12.440	12.440	12.440	12.100	12.103	11.90
ALBA52	37.020	36.930	36.570	36.270	36.730	36.570	36.790	36.790	36.340	36.19
ALBA33	11.315	11.102	11.102	11.102	11.315	11.092	11.102	11.102	11.060	11.06
ALBA34	13.780	13.780	13.780	13.780	13.780	13.780	13.780	13.780	13.784	13.62
ALBA35	17.640	17.520	17.520	17.520	17.640	17.640	17.520	17.640	17.525	17.52
Gap (%)	1.60	1.05	0.81	0.76	1.51	1.27	1.20	1.05	0.65	

Table 7.4: The results of the tests done on the heuristics for PC-ARCP to decide on the order and choice of the neighborhoods

Instance	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5	Setting 6	Setting 7	Setting 8	Setting 9	Best
I20	25	49	49	49	49	25	49	49	49	49
I21	35	36	39	38	41	38	37	39	41	41
I22	41	41	42	41	42	42	41	42	42	43
I23	41	43	44	43	43	41	44	44	44	44
I24	31	33	31	33	33	33	33	31	33	33
ALBA31	108	112	108	112	112	108	108	112	112	112
ALBA52	92	99	108	105	108	106	103	107	108	108
ALBA33	103	111	111	111	112	112	111	111	112	112
ALBA34	106	110	109	110	106	110	109	110	110	110
ALBA35	95	107	109	109	104	109	109	104	109	109
Gap (%)	12.4	3.01	1.86	1.79	1.28	7.08	2.44	1.96	0.15	

to the Cplex solutions. Note that for H_0, $P_{acc}(\Delta_{ij}) = 0 \quad \forall (i,j) \in \delta_Q$, for $H_{0.5}$, $P_{acc}(\Delta_{ij}) = 0.5 \quad \forall (i,j) \in \delta_Q$ and for H_1, $P_{acc}(\Delta_{ij}) = 1 \quad \forall (i,j) \in \delta_Q$. H_{linear} uses a probability function that is totally linear, similar to the function described in Equation (10.6), except that it does not have the nonlinear part and is all linear. $H_{nonlinear1}$ uses a probability function that is nonlinear, similar to the function described in Equation (10.6), except that the slope of the linear part is bigger, hence the point where the linear function changes to exponential, is also different. Finally, $H_{nonlinear2}$ uses the probability function described in Equation (10.6). We observe that when the probabilities of acceptance decrease, the solution quality deteriorates and the heuristic

becomes more time-efficient. Although the average percentage gap given by H_1 is less, $H_{nonlinear2}$ is more time-efficient and in the instances having high $|B|$ values, the run-time of H_1 is prohibitively high. This justifies the benefit of using the settings of $H_{nonlinear2}$ in our VND algorithm.

7.2 ARCP Results

In this section, we present the test results on 32 instances entailing 12 Istanbul city road network scenarios and 20 different city-based instances from RPP (see Chapter 6). The performance of the heuristics are compared to the exact solution found by solving the MIP Model for ARCP to optimality for the smaller sized instances and in the cases that Cplex could not find the optimal solution in the given time limit, the solution of the heuristics are compared to the upper bound found by Cplex in the given time limit.

Table 7.6 shows the results for Istanbul city road network instances. The percentage gaps of H1 and H2 are given in the second and fourth columns. Note that for H2, the average over five replications is reported. As we see, optimal solutions are found in four instances by Cplex and near optimal solutions in eight instances. The percentage optimal gap reported by Cplex is presented in the sixth column. The average gap for Cplex runs is 1.2%.

The percentage gaps in the second and fourth columns of Table 7.6 are calculated as follows:

$$Gap_H(\%) = \frac{X_H - X_{UB}}{X_{UB}}.100\ \% \tag{7.1}$$

where X_{UB} is the best integer solution known for the problem which is given by the Cplex upper bound in the given run-time limit. The Cplex optimality gap is calculated as follows:

$$Gap_{UB}(\%) = \frac{X_{UB} - X_{LB}}{X_{LB}}.100\ \% \tag{7.2}$$

where X_{LB} is the lower bound given by Cplex. In terms of solution quality, H2

Table 7.5: The results of the tests done on H2 for ARCP to decide on $P_{acc}(\Delta_{ij})$

Instance	|V|	|E|	|B|	|Q|	H_0	time	$H_{0.5}$	time	H_1	time	H_{linear}	time	$H_{nonlinear1}$	time	$H_{nonlinear2}$	time
N502	50	204	125	7	0	0.17	0	0.5	0	1	0	0.5	0	0.4	0	0.4
NC1003	100	200	92	11	6.111	0.66	4.717	13.786	1.187	26.38	7.932	13.4	3.50	14.935	3.435	13.216
NC501	50	98	60	17	7.145	0.579	5.629	4.252	5.629	11.7	5.629	4.18	5.629	3.40	5.629	2.43
N503	50	163	140	29	0.931	3.59	0.168	14.13	0.056	27.5	0.379	10.22	0.056	16.086	0.075	15.988
A3101	116	174	73	18	9.859	1.22	8.325	17.584	6.5308	48.08	9.860	21.13	7.216	22.054	6.530	25.723
A3101’	116	174	73	18	1.075	2.6	2.953	38.186	4.429	58.56	4.429	78	3.691	37.371	1.476	59.276
A3101”	116	174	73	18	0.616	2.5	0	55.849	0	111	0	67	0	52.599	0	49.5
NC1001	100	200	98	14	1.157	3	0.83	534	-3.73	1160	0.863	453	0.87	338	0.878	333
Average					3.362	2	2.828	85	1.763	181	3.635	81	2.621	61	2.253	62

outperforms H1 but the run-times of H2 are much higher than those of H1. Both H1 and H2 did better in low magnitude instances (smaller $|Q|$) compared to higher magnitude instances. Except in one case, H1 and H2 could not find the optimum but their run times are generally much less than Cplex. The run-times are shown in third, fifth and seventh columns. In the seventh column, 10800 means the run-time limit of three hours was imposed.

Although there is not a big difference between the solution qualities of the heuristic algorithm and the exact model (H1 average gap is 1.6% and H2 average gap is 1%), there is a relatively considerable run time difference between H1 and Cplex.

Table 7.6: Percentage gaps of ARCP for Istanbul network instances

Instance	**Gap_{H1} (%)**	**H1 time (sec)**	**Gap_{H2} (%)**	**H2 time (sec)**	**Gap_{UB} (%)**	**Cplex time (sec)**
L1	5.40	14	4.51	129	0.00	420
L2	4.76	14	0.00	310	0.00	1258
L3	3.51	14	3.51	187	1.42	10800
L4	0.20	20	0.27	177	0.85	10800
M1	1.43	52	1.14	1278	0.66	10800
M2	0.13	132	0.13	1015	0.90	10800
M3	0.88	44	0.02	243	0.93	10800
M4	0.23	237	0.04	2523	0.00	2198
H1	0.92	148	0.78	1171	0.68	10800
H2	0.29	381	0.34	1274	0.75	10800
H2	-0.04	386	-0.04	1853	8.56	10800
H4	1.98	314	1.60	3563	0.00	9800
Average	1.64	146	1.03	1143	1.23	8339

In Tables 7.7 and 7.8, the results for the RPP data are presented. In Table 7.7, unblocking times b_{ij}'s are generated randomly using the parameter settings described in Section 6.3, and Table 7.8 has $b_{ij} = c_{ij}$. The optimal solutions were found by Cplex in all instances except two.

By looking at each of the Tables 7.7 and 7.8, one can easily observe that the problem becomes more difficult with increasing number of components ($|Q|$). Moreover, number of nodes is influential on the problem difficulty and the upward trend in the run times is especially seen in the model solution.

Table 7.7: Percentage gaps of ARCP for RPP instances (Random b_{ij})

Instance	**Gap$_{H1}$ (%)**	**H1 time (sec)**	**Gap$_{H2}$ (%)**	**H2 time (sec)**	**Gap$_{UB}$ (%)**	**Cplex time (sec)**
I20	0	7	0	17	0	13
I21	0	3	0	5	0	4
I22	1	1	0	2	0	14
I23	2	2	2	6	0	24
I24	2	1	1	1	0	8
ALBA31	2	1	2	8	0	3
ALBA52	0	47	0	123	0	965
ALBA33	0	1	0	3	0	3
ALBA34	1	1	1	1	0	12
ALBA35	0	1	0	9	0	3
MADR31	0	3	0	36	0	1282
MADR32	6	7	2	232	0	6460
MADR55	0	49	0	444	1	10800
MADR34	0	3	0	21	0	19
MADR35	0	2	0	4	0	99
dv1	0	27	0	164	0	1770
dv2	0	95	0	644	0	140
dv3	0	361	0	1444	0	200
dv4	0	845	0	1808	0	297
B451	1	70	1	2195	37	14400
Average	0.75	76	0.45	358	1.90	1825

Comparing the results in Tables 7.7 and 7.8, we see different solution qualities with respect to different cost structures. The Cplex bounds are loose in some instances in Table 7.8 and in some cases, the heuristics are doing worse than in Table 7.7. We associate this difference to the fact that when b_{ij}'s are higher, the fixed cost (total cost of unblocking the edges in R) to make the whole network connected is higher too, knowing that the choice of R among different solutions is usually the same. Hence, the gap between different feasible solutions which we believe is usually due to the routing aspect of the algorithms (choosing the order of unblocking the edges and the walk), diminishes. H2 showed a better average quality than H1 in both Tables 7.7 and 7.8. The average Cplex gap increased from 2% in Table 7.7 to 9% in Table 7.8. Average percentage gaps of H1 and H2 also increase from less than 1% in Table 7.7 to 9% and 6% in Table 7.8 respectively.

Finally, we analyze the sensitivity of H1 and H2 to the problem parameters. To

Table 7.8: Percentage gaps of ARCP for RPP instances ($b_{ij} = c_{ij}$)

Instance	**Gap$_{H1}$ (%)**	**H1 time (sec)**	**Gap$_{H2}$ (%)**	**H2 time (sec)**	**Gap$_{UB}$ (%)**	**Cplex time (sec)**
I20	4	7	2	13	0	171
I21	7	3	9	6	0	131
I22	6	1	5	2	0	1601
I23	5	2	4	7	0	74
I24	9	1	3	3	0	25
ALBA31	10	1	13	4	0	75
ALBA52	12	60	10	81	0	6285
ALBA33	12	1	9	9	0	5
ALBA34	8	1	5	7	0	63
ALBA35	0	1	0	14	0	6
MADR31	13	3	12	49	8	10800
MADR32	12	4	12	128	16	10800
MADR55	18	35	1	667	31	10800
MADR34	4	4	-1	138	22	10800
MADR35	25	2	25	4	0	9080
dv1	6	30	5	408	0	3793
dv2	5	61	2	1535	0	8396
dv3	9	169	5	2917	0	4410
dv4	9	898	4	3785	2	14400
B451	5	47	4	3004	113	14400
Average	9.04	66	6.54	639	9.58	5305

this end, we look at the results from solving the 12 Istanbul instances using H1 and H2 for ARCP. Note that we chose Istanbul instances for this purpose, since they all have the same network structure but different disaster scenarios, meaning different number and positions of blocked edges and connected components. First, we sorted the instances with regard to parameter |Q|, and plotted the elapsed time of H1 and H2 and then sorted according to |B| and plotted the elapsed time of H1 and H2. As seen in Figures 7.1 and 7.2, there is a general upward trend in these plot. However, the run-time of H1 or H2 for ARCP is not strictly increasing with regard to |Q| or |B|. Number and positions of the blocked edges on the network is influential and may cause the run-time plot not to be strictly increasing with regard to |Q| or |B|.

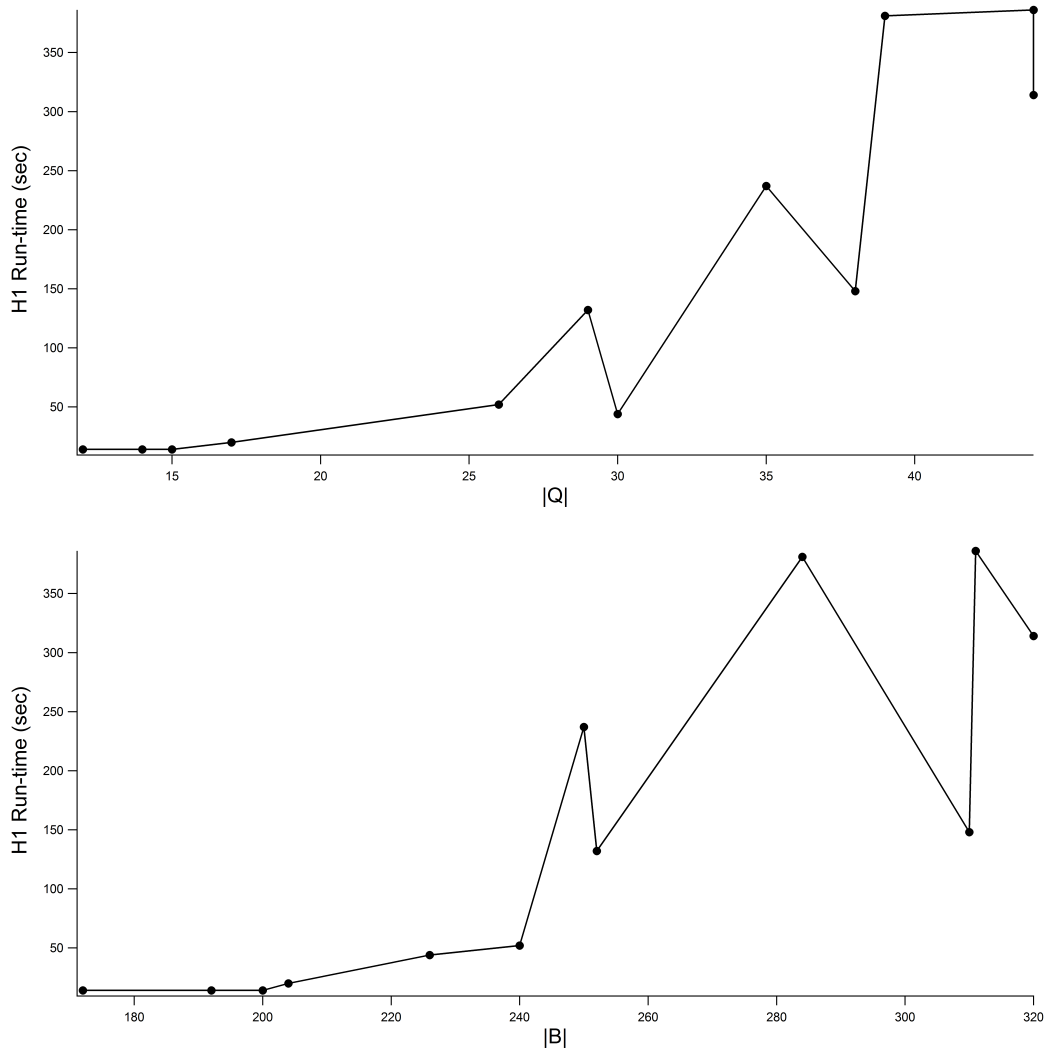


Figure 7.1: H1 for ARCP run-time analysis

7.2.1 Solving Z-Relaxed MIP Model for ARCP

In order to evaluate the routing aspect of our heuristic algorithms, we relaxed the binary variables Z_{ij}, $\forall (i,j) \in B$ by letting $Z_e^0 = 1$, $\forall e \in U$, where U is the set of edges to be unblocked in the final solution to the algorithm H1 for ARCP, and $Z_e^0 = 0$, $\forall e \in B \setminus U$. The optimal solution to this relaxed model which can be found way more quickly than the MIP Model for ARCP shows us the best we can do if the edges to be unblocked are known in advance. Comparing the results of heuristic H1 for ARCP with this Z-Relaxed MIP Model for ARCP using a few RPP instances, we observed a

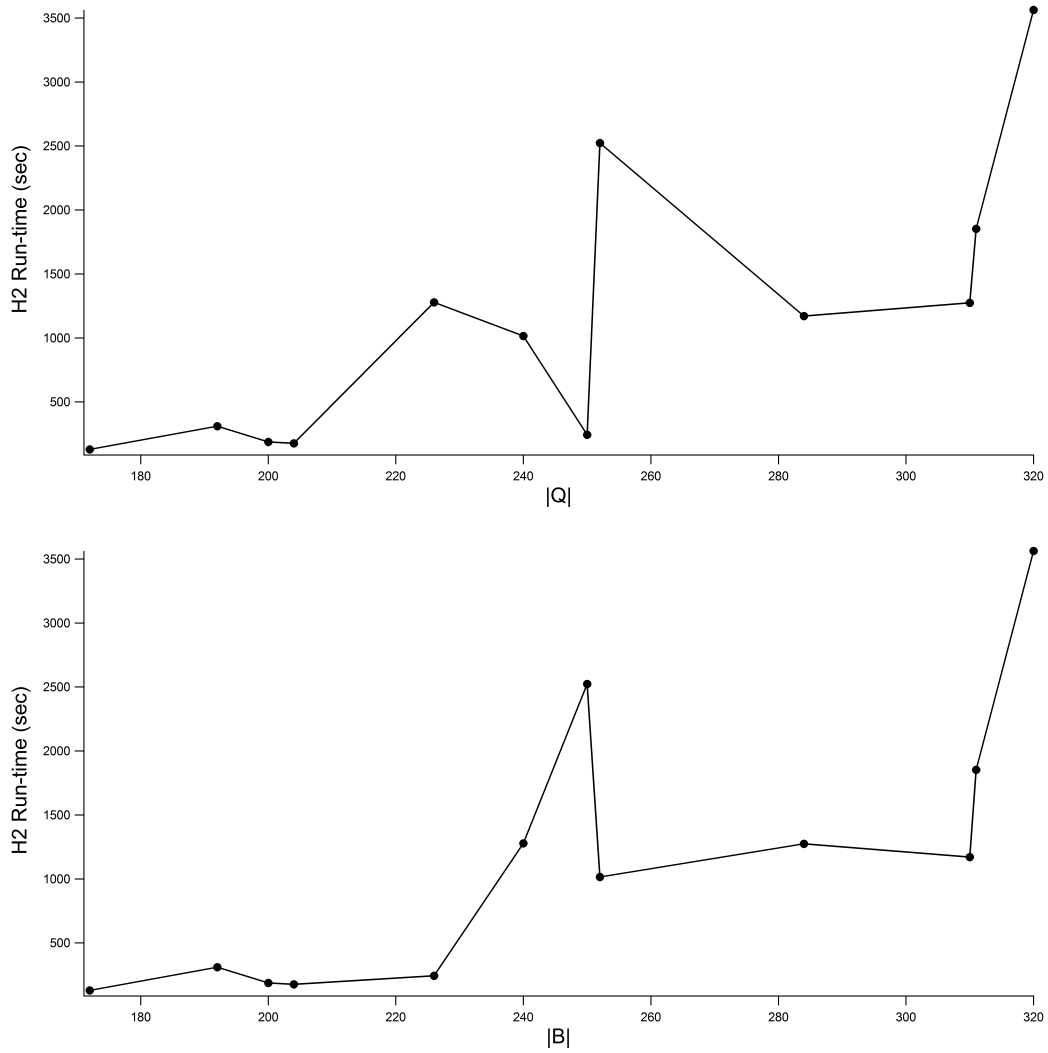


Figure 7.2: H2 for ARCP run-time analysis

very tight gap between the results of the two solution methods. This implies that our heuristic algorithms are doing well in the routing aspect.

7.3 PC-ARCP Results

Here, we present the test results for PC-ARCP on the same 32 instances that we used for ARCP. The performance of the heuristics are compared to the solutions found by solving the MIP Model for PC-ARCP, and in the cases where Cplex could not find the optimal solution in the given time limit, the heuristics are compared to the lower

bound found by Cplex in the given run-time limit. Cplex optimality gap is calculated as follows:

$$Gap_{LB}(\%) = \frac{|X_{LB} - X_{UB}|}{X_{UB}}.100\ \% \tag{7.3}$$

where X_{UB} is the upper bound given by Cplex and X_{LB} is the lower bound.

The percentage gaps for the second and fourth column of Tables 7.9 and 7.10 are calculated as follows:

$$Gap_{H}(\%) = \frac{|X_{H} - X_{LB}|}{X_{LB}}.100\ \% \tag{7.4}$$

where X_{LB} is the best integer solution known for the problem which is given by the Cplex lower bound in the given run-time limit.

In Table 7.9, the results of solving H1, H2 and the MIP Model for PC-ARCP for the Istanbul data are presented. Parameter t_{max} is assigned after solving H1 for ARCP to see how long it takes to reconnect the whole network. Then, t_{max} is calculated by dividing the tour length by two. As we see, the optimal solution was found only in one of the instances by Cplex and in others, the elapsed time shows 10800 seconds meaning that the optimal solutions were not found.

According to Table 7.9 results, the average percentage gap of H1 and H2 is 1.3% and 1.4% with respect to Cplex lower bound, respectively. However, their run times are much less than Cplex. In each of the algorithms' results, it is easily seen that by increasing $|B|$ and $|Q|$, the solution qualities worsen and run-times become higher. In most instances, the three algorithms could not find the same total prize; except L3, L4 and M1 (25 percent of the instances). In the other 75% of the instances, Cplex found the superior solution in terms of prizes collected during t_{max}.

In Table 7.10, the results for the RPP data is presented with random (b_{ij})'s (same settings as in Table 7.7). Parameter t_{max} is calculated as described for Table 7.9. The table is dedicated to the results of heuristics H1, H2 and the MIP Model for PC-ARCP solved by Cplex. As we see, the optimal (or very close to optimal) solutions were found in 14 instances out of 20 by Cplex.

According to Table 7.10 results, H1 and H2 found the optimum in 12 instances.

Table 7.9: Percentage gaps of PC-ARCP for Istanbul network instances

Instance	**Gap$_{H1}$ (%)**	**H1 time (sec)**	**Gap$_{H2}$ (%)**	**H2 time (sec)**	**Gap$_{LB}$ (%)**	**Cplex time (sec)**
L1	0.01	33	0.20	226	0.00	247
L2	0.01	37	0.47	255	0.00	10800
L3	0.00	50	0.00	412	0.21	10800
L4	0.00	76	0.00	332	0.00	10800
M1	0.00	640	0.00	1678	0.00	10800
M2	0.56	388	0.56	672	0.00	10800
M3	0.01	605	0.01	1262	1.08	10800
M4	1.62	252	1.03	1635	0.00	10800
H1	1.29	598	2.68	1685	2.58	10800
H2	6.99	171	0.47	738	0.01	10800
H3	3.17	273	6.49	2408	0.64	10800
H4	1.65	601	5.28	2752	0.02	10800
Average	1.28	310	1.43	1171	0.38	9920

The average percentage gap for H1 is 0.2% and for H2, it is 0.1% and the run times are much lower than the Cplex. In most cases, the three algorithms find the same total prize, except in I22, dv2, dv4 and B451 (20 % of the instances), for which Cplex found a superior solution in terms of prizes collected.

In Table 7.11, a brief performance summary of the three solution methods used for ARCP and PC-ARCP is presented. Average percentage gaps and run-times and maximum percentage gaps and run-times over all Istanbul network and RPP instances are presented for each algorithm. According to this table, PC-ARCP is solved in longer time than ARCP but results in higher-quality solutions. Algorithm H2 for PC-ARCP has higher average percentage gap than that of H1 which proves that changing the R set throughout the search does not necessarily result in better solutions, yet it may lead to sub-optimality. Algorithm H2 has higher average and maximum run-times than H1 and our computational tests showed that its time complexity rises with increasing $|B|$ since the choices to substitute in the R-set are more. Moreover, we observed that Cplex gives better-quality solutions but has prohibitively long run-times and its time complexity depends highly on $|V|$.

From the results presented in this section, we can claim that the performance of

Table 7.10: Percentage gaps of PC-ARCP for RPP instances

Instance	**Gap$_{H1}$ (%)**	**H1 time (sec)**	**Gap$_{H2}$ (%)**	**H2 time (sec)**	**Gap$_{LB}$ (%)**	**Cplex time (sec)**
I20	0	197	0	170	0	25
I21	0	28	0	36	0	23
I22	1	3	1.55	4	0	87
I23	0	10	0	15	0	99
I24	0	2	0	3	0	339
ALBA31	0	2	0	4	0	10
ALBA52	0	266	0	394	1.72	10800
ALBA33	0	2	0	4	0	13
ALBA34	0	3	0	6	0	158
ALBA35	0	5	0	13	0	4
MADR31	0	4	0	17	0	10800
MADR32	0	21	0	149	1.04	10800
MADR55	0	959	0	1372	0.03	10800
MADR34	0	9	0	28	0	10800
MADR35	0	2	0	4	0	837
dv1	0	111	0	404	0	1503
dv2	1	231	0.33	915	0	2402
dv3	0	593	0	1989	0.22	14400
dv4	0.98	1034	0.33	3725	0.06	14400
B451	1	348	0.15	4838	0.89	14400
Average	0.20	191	0.12	704	0.20	5135

H1 is preferred since it finds solutions with almost the same quality and significantly much faster. Hence, to be able to run the algorithm repetitively for many small sized instances quickly, like for different districts of a city with distinct depots, we claim that H1 outperforms H2 and Cplex. Using this heuristic algorithm can be beneficial in a decision support system devised to be used for the critical disaster response phase.

7.4 Integrality Relaxation Analysis

In this section, we show how the resulting cut from Proposition 4.1.1 improved the solution times of our proposed Model for ARCP and Model for PC-ARCP. According to the dominance property resulting from Proposition 4.1.1, we modelled and solved ARCP and PC-ARCP using binary X_{ij} variables rather than integer X_{ij}'s. In Tables 7.12, 7.13 and 7.14, we show how the solution times decreased upon cutting the solution

Table 7.11: Summary table

	ARCP			PC-ARCP		
Algorithm	H1	H2	Cplex	H1	H2	Cplex
Avg. Gap (%)	4.14	2.92	4.70	0.60	0.61	0.27
Max. Gap (%)	25.42	25.42	112.73	6.99	6.49	2.58
Avg. Time (s)	89	648	4667	236	879	6930
Max. Time (s)	898	3785	14400	1034	4838	14400

space, hence obtaining better bounds in the cases where Cplex is not able to find the optimum in the given time limit. In these tables, MBC stands for Model before Cut (that has integer X_{ij}'s) and MAC stands for Model after Cut (that has binary X_{ij}'s). Moreover, in Table 7.14, L(hour) shows the tour length of the solution and time(sec) is the elapsed time. In Tables 7.12 and 7.13, the gaps are Cplex optimality gaps reported by Cplex in the given time limit. As we see, the gaps are tighter, solution qualities are better, and the run-times are less in MAC. This analysis justifies the use of the dominance property in our modeling approach.

7.5 ARCNCP Results

In this section, we present the results of solving the MIP Model for ARCNCP on five small-sized instances (I20-I24). The optimal solution was found in all instances by Cplex in less than five seconds. The second and third columns of Table 7.15, list the S-nodes and D-nodes of each instance respectively. We present the optimal walk, node by node, in the fifth column of this table, while the optimal tour length is shown in column four. All in all, we can claim that this model is a time-efficient model that could be used practically in post-disaster situations.

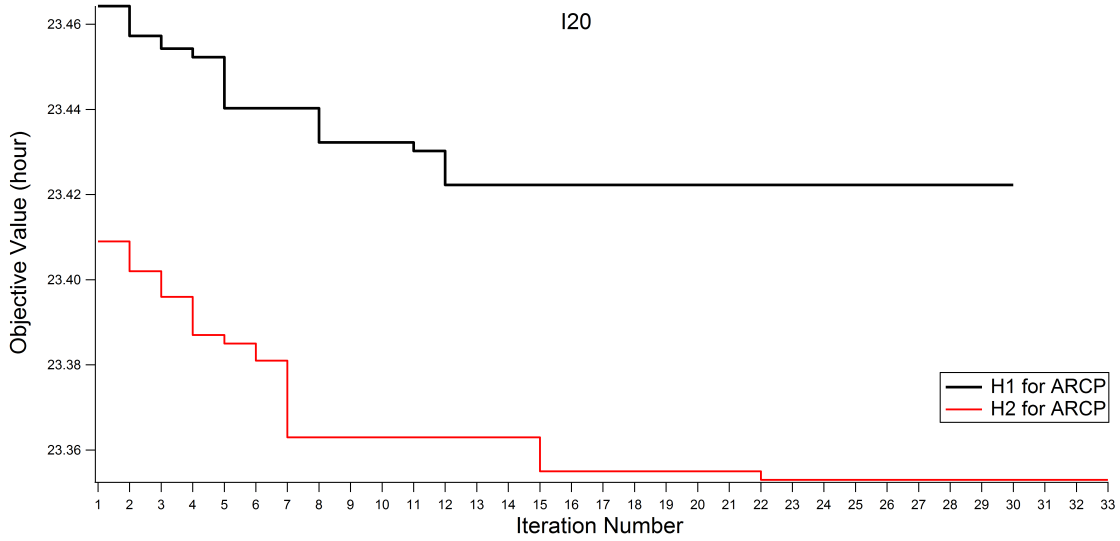
7.6 Convergence Analysis of H1 and H2

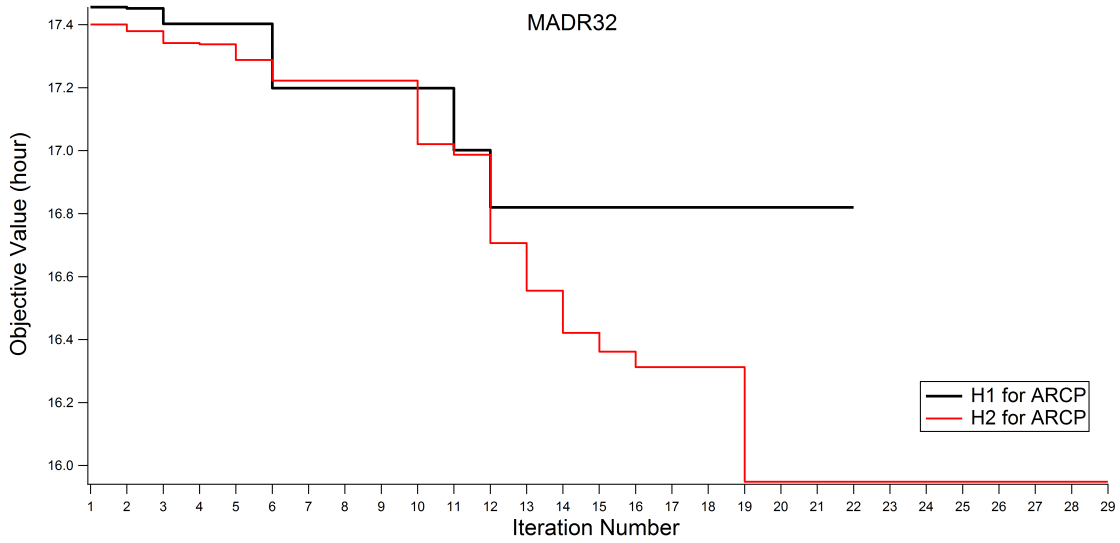
Here, we show how the algorithms H1 and H2 for ARCP converge while solving instances I20, MADR32 and M4. As we observed almost the same behavior in other

Table 7.12: Comparing MBC and MAC for ARCP on RPP instances (Random b_{ij}'s)

Instance	MBC Gap (%)	MBC time (sec)	MAC Gap (%)	MAC time (sec)
I20	16	10800	0	171
I21	15	10800	0	131
I22	22	10800	0	1601
I23	18	10800	0	74
I24	15	10800	0	24.5
ALBA31	0	2011	0	75
ALBA52	26	10800	0	6285
ALBA33	0	18	0	4.5
ALBA34	0	1705	0	62.5
ALBA35	0	322	0	6
MADR31	28	10800	8.25	10800
MADR32	38	10800	15.97	10800
MADR55	70	10800	31.16	10800
MADR34	56	10800	21.57	10800
MADR35	15	10800	0	9080
dv1	14	14400	0	3793
dv2	13	14400	0	8396
dv3	18	14400	0	4410
dv4	17	14400	1.83	14400
B451	187	10800	112.72	14400
Average Gap (%)	28.4	9562	9.57	5305

instances while solving H1 and H2 for ARCP and PC-ARCP, we just present the three plots shown in Figure 7.3 to show the overall behavior of the algorithms. In these plots, the horizontal axis represents the iteration numbers. Note that one iteration is counted when a neighborhood is searched no matter it ended in improving the current solution or not. By this analysis, we can conclude that the algorithms converge mostly to a local optimum in a certain amount of time and H1 mainly converges in less number of iterations than H2. The horizontal length of each step shows the number of neighborhoods (iterations) it took starting from the first neghiborhood, to obtain a better solution. Note that each time a better solution is found, the VND starts the search over from the first neighborhood. Since the horizontal length of most of the steps equal one, we can conclude that the first neighborhood is responsible for most of the improvements done on the objective function.





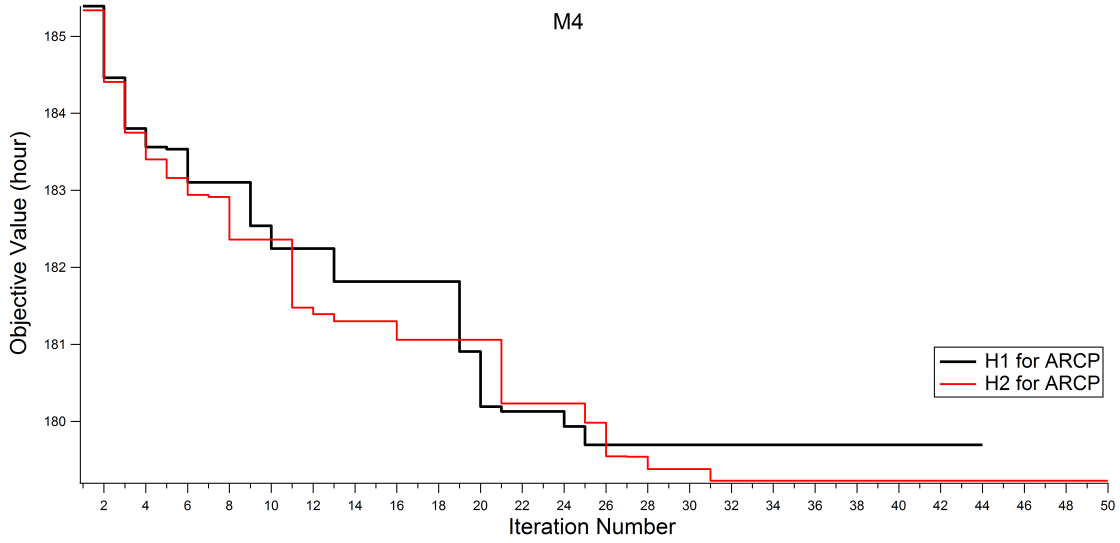


Figure 7.3: Convergence analysis examples

Table 7.13: Comparing MBC and MAC for ARCP on RPP instances ($b_{ij} = c_{ij}$)

Instance	MBC Gap (%)	MBC time (sec)	MAC Gap (%)	MAC time (sec)
I20	0	2485	0	13
I21	0	270	0	4
I22	0	2614	0	14
I23	1	10800	0	24
I24	0	180	0	8
ALBA31	0	18	0	3
ALBA52	5	10800	0	965
ALBA33	0	12	0	3
ALBA34	0	267	0	12
ALBA35	0	12	0	3
MADR31	10	10800	0	1282
MADR32	11	10800	0	6460
MADR55	2	10800	1	10800
MADR34	0	10800	0	19.35
MADR35	0	1151	0	99
dv1	2	14400	0	1770
dv2	1	14400	0	140
dv3	4	14400	0	200
dv4	3	14400	0	297
B451	72	10800	37	14400
Average Gap (%)	5.55	7010	1.9	1825

7.7 Effectiveness of the Moves

We evaluate the effectiveness of the moves in finding a better solution to ARCP and PC-ARCP with regard to their specific objective functions. These values are cumulative over three replications of algorithm H2 for all 12 Istanbul network instances. We did this analysis for ARCP and PC-ARCP separately. We did this analysis merely for H2, because H1 and H2 have the same neighborhoods, except for the first neighborhood (N_1), that H2 entails Deletion/Reconnection. The first and third columns of Table 7.16 show the neighborhood structures used in ARCP and PC-ARCP, while the second and fourth columns present the fraction of improvements done by each move over all improvements on the solutions in all runs. According to this table, the first move is by far more influencial than other moves in both ARCP and PC-ARCP.

Table 7.14: Comparing MBC and MAC for PC-ARCP on RPP instances (Random b_{ij}'s)

	MBC			MAC		
Instance	prize	L (hour)	time (sec)	prize	L (hour)	time (sec)
I20	49	1.31	10800	49	1.30	25
I21	41	0.49	1044	41	0.48	23
I22	43	0.49	3404	43	0.48	87
I23	44	0.46	10800	44	0.46	99
I24	33	0.35	59	33	0.35	339
ALBA31	112	5.4	2929	112	5.39	10
ALBA52	108	17.39	10800	108	17.34	10800
ALBA33	112	4.8	33	112	4.8	13
ALBA34	110	6.04	632	110	6.04	158
ALBA35	109	7.27	12	109	7.27	4
MADR31	194	4.94	10800	194	4.94	10800
MADR32	191	7.94	10800	191	7.67	10800
MADR55	195	29.18	10800	195	29.18	10800
MADR34	195	11.03	10800	195	10.927	10800
MADR35	194	2.67	8358	194	2.67	837
dv1	315	45.06	14400	315	45.06	1503
dv2	309	46.51	14400	309	46.51	2402
dv3	305	46.82	14400	305	45.81	14400
dv4	296	45.81	14400	297	47.99	14400
B451	457	4.38	14400	457	4.29	14400

Table 7.15: Optimal results for ARCNCP

Instance	S	D	Z* (hour)	W*
I20	3,15,42	35	0.220	1-2-3-2-1-4-39-42-39-9-46-36-35-36-11-14-15
I21	3,15,42	35	0.191	1-2-3-2-1-25-27-36-35-36-38-30-42-10-9-14-15
I22	3,15,42	35	0.238	1-2-3-2-1-10-28-40-35-34-42-31-16-15
I23	3,15,42	35	0.167	1-2-3-2-1-43-35-42-10-14-15
I24	3,15	35	0.161	1-2-3-23-22-21-35-11-13-15

7.8 Model Performance

In this section, we analyze the performance of our proposed models for ARCP and PC-ARCP. To this end, we compare the upper bounds and lower bounds of the solutions given by Cplex at the root node and the final MIP solution to see how much the Cplex improves the solutions. Second, third and fourth columns of Tables 7.17-7.20

Table 7.16: Percentage share of the moves in solution improvement

ARCP		PC-ARCP	
Moves	Share(%)	Moves	Share(%)
Combined Pairwise Edge Exchange and Deletion/Reconnection	66	Combined Pairwise Edge Exchange and Deletion/Reconnection	86
Postpone	3	Postpone	1
Drop/Add	12	Add	1.25
Block moves	19	Drop	1.25
		Drop/Add	4.5
		Block moves	6

show the lower bound, upper bound and the percentage gap of the MIP solution at the root node given by Cplex, while the fifth, sixth and seventh columns of Tables 7.17-7.20 show the lower bound, upper bound and the percentage gap of the final MIP solution given by Cplex.

Table 7.17: Comparing the gaps of the root node and the final MIP solution to the Model for ARCP on Istanbul network instances

Instance	LB_{root}	UB_{root}	$Gap_{root}(\%)$	LB_{final}	UB_{final}	$Gap_{final}(\%)$
L1	10.76	1419.32	99.24	41.95	41.95	0
L2	63.83	2015.26	96.63	92.14	93.45	1.41
L3	37.28	1627.25	97.71	60.88	60.88	0
L4	128.69	1772.02	92.74	155.94	157.26	0.84
M1	150.11	3078.32	95.12	210.13	211.53	0.66
M2	83.69	2545.71	96.7	153.44	154.87	0.93
M3	96.67	3398.83	97.16	156.21	157.62	0.90
M4	67.77	2766.96	97.55	179.22	179.22	0
H1	146.22	5757.74	97.46	316.7	318.92	0.68
H2	170.97	5290.62	96.77	280.12	304.09	8.55
H3	366.49	5174.12	92.92	500.80	504.55	0.74
H4	219.09	5610.7	96.1	292.45	292.45	0

Table 7.18: Comparing the gaps of the root node and the final MIP solution to the Model for ARCP on RPP instances

Instance	LB_{root}	UB_{root}	$Gap_{root}(\%)$	LB_{final}	UB_{final}	$Gap_{final}(\%)$
I20	22.82	74.37	69.3	23.35	23.35	0
I21	0.59	55.52	98.93	0.90	0.90	0
I22	0.72	1.49	51.4	0.95	0.95	0
I23	0.72	146.43	99.5	0.95	0.95	0
I24	0.725	0.986	26.45	0.88	0.88	0
ALBA31	8.48	835.24	98.98	11.90	11.90	0
ALBA52	26.49	50.4	47.22	36.19	36.19	0
ALBA33	9.23	15.44	40	11.06	11.06	0
ALBA34	11.11	29.52	62.35	13.62	13.62	0
ALBA35	6.02	20.96	27.06	17.52	17.52	0
MADR31	6.26	484.4	98.7	10.14	10.14	0
MADR32	10.03	3588.13	99.7	15.84	15.84	0
MADR55	262.6	2714.1	90.3	275.70	278.90	1
MADR34	253.6	2268.04	88.8	258.63	258.63	0
MADR35	4.93	469.1	98.95	6.83	6.83	0
dv1	140.2	158.22	11.38	144.85	144.85	0
dv2	99.95	1086.2	91.79	171.01	171.01	0
dv3	150.2	469.9	68.03	188.49	188.49	0
dv4	218.51	324.69	32.7	244.80	244.80	0
B451	3.518	5096.72	99.93	6.44	8.80	37

Table 7.19: Comparing the gaps of the root node and the final MIP solution to the Model for PC-ARCP on Istanbul network instances

Instance	LB_{root}	UB_{root}	$Gap_{root}(\%)$	LB_{final}	UB_{final}	$Gap_{final}(\%)$
L1	1341.35	1435.31	7.01	1419.45	1419.45	0
L2	870.32	1434.96	64.88	1431.76	1431.77	0
L3	177	1438.8	709.47	1431.61	1434.62	0.21
L4	1340.4	1425.62	6.35	1425.36	1425.39	0
M1	717.3	1431.2	99.53	1415.46	1415.53	0
M2	612.46	1432.5	133.9	1418.73	1418.78	0
M3	789.36	1433.96	81.6	1399.16	1414.47	1.08
M4	154.008	1429.17	829.9	1409.60	1409.60	0
H1	1121.83	1479.79	24.19	1396.42	1433.33	2.58
H2	419.2	1433.65	241.99	1413.63	1413.83	0.01
H3	1362.78	1402.95	2.86	1391.66	1400.61	0.64
H4	249.77	1481.59	83.14	1416.167	1416.51	0.02

Table 7.20: Comparing the gaps of the root node and the final MIP solution to the Model for PC-ARCP on RPP instances

Instance	LB_{root}	UB_{root}	$Gap_{root}(\%)$	LB_{final}	UB_{final}	$Gap_{final}(\%)$
I20	48.98	48.99	0.01	48.99	48.99	0.00
I21	23	44.34	92.8	41.00	41.00	0.00
I22	38.99	43.66	10.69	43.00	43.00	0.00
I23	27	47.55	76.12	44.00	44.00	0.00
I24	31.99	33.99	10.37	33.00	33.00	0.00
ALBA31	104	114.114	9.73	111.95	111.95	0.00
ALBA52	30.95	109.71	262.71	107.83	109.71	1.72
ALBA33	99	113.92	15.11	111.95	111.95	0.00
ALBA34	105.93	112.14	6.79	109.94	109.94	0.00
ALBA35	93.97	108.96	15.94	108.93	108.93	0.00
MADR31	180	195.07	8.37	193.95	193.95	0.00
MADR32	84	195.32	132.59	190.92	192.92	1.04
MADR55	82	194.87	137.63	194.71	194.77	0.03
MADR34	176	194.93	10.76	194.89	194.89	0.00
MADR35	191	194.97	2.08	193.97	193.97	0.00
dv1	293	315.05	19.53	314.55	314.55	0.00
dv2	279	311.97	11.82	308.53	308.53	0.00
dv3	270	305.71	11.68	304.54	305.20	0.22
dv4	259	299.13	13.41	296.52	296.71	0.06
B451	389	464.96	19.53	456.96	461.08	0.89

Chapter 8

ON THE USE OF THE ADDRESSED PROBLEMS IN POST-DISASTER DECISION SUPPORT

The three problems discussed in this thesis can be used in different disaster situations. PC-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. On the other hand, ARCP can be employed when fairness policy is a must; meaning that we have to restore connectivity of all components while ARCNCP is useful when the connectivity of the whole network is not a must and/or there are pre-specified supply points.

PC-ARCP can be utilized in two different frameworks one of which is more reactive to the evolving disaster situation. In this reactive approach, we make use of the updated information in almost real-time and run the PC-ARCP at distinct time points with updated input data. The second approach is that we run PC-ARCP back-to-back with the information we have at the beginning. In this case, we set a small t_{max} and solve the problem many times back-to-back.

In this chapter, we compare ARCP and PC-ARCP and the solution methods with regard to the latency criterion. Latency is the time at which the component is connected and in disaster situations, the earlier we make the connectivity, the more victims are reached and serviced in that component. Although, the aim of this work is not minimizing the latency, we want to show that the second problem exhibits a better performance in this aspect.

8.1 Latency Analysis

We analyze and compare PC-ARCP and ARCP with regard to latency criterion when reconnecting the whole network. In this analysis, we utilize PC-ARCP by running it back-to-back at distinct time points until we get the whole network connected and ARCP by running it once. Note that for running PC-ARCP back-to-back, we specified $t_{max} = t_{ARCP}/10$, where t_{ARCP} is the time to reconnect the whole network obtained from H1 algorithm. We run PC-ARCP 10 times back-to-back, so the run-time for this algorithm will be almost 10 times the run-time of PC-ARCP.

For instance, during the first $t_{ARCP}/10$, prize P_1 is collected. After updating the set of blocked edges B and connected components Q, and the depot node d which will be the position of the vehicle from the previous period, we run PC-ARCP again with $t_{max} = t_{ARCP}/10$, and the total prizes collected will be $P_1 + P_2$. We continue this way until we connect the whole network.

We took 6 instances from the ones described in Table 6.1 and Table 6.2 to experiment. A data point in each chart of Figure 8.1 shows the addition of an amount of prize that is collected by the walk at a specific time. The consecutive data points in each chart are connected using lines for better presentation, although the prizes increase as a step function of the tour length. As seen in Figure 8.1, in most cases, the total prize collected using this PC-ARCP-based method at different time points is more than the one in ARCP, meaning more people wait less until they are connected. For instance, in I20, 9 victims are waiting an additional time of almost 22 hours to get connected in the solution to ARCP. Moreover, comparing the charts of the instances L4, M4 and H4 that are all from the same road network, but with different disaster scenarios, one can see that the relative performance of the two solution methods depends on the network topology, namely the position and number of blocked edges and connected components on the network.

Looking at the last data point of each method in different charts of Figure 8.1, we see that using heuristic H1, the whole network is connected in less time (tour length) using this method than in ARCP in most cases; which makes this approach

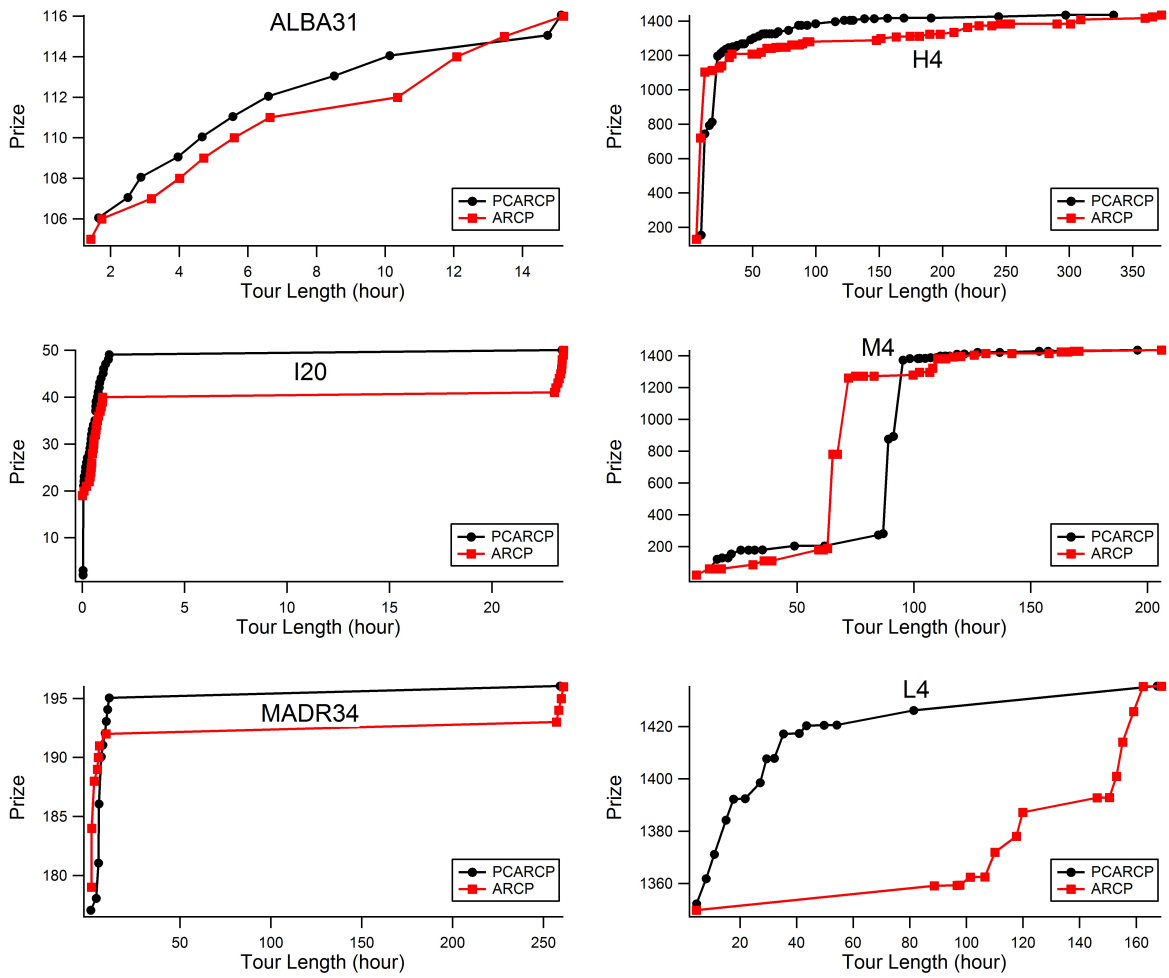


Figure 8.1: Latency analysis examples

an alternative solution method to ARCP due to its time efficiency and high solution quality.

8.2 Comparison of ARCP and PC-ARCP Solutions

If fairness is not the aim of the disaster manager, they can have more people wait less to be connected to the depot using PC-ARCP solution. We took 10 instances from the ones described in Table 6.2 and solved ARCP and PC-ARCP once by H1. In Figure 8.2, we show the analysis on two of the instances ALBA31 and DV3. The charts show the number of nodes connected to the depot over time. In each chart, the

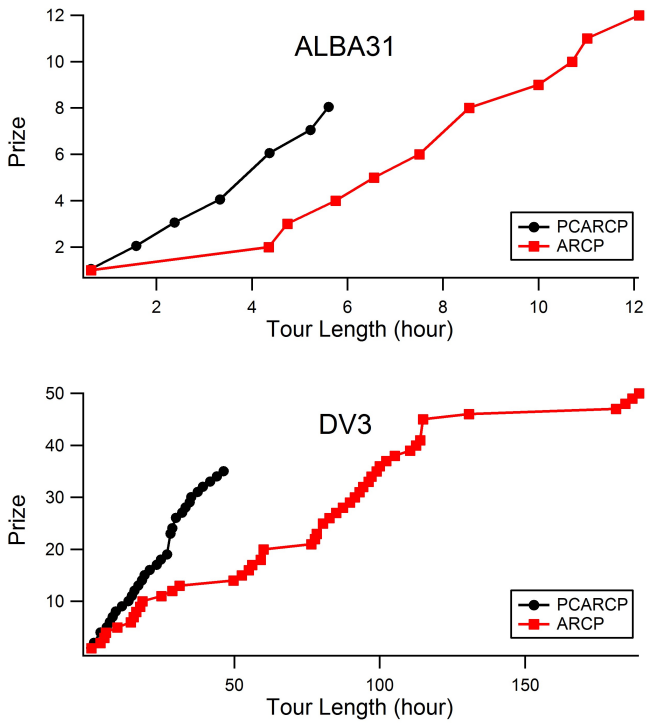


Figure 8.2: Examples to comparing ARCP and PC-ARCP solutions

progress made by PC-ARCP over time is considerably higher than ARCP. This is true because in PC-ARCP, the solution should be able to connect as many nodes as it can within t_{max}.

In Table 8.1, we present the results of solving the sample instances using algorithm H1. The percentage of nodes that are connected during t_{max} is reported after solving both ARCP and PC-ARCP for each instance. For the ARCP case, we stop the algorithm whenever the tour length reaches t_{max}. Note that the number of nodes that are inside of Q_d (the component containing the depot) is deducted since they are already connected to the depot. It is simply observed from the table that the PC-ARCP connects a total sum of more nodes than ARCP in all cases except two cases in which they connect the same number of nodes. Also, we could see that at each time point, the number of nodes connected in PC-ARCP is higher than ARCP in almost all cases showing the less total latency this solution strategy brings for a disaster manager.

Table 8.1: Percentage of nodes connected during t_{max}

Instances	ARCP (%)	PC-ARCP (%)
I20	90	98
I21	38	58
ALBA31	33	67
ALBA33	76	76
ALBA34	95	95
MADR31	50	88
MADR32	67	96
MADR34	10	95
dv2	10	71
dv3	26	70

Chapter 9

CONCLUSIONS

Within the context of the disaster response logistics, we studied three challenging arc routing problems with applications in clearing the roads and restoring the connectivity of the road network damaged by a disaster. The first problem (ARCP) aims to reconnect the whole network in shortest time. The second problem (PC-ARCP) is a prize collecting arc routing problem that aims to collect the most prize by connecting components within specified time limit. The third problem (ARCNCP) reconnects the critical supply and demand points. PC-ARCP and ARCNCP are proposed and solved for the first time.

We developed mathematical models for the three problems and two versions of the VND metaheuristic for solving the first two problems efficiently. The solution qualities of the heuristic algorithms were close to the bounds found by Cplex in the given time limit in most cases. Computational tests on the Istanbul case and instances adapted from the literature showed that our heuristic approach performs better than the exact model in terms of obtaining a good solution in short time in larger instances. In smaller instances, our proposed exact models reach the optimum almost instantly. Of the two versions of the VND, H1 results in shorter run times but close enough objective values. So, we find it practically possible to solve instances of relevant size in quasi-real time using H1. We also discuss the use of the problems in practice.

Promising extensions to our work are studying the stochastic problem which considers probabilistic unblocking times to make it more realistic, and a multi-vehicle case. Also, total latency of all components is more challenging to optimize and could be a future research direction.

Chapter 10

APPENDIX

10.1 Parameters for Generating Unblocking/Traversal Times

The notations used in this section are shown in Table 10.1. We used the parameter values given in Table 10.2. Note that in our settings, l_{ij}, n_{ij} and r_{ij} are set to 1 or 0 randomly for each edge $(i,j) \in E$. Moreover, our setting of the parameters related to the impact of disaster magnitude on the road blockage status is given in Table 10.3. We use the following to determine road unblocking and traversal times.

$$b_{ij}^1 = (\frac{d_{ij}.p_{ij}^1}{s_1}) f_{n_{ij}}.f_{r_{ij}}.f_{l_{ij}} \qquad (i,j) \in E \tag{10.1}$$

$$b_{ij}^2 = (\frac{d_{ij}.p_{ij}^2}{s_2}) f_{n_{ij}}.f_{r_{ij}}.f_{l_{ij}} \qquad (i,j) \in E \tag{10.2}$$

$$b_{ij}^3 = (\frac{d_{ij}.p_{ij}^3}{s_3}) \qquad (i,j) \in E \tag{10.3}$$

$$b_{ij} = b_{ij}^1 + b_{ij}^2 + b_{ij}^3 \qquad (i,j) \in E \tag{10.4}$$

$$c_{ij} = (\frac{d_{ij}}{v}) f_{n_{ij}}.f_{r_{ij}}.f_{l_{ij}} \qquad (i,j) \in E \tag{10.5}$$

For each edge $(i,j) \in E$, we determine its condition ($k = 1, 2, 3$) according to random percentages p_{ij}^k that are generated between 0 and P_k by a uniform distribution.

Example Let us say an earthquake of moderate magnitude has occurred. We know that edge (i,j) has length of 1.5 kilometers ($d_{ij} = 1.5$), it is located in a residential area ($r_{ij} = 1$ and $f_{r_{ij}} = 1.1$) and the corresponding street is narrow ($n_{ij} = 1$ and $f_{n_{ij}} = 1.1$). Suppose p_{ij}^k turned out to be such that 20 % of this edge is under type 1 debris and 20 % is under type 2 debris and 60 % is in normal condition ($p_{ij}^1 = p_{ij}^2 = 20\%$ and $p_{ij}^3 = 0\%$). Moreover, there is no flooding or liquefaction on this street ($l_{ij} = 0$

Table 10.1: Notations used in description of parameters and factors

v	The speed of the vehicle in normal condition.
d_{ij}	The length of the edge (i, j).
b_{ij}^k	The time it takes to clear the debris in condition $k \in \{1, 2, 3\}$ from edge (i, j).
p_{ij}^k	The percentage of edge (i, j) under condition $k \in \{1, 2, 3\}$.
P_k	The probability that an edge is under condition $k \in \{1, 2, 3\}$.
s_k	The speed of recovering an edge from condition $k \in \{1, 2, 3\}$.
n_{ij}	Parameter which equals 1 if edge (i, j) is narrow and 0 otherwise.
r_{ij}	Parameter which equals 1 if edge (i, j) is in residential areas and 0 otherwise.
l_{ij}	Parameter which equals 1 if edge (i, j) is under liquefaction or flooding and 0 otherwise.
$f_{n_{ij}}$	The factor of narrowness of edge (i, j).
$f_{r_{ij}}$	The factor of being in residential urban areas of edge (i, j).
$f_{l_{ij}}$	The factor of flooding of edge (i, j).

Table 10.2: Factors related to b_{ij} estimation

$s_1 = 0.050$	$f_{l_{ij}}^1 = 1.2$, if $l_{ij} = 1$; and $f_{l_{ij}}^1 = 1$, otherwise
$s_2 = 0.015$	$f_{n_{ij}}^1 = 1.1$, if $n_{ij} = 1$; and $f_{n_{ij}}^1 = 1$, otherwise
$s_3 = 0.0001$	$f_{r_{ij}}^1 = 1.1$, if $r_{ij} = 1$; and $f_{r_{ij}}^1 = 1$, otherwise
$v = 20$	

Table 10.3: Estimated P_k's in different disaster magnitudes

Disaster Magnitude	$P_1(\%)$	$P_2(\%)$	$P_3(\%)$
Low	80	15	5
Moderate	50	40	10
High	30	50	20

and $f_{l_{ij}} = 1$). According to Equation (10.5), $c_{ij} = 0.09$ and Equations (10.1)-(10.4), $b_{ij} = 7.27 + 24.2 = 31.47$.

10.2 Probabilities of Acceptance for Algorithm H2

In Equation (10.6) below, we can see the formula for calculating the probabilities of acceptance for each blocked edge in the cut-set in H2, where $\Delta_{ij} = a$ is the point where the linear function turns into exponential function. An example of this function is plotted in Figure 10.1 for $a = 1.1$, $P_{acc}(a) = 0.56$ and $\gamma = 2$. Finally, in (10.7), σ is

the standard deviation of unblocking and traversal times of different blocked edges in the cut-set. This relation ensures having higher probabilities of acceptance for higher-cost/time edges if there is a higher standard deviation. Note that Δ_{ij}, γ and $\delta(C)$ were defined earlier in Section 5.3.1.

$$P_{acc}(\Delta_{ij}) = \begin{cases} 1 + (\frac{\Delta_{ij}+\gamma}{a+\gamma})(P_{acc}(a) - 1), & \text{if } -\gamma < \Delta_{ij} \leq a \\ \frac{1}{\Delta_{ij}^6}, & \text{if } \Delta_{ij} \geq a \end{cases} \qquad (i,j) \in \delta(C) \quad (10.6)$$

$$a = Max\{\frac{\sigma}{3}, 1.1\} \qquad (10.7)$$

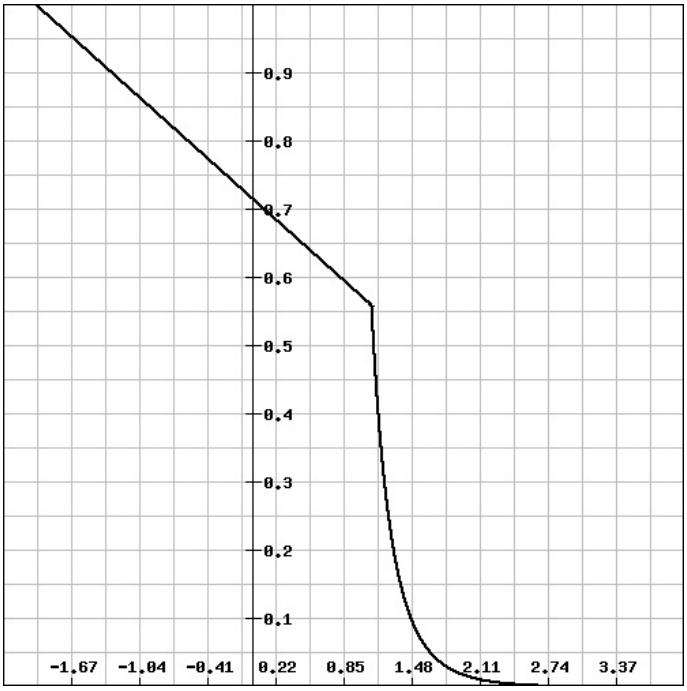


Figure 10.1: Probability of acceptance plot w.r.t. Δ

10.3 Seismic Zones of Istanbul

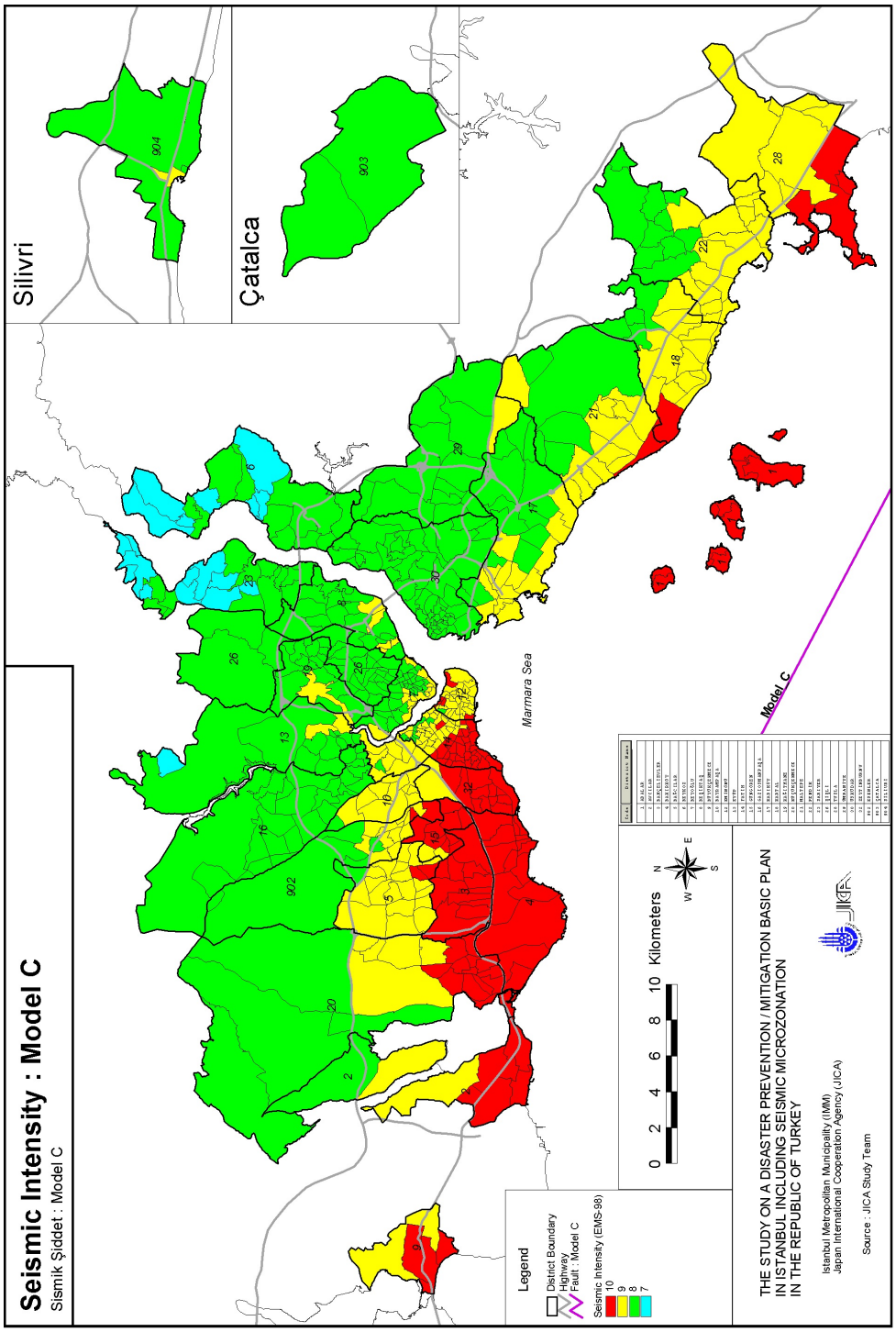


Figure 10.2: Seismic zone classification of Istanbul in JICA report (Model C)

BIBLIOGRAPHY

[1] V. Akbari and F. S. Salman. A model-based heuristic to the min max k-arc routing for connectivity problem. In *Proceedings of 4th Student Conference on Operational Research*, volume 37, pages 76–88. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014.

[2] K. Akoudad and F. Jawab. Recent survey on bases routing problems: Cpp, rpp and carp. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications, 2013.

[3] A. S. Alfa and D. Q. Liu. Postman routing problem in a hierarchical network. *Engineering Optimization*, 14(2):127–138, 1988.

[4] A. M. Anaya-Arenas, J. Renaud, and A. Ruiz. Relief distribution networks: a systematic review. *Annals of Operations Research*, 223:53–79, 2014.

[5] J. Aráoz, E. Fernández, and C. Franquesa. The clustered prize-collecting arc routing problem. *Transportation Science*, 43(3):287–300, 2009.

[6] J. Aráoz, E. Fernández, and O. Meza. Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3):886–896, 2009.

[7] J. Aráoz, E. Fernández, and C. Zoltan. Privatized rural postman problems. *Computers and Operations Research*, 33(12):3432–3449, 2006.

[8] C. Archetti, M. G. Speranza, Á. Corberán, J. M. Sanchis, and I. Plana. The team orienteering arc routing problem. *Transportation Science*, 48(3):442–457, 2013.

[9] A. A. Assad, W.-L. Pearn, and B. L. Golden. The capacitated chinese postman problem: Lower bounds and solvable cases. *American Journal of Mathematical and Management Sciences*, 7(1-2):63–88, 1987.

[10] E. Balas. The prize collecting traveling salesman problem and its applications. In *The traveling salesman problem and its variations*, pages 663–695. Springer, 2007.

[11] J. Belenguer and E. Benavent. Polyhedral results on the capacitated arc routing problem. *Unpublished Manuscript*, 1992.

[12] D. Black, R. Eglese, and S. Wøhlk. The time-dependent prize-collecting arc routing problem. *Computers and Operations Research*, 40(2):526–535, 2013.

[13] J. Brimberg, P. Hansen, N. Mladenovic, and E. D. Taillard. Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operations Research*, 48(3):444–460, 2000.

[14] P. Brucker. The chinese postman problem for mixed graphs. In *Graphtheoretic Concepts in Computer Science*, pages 354–366. Springer, 1981.

[15] I. K. Busch. *Vehicle routing on acyclic networks*. PhD thesis, Johns Hopkins University, 1991.

[16] A. M. Campbell, T. J. Lowe, and L. Zhang. Upgrading arcs to minimize the maximum travel time in a network. *Networks*, 47(2):72–80, 2006.

[17] M. Çelik, Ö. Ergun, and P. Keskinocak. The post-disaster debris clearance problem under incomplete information. *Operations Research*, 63(1):65–85, 2015.

[18] S. E. Chang and N. Nojima. Measuring post-disaster transportation system performance: the 1995 kobe earthquake in comparative perspective. *Transportation Research Part A: Policy and Practice*, 35(6):475–494, 2001.

[19] I.-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

[20] Y.-W. Chen and G.-H. Tzeng. A fuzzy multi-objective model for reconstructing the post-quake road-network by genetic algorithm. *International Journal of Fuzzy Systems*, 1(2):85–95, 1999.

[21] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[22] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem on a directed graph. *Mathematical Programming*, 26:155–166, 1986.

[23] C. Cook, D. A. Schoenfeld, and R. L. Wainwright. Finding rural postman tours. In *Proceedings of the 1998 ACM symposium on Applied Computing*, pages 318–326. ACM, 1998.

[24] Á. Corberán, E. Fernández, C. Franquesa, and J. M. Sanchis. The windy clustered prize-collecting arc-routing problem. *Transportation Science*, 45(3):317–334, 2011.

[25] A. Corberán and C. Prins. Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69, 2010.

[26] A. Corberán and J. Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79(1):95–114, 1994.

[27] P. F. de Córdoba, L. G. Raffi, and J. Sanchis. A heuristic algorithm based on monte carlo methods for the rural postman problem. *Computers and Operations Research*, 25(12):1097–1106, 1998.

[28] I. Demgensky, H. Noltemeier, and H.-C. Wirth. On the flow cost lowering problem. *European Journal of Operational Research*, 137(2):265–271, 2002.

[29] M. Dror, editor. *Arc routing: theory, solutions, and applications*. Kluwer Academic Publishers, 2000.

[30] M. Dror, H. Stern, and P. Trudeau. Postman tour on a graph with precedence relation on arcs. *Networks*, 17(3):283–294, 1987.

[31] P. M. Duque and K. Sörensen. A GRASP metaheuristic to improve accessibility after a disaster. *OR Spectrum*, 33(3):525–542, 2011.

[32] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part I: The chinese postman problem. *Operations Research*, 43(2):231–242, 1995.

[33] M. Erdik, K. Şeşetyan, M. Demircioğlu, U. Hancılar, and C. Zülfikar. Rapid earthquake loss assessment after damaging earthquakes. *Soil Dynamics and Earthquake Engineering*, 31(2):247–266, 2011.

[34] D. Feillet, P. Dejax, and M. Gendreau. The profitable arc tour problem: solution with a branch-and-price algorithm. *Transportation Science*, 39(4):539–552, 2005.

[35] C.-M. Feng and T.-C. Wang. Highway emergency rehabilitation scheduling in post-earthquake 72 hours. *Journal of the 5th Eastern Asia Society for Transportation Studies*, 5:3276–3285, 2003.

[36] E. Fernández, O. Meza, R. Garfinkel, and M. Ortega. On the undirected rural postman problem: Tight bounds based on a new formulation. *Operations Research*, 51(2):281–291, 2003.

[37] F. Fiedrich, F. Gehbauer, and U. Rickers. Optimized resource allocation for emergency response after earthquake disasters. *Safety Science*, 35(1):41–57, 2000.

[38] G. N. Frederickson. Approximation algorithms for some postman problems. *Journal of the ACM (JACM)*, 26(3):538–554, 1979.

[39] H. Furuta, K. Ishibashi, K. Nakatsu, and S. Hotta. Optimal restoration scheduling of damaged networks under uncertain environment by using improved genetic algorithm. *Tsinghua Science and Technology*, 13:400–405, 2008.

[40] R. S. Garfinkel and I. R. Webb. On crossings, the crossing postman problem, and the rural postman problem. *Networks*, 34(3):173–180, 1999.

[41] G. Ghiani, D. Laganà, and R. Musmanno. A constructive heuristic for the undirected rural postman problem. *Computers and Operations Research*, 33(12):3450–3457, 2006.

[42] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87(3):467–481, 2000.

[43] B. L. Golden and R. T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[44] M. Grötschel and Z. Win. A cutting plane algorithm for the windy postman problem. *Mathematical Programming*, 55(1-3):339–358, 1992.

[45] G. Groves and J. Van Vuuren. Efficient heuristics for the rural postman problem. *ORiON: The Journal of ORSSA*, 21(1):33–51, 2005.

[46] A. Hertz. *Graph theory, Combinatorics and algorithms: Operations research/-computer science interfaces series*, chapter Recent trends in arc routing. M.C. Golumbic and I.B.A Hartman, 2005.

[47] A. Hertz, G. Laporte, and P. N. Hugo. Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing*, 11(1):53–62, 1999.

[48] K. Holmberg. Heuristics for the rural postman problem. *Computers and Operations Research*, 37(5):981–990, 2010.

[49] JICA. 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, 2002.

[50] A. N. Kibar. Logistics planning for restoration of network connectivity after a disaster. Master's thesis, Koc University, September 2013.

[51] D. Lagana, G. Laporte, F. Mari, R. Musmanno, and O. Pisacane. An ant colony optimization metaheuristic for the undirected rural postman problem. Technical report, Les Cahiers du GERAD, 2007. G-2007-106.

[52] X. Le Pichon, A. Şengör, E. Demirbağ, C. Rangin, C. Imren, R. Armijo, N. Görür, N. Çağatay, B. M. De Lepinay, B. Meyer, et al. The active main marmara fault. *Earth and Planetary Science Letters*, 192(4):595–616, 2001.

[53] J. K. Lenstra and A. Kan. On general routing problems. *Networks*, 6(3):273–280, 1976.

[54] A. N. Letchford. New inequalities for the general routing problem. *European Journal of Operational Research*, 96(2):317–322, 1997.

[55] F. Liberatore, M. T. Ortuño, G. Tirado, B. Vitoriano, and M. P. Scaparra. 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, 2014.

[56] E. Lucas. *Recreations Mathematiques*. Gauthier-Villares, Paris, 1891.

[57] F. J. Z. Martinez. *Postman Problems on Mixed Graphs*. PhD thesis, University of Waterloo, 2003.

[58] E. Minieka. The chinese postman problem for mixed networks. *Management Science*, 25(7):643–648, 1979.

[59] C. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

[60] L. Özdamar, D. T. Aksu, and B. Ergüneş. Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, 48(4):249–262, 2014.

[61] L. Özdamar and M. A. Ertem. Models, solutions and enabling technologies in humanitarian logistics. *European Journal of Operational Research*, 244(1):55–65, 2015.

[62] G. Panoussis. Seismic reliability of lifeline networks (MIT-CE R74-57). Technical report, SDDA Report No. 15, MIT Department of Civil Engineering, 1974.

[63] C. H. Papadimitriou. On the complexity of edge traversing. *Journal of the ACM (JACM)*, 23(3):544–554, 1976.

[64] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Automata, Languages and Programming*, pages 610–620. Springer, 1989.

[65] W. L. Pearn and T. Wu. Algorithms for the rural postman problem. *Computers and Operations Research*, 22(8):819–828, 1995.

[66] M. L. Perez-Delgado. A solution to the rural postman problem based on artificial ant colonies. In *Current Topics in Artificial Intelligence*, pages 220–228. Springer, 2007.

[67] N. Perrier, A. Langevin, and J. F. Campbell. A survey of models and algorithms for winter road maintenance. part I: system design for spreading and plowing. *Computers and Operations Research*, 33(1):209–238, 2006.

[68] N. Perrier, A. Langevin, and J. F. Campbell. A survey of models and algorithms for winter road maintenance. part II: system design for snow disposal. *Computers and Operations Research*, 33(1):239–262, 2006.

[69] N. Perrier, A. Langevin, and J. F. Campbell. A survey of models and algorithms for winter road maintenance. part III: Vehicle routing and depot location for spreading. *Computers and Operations Research*, 34(1):211–257, 2007.

[70] N. Perrier, A. Langevin, and J. F. Campbell. A survey of models and algorithms for winter road maintenance. part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers and Operations Research*, 34(1):258–294, 2007.

[71] B. Raghavachari and J. Veerasamy. 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, 1999.

[72] A. M. Rodrigues and J. S. Ferreira. Cutting path as a rural postman problem: solutions by memetic algorithms. *International Journal of Combinatorial Optimization Problems and Informatics*, 3(1):31–46, 2012.

[73] H. Şahin, O. E. Karaşan, and B. Y. Kara. On debris removal during the response phase. In *International Network Optimization Conference*, 2013.

[74] M. A. Salazar-Aguilar, A. Langevin, and G. Laporte. Synchronized arc routing for snow plowing operations. *Computers and Operations Research*, 39(7):1432–1440, 2012.

[75] M. P. Scaparra and R. L. Church. A GRASP and path relinking heuristic for rural road network development. *Journal of Heuristics*, 11(1):89–108, 2005.

[76] K. Stilp, O. Ergun, P. Keskinocak, A. Carbajal, and M. Villarreal. Management of debris operations. In *Health and Logistics Conference Poster Session: Georgia Tech*. Supply Chain and Logistics Institute Center for Humanitarian Logistics, 2011.

[77] M. Tagmouti, M. Gendreau, and J.-Y. Potvin. Arc routing problems with time-dependent service costs. *European Journal of Operational Research*, 181(1):30–39, 2007.

[78] D. Tuzun Aksu and L. Ozdamar. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61:56–67, 2014.

[79] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[80] K. Viswanath and S. Peeta. The multicommodity maximal covering network design problem. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 505–510. IEEE, 2002.

[81] Z. Win. *Contributions to Routing Problems*. PhD thesis, University of Augsburg, 1987.

[82] S. Yan and Y.-L. Shih. A time-space network model for work team scheduling after a major disaster. *Journal of the Chinese Institute of Engineers*, 30(1):63–75, 2007.

[83] S. Yan and Y.-L. Shih. Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers and Operations Research*, 36(6):2049–2065, 2009.

VITA

Maziar Kasaei was born in Esfarayen, Iran, on March 30, 1990. He received his B.Sc. degree in Industrial Engineering from Sharif University of Technology, Tehran, Iran, in 2012. In September 2012, he started his M.Sc. degree and worked as a teaching and research assistant at Industrial Engineering Department of Koc University, Istanbul, Turkey.