SOLUTION METHODOLOGIES FOR DEBRIS REMOVAL DURING DISASTER RESPONSE PHASE

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Nihal Berktaş

July, 2014

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

Assoc. Prof. Bahar Yetiş Kara(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Oya Karaşan(Co-Advisor)

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

Assoc. Prof. Osman Oğuz

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

Asst. Prof. M. Alp Ertem

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

SOLUTION METHODOLOGIES FOR DEBRIS REMOVAL DURING DISASTER RESPONSE PHASE

Nihal Berktaş
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Bahar Yetiş Kara
Co-Supervisor: Assoc. Prof. Oya Karaşan
July, 2014

During the disaster response phase of the emergency relief, the aim is to reduce loss of human life by reaching disaster affected areas with relief items as soon as possible. Debris caused by the disaster blocks the roads and prevents emergency aid teams to access the disaster affected regions. Deciding which roads to clean in order to transport relief items is crucial to diminish the negative impact of a disaster on human health. Despite the significance of the problem during response, in the literature debris removal is mostly studied in recovery or reconstruction phases of a disaster. The aim of this study is providing solution methodologies for debris removal problem in response phase. In particular, debris removal activities on certain blocked arcs have to be scheduled in order to reach a set of critical nodes such as schools and hospitals. Two mathematical models are developed with different objectives. The first model aims to minimize the total time spent to reach all critical nodes whereas the second minimizes weighted sum of visiting times where weights indicate the priorities of critical nodes. Since obtaining solutions quickly is important in the early post-disaster, heuristic algorithms are also proposed. Two data sets belonging to Kartal and Bakırköy districts of İstanbul are used to test the mathematical models and heuristics.

Keywords: Debris management, debris removal, relief transportation, node routing.

ÖZET

AFET MÜDAHALE SAFHASINDA ENKAZ YÖNETİMİ İÇİN ÇÖZÜM YÖNTEMLERİ

Nihal Berktaş
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Bahar Yetiş Kara
Eş-Tez Yöneticisi: Doç. Dr. Oya Karaşan
Temmuz, 2014

Afetten etkilenen bölgelere yardım ekibinin ve acil yardım malzemelerinin ulaştırılması afet lojistiğinin en önemli safhalarından birini oluşturmaktadır. Afetin sebep olduğu enkaz, afetzedelere ulaşımı zorlaştırarak barınma, beslenme ve sağlık hizmetlerini geciktirmekte, can kayıplarını artırmaktadır. Literatürde çoğunlukla afet yönetiminin iyileştirme ve yeniden inşa safhasında çalışılmış enkaz kaldırma problemi, bu çalışmada müdahale safhasında incelenmiştir. Afet sonrasında popülâsyonu ve önemi yüksek olan hastane, okul gibi kritik noktalara en kısa sürede ulaşılmasını amaçlayan ve temizlenecek ayrıtlara karar vererek kritik noktalar arasında rota oluşturan matematiksel modeller geliştirilmiştir. Geliştirilen ilk modelde amaç fonksiyonu en son ulaşılan kritik noktanın varış zamanını enazlamaktır. Kritik noktalara ağırlıklar atanarak, ağırlıklı toplam varış zamanını enazlamayı amaçlayan ikinci bir model geliştirilmiştir. Bu modellerin testinde Kartal ve Bakırköy ilçelerinin verileri kullanılmıştır. Afet ortamında hızlı karar almak büyük önem arz ettiğinden, kısa sürede iyi çözümler üretebilecek sezgisel yöntemler geliştirilmiş, veri setleri üzerinde test edilmiştir.

Anahtar sözcükler: Afet yönetimi, enkaz kaldırma, acil yardım ulaştırma, düğüm rotalama.

Acknowledgement

I would like to express my gratitude to my advisor Assoc. Prof. Bahar Yetiş Kara for her guidance, patience and being there whenever I needed her. I would like to thank my co-advisor Assoc. Prof. Oya Karaşan for her valuable ideas and support. It has been a grate experience to work with them.

I am grateful to Assoc. Prof. Osman Oğuz and Asst. Prof. M. Alp Ertem for accepting to read this thesis and for their valuable comments. I would like to thank Halenur Şahin and Fırat Kılcı for their support in this research.

I would like to acknowledge the financial support of The Scientific and Technological Research Council of Turkey (TUBITAK).

I would like to thank Özüm Korkmaz for her invaluable friendship and support on every subject during these last two years. I will never forget the sleepless nights we spent with her and Nil Karacaoğlu who made the hard times bearable with her laughter. I would like to thank my dearest friend Merve Meraklı for her love and support. I am grateful to my friends Hüseyin Gürkan and Oğuz Çetin for the patience they have shown to all of my questions for the last five years. I am grateful to Burcu Tekin for the joy she brought during our study. I would like to thank Ramez Kian, Gizem Özbaygın, Esra Koca, Ece Demirci, İrfan Mahmutoğulları, Meltem Peker, Sinan Bayraktar for their friendship and support during our graduate studies. I am also thankful to my homemate Bihter Dağlar for her moral support.

I am most grateful to my family; my mother and my role model Hatice Berktaş for her eternal love and support, my father İzzet Berktaş who motivates me with his spirit, my sister Seda Sezgin Berktaş for her helpful advice and encouragement, and my brother İhsan Berktaş although he will be judgmental about the language of this thesis.

Last but not least, I am deeply grateful to Mesut Kaya for his love, patience and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Problem Definition

Disaster operations include activities that are carried out before, during and after a disaster in order to reduce loss of human life, minimize the economical damage and restore the normal state or well-being of the community. Disasters could be natural and man-made; their timing and/or place might be known beforehand. While accidents and terrorist attacks fall into the man-made category, earthquakes and hurricanes are natural disasters for which the locations are predictable. Regardless of the type of the disaster, the operations can be classified as pre-disaster and post-disaster operations. Disaster management cycle which includes these operations is mostly analyzed in four stages: mitigation, preparedness, response and recovery as in Figure 1.1. Mitigation consists of precautionary measures to avoid disaster or reduce its impact so mitigation activities take place both before and after a disaster. The purpose of preparedness activities is gaining the ability to respond and rescue when disaster strikes so it includes the activities prior to a disaster. Response is the stage where resources are utilized to reach disaster area, save lives and prevent further economical and environmental damage. This stage is more complex since it takes place just after the disaster where resources should be activated immediately to reach affected people, realize the severity of the situation and plan accordingly. Recovery involves the

post-disaster activities to return to a normal state and provide stable life to the disaster victims [4].

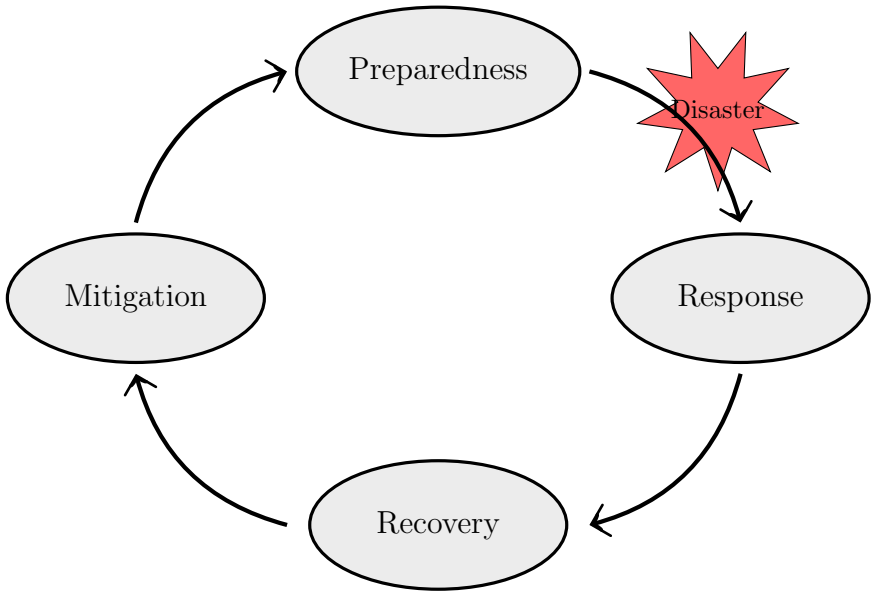


Figure 1.1: Disaster Management Cycle

Despite all kinds of precautions, disasters, especially natural ones are inevitable. Therefore planning disaster relief operations before it happens, and implementing them in early post-disaster phases are significant to diminish its impact. As stated by Wassenhove [5], huge part of disaster relief operations is about logistics. Hence to re-establish normal living conditions with the minimum loss of life and property we need to lead and carry out logistics operations effectively.

Sheu [6] defines emergency logistics as a process of planning, managing and controlling the flow of resources in order to provide relief and urgent services to affected people. Emergency logistics differs from the commercial supply chain due to the unique and extraordinary circumstances caused by disaster [7]. One of the challenges in emergency logistics is lacking capable resources to handle the situation or not being able to reach and activate them on time. During Haiti earthquake in 2010 the limited ramp space of the airport and lack of fuel prevented humanitarian flights from entering the country [8]. Furthermore the uncertainties about demand may result in wrong or excessive donations which complicate

handling and storage operations. For example, in Japan Earthquake in 2011 it is reported that too much blankets and clothing are donated. After the Joplin tornado happened in the same year, huge number of donations overwhelmed the storage and became an obstacle for distribution of actual needs [9].

Damage in communication systems and other infrastructure such as roads increases the complexity and difficulty in logistics. Again in Haiti earthquake the port was damaged and could not handle large ships so delivery of the emergency aids transported via ships were planned accordingly. Involvement of many parties to control these resources creates another challenge since they have to communicate and coordinate efficiently. This challenge was sadly proved by Hurricane Mitch in 1998 when it took weeks for The International Federation of Red Cross and Red Crescent Societies (IFRC) to coordinate and distribute the donated reliefs [10].

When governments and institutions are not able to overcome these challenges, the effects of the disaster last for a long period of time, like in example of Haiti earthquake; 98% of the debris remained after six months from the earthquake and made the transportation impossible for the most part of the capital city [11]. Debris are caused by destruction of structures and vegetation and they block the roads and prevent accessibility to disaster affected areas. There are different type of debris; construction, vegetative, hazardous waste, properties such as white goods, vehicles etc. [12]. Hence debris differs from the normal waste in terms of content and amount as Hurricane Katrina proved it by producing more than fifty times the annual amount of daily solid waste in the U.S. in few hours [13].

The magnitude, type and place of a disaster change the characteristics of debris. For instance debris caused by an earthquake on an urban area mostly consists of ruins of buildings where hurricane debris contains trees, part of structures such as fences and rooftops, and other properties. Some of the recent worldwide disasters which caused high volume of debris can be seen in Table 1.1.

Table 1.1: Debris Amount of Recent Disasters

Year	Event	Debris Amount
2011	Japan Earthquake and Tsunami	250 million ton [14]
2008	Wenchuan Earthquake, China	380 million ton [15]
2005	Hurricane Katrina, USA	76 million m^3 [16]
2004	Indian Ocean Tsunami	10 million m^3 (Only Indonesia) [17]
2004	Hurricane Charley, USA	14 million m^3 [18]
1999	Marmara Earthquake, Turkey	13 million ton [19]
1999	Chi-Chi Earthquake, Taiwan	20 million m^3 [20]
1995	Kobe Eartquake,Japan	15 million m^3 [21]
1995	Hyogoken-Nambu Eartquake, Japan	2 billion ton [22]

There are different operations related to debris through the disaster timeline. In the pre-disaster phase considering possible disaster scenarios volume and characteristics of debris are estimated. Based on these estimations debris collection strategy is established. Recycling and collection sites are determined in this phase and necessary equipment is obtained. Just after the disaster, debris clearance starts in order to access affected areas and transport relief. Complete collection and recycling of debris are the post-disaster operations which usually take months. Çelik et al. [23] summarize these operations in Figure 1.2.

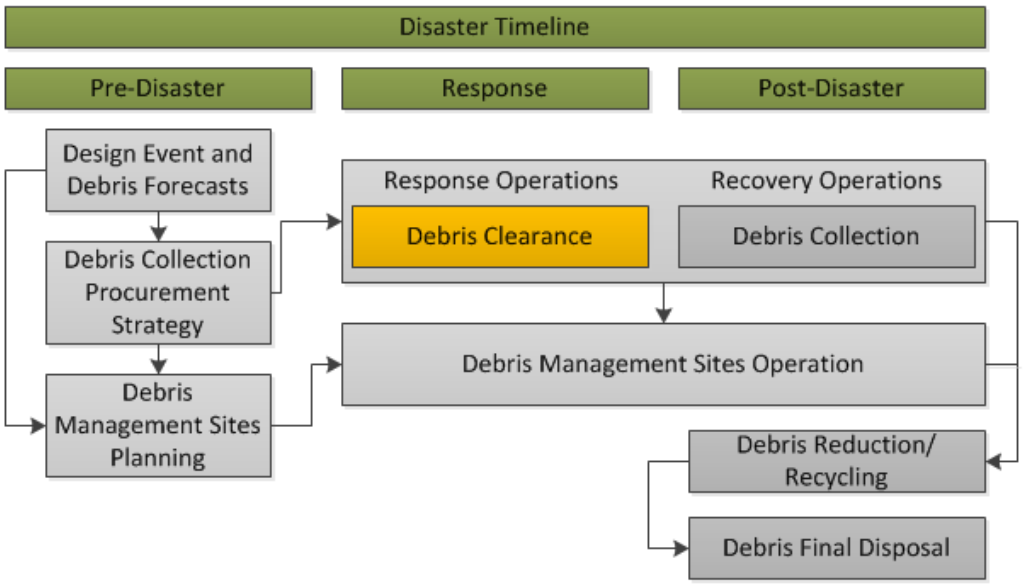


Figure 1.2: Debris Related Operations in Disaster Timeline

Clearance of debris after a disaster is a must to normalize the life of the victims. Debris have significant impact on people's health both physically and mentally. While hazardous wastes threaten victims' lives directly, living near the wreckages affects people's psychology. These issues are the concerns of complete removal and recycle of debris in the post-disaster phase. Due to the nature of the response phase, the aim and characteristics of debris related activities are different. In this phase, the main purpose is to reach disaster affected areas in order to deliver relief. By relief we mean all kind of emergency aid items such as food and medicine. Hence the debris removal operations during this phase are only performed if it is necessary to reach an area and deliver relief as soon as possible.

In this study we focus on the debris removal or clearance operations in the response phase of a disaster, more precisely an earthquake. Earthquakes are not rare events; average annual number of earthquakes occurred worldwide in the last decade is 28 [24]. As it can be seen from the seismic zone map, Turkey is an earthquake prone country. Statistics show that in every 8 months a serious earthquake occurs in Turkey [25].

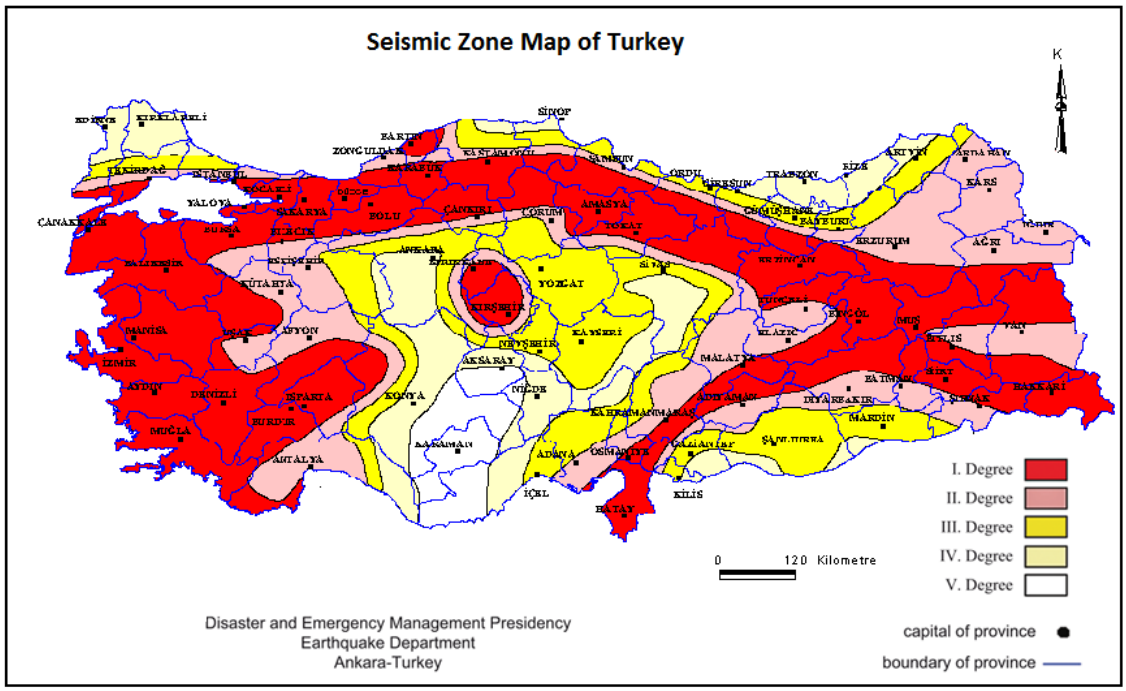


Figure 1.3: Seismic zone map of Turkey [1]

Coordination and execution of operations related to debris removal fall under responsibility of Republic of Turkey Prime Ministry Disaster and Emergency Management Presidency (T.C Afet ve Acil Durum Yönetim Başkanlığı)(AFAD). According to Turkey Disaster Intervention Plan published on May 2013 by AFAD, there are two main solution partners concerning these operations. Ministry of Transport, Maritime Affairs and Communication is responsible for providing fast and safe transportation to disaster areas by clearing debris, especially on the main roads. The duties of Ministry of Environment and Urban Planning are mostly related to the recovery phase such as debris removal after search and rescue operations, determining debris collection areas and destroying damaged buildings. There are other supportive ministries and sectors which coordinate with and provide equipments and personnel to AFAD.

Since the aim in disaster response phase is distributing relief to the people as soon as possible, the debris removal operations in this phase do not intend to clean all the blocked roads. Especially after a severe earthquake in a vulnerable area, unblocking all the road may take months. However people affected by the disaster need food, medicine, treatment, shelter etc. within minutes to fulfill basic needs. We call all of these needs as emergency aid or relief and in order to minimize the devastating effects of the disaster, these relief items must be delivered to the disaster affected people as soon as possible. For that purpose, complete clearance of the debris should be postponed to the post-disaster phase and roads should be cleaned only when it is necessary to use that road to enable accessibility a disaster affected region.

Reaching all the settlement areas of the disaster affected region in a short amount of time is not possible after a serious earthquake. Therefore we focus on a subset of them. This subset is called critical and it includes areas which are densely populated and consequently having an urgent need of relief items such as schools. Moreover, some areas are accepted as critical, such as hospitals and shelter areas, not only because they need urgent relief distribution but also they should stay open to public access. To enable transportation to these critical districts we have to decide which blocked roads to clean. Based on these concerns, *Debris Removal Problem in Response Phase* is defined as reaching a set of predetermined

critical disaster affected nodes as soon as possible by traversing arcs which may be blocked due to debris [3]. Hence to visit a critical node, we may chose to use a blocked arc but to be able to use a blocked arc we need to remove the debris on it in expense of some effort.

We assume that there is depot or supplier from which a vehicle departs and visits each critical node to deliver relief items. Once debris on a blocked arc are removed the arc remains clean. Thus, when a blocked arc is traversed for the first time, debris are removed with some effort which is measured in terms of time, and no debris removal effort is spent in the subsequent traversals of the arc. Hence the problem is constituting a travel path from supplier to critical nodes by deciding which arcs to use, which arcs to clean and visiting order of critical nodes. While constituting this path, we focus on two objectives. The first one is minimizing the total time spent to visit all critical nodes and the second is minimizing sum of weighted visiting times where the weights determine the priority relationship among the critical nodes. Therefore we have two problems and according to the objectives we refer to these problems as *Debris Removal in Response* (DRR) and *Prioritized Debris Removal in Response* (PDRR), respectively. While constructing solution methodologies we investigate these problems separately.

Our problems have a set of nodes required to be visited as in node routing problems. They also possess some characteristics of arc routing problems because of the presence of unblocked arcs. Thus they can be seen as a variant of general routing problem and defined as one by Şahin [3]. Her study provides a comprehensive search on general routing problems, focusing on arc routing in more detail. In the next chapter, we expand this research on general routing problems and present literature on relief transportation and debris removal. This literature review shows that for the disaster response phase, debris removal is an under-researched area which highlights the contribution of this study. In chapters 3 and 4, mathematical models and heuristic methodologies which are developed for the problems *DDR* and *PDRR* are explained, respectively. Computational results of these solution techniques are represented in Chapter 5. A conclusion and possible future research directions are given in Chapter 6.

Chapter 2

Literature Review

We examine the literature in three sections titled as General Routing Problems, Relief Transportation and Debris Removal. The first part aims to illustrate the similarities and differences of our problem with various types of Arc and Node Routing Problems defined in the literature. Since the main purpose of the study is to deliver relief items to disaster affected people, Relief Transportation literature is also investigated. Finally studies on Debris Removal is summarized and contribution aimed by this thesis is stated in the last section.

2.1 General Routing Problems

General Routing Problem (GRP) was first defined by Orloff as the problem of finding a minimum cost tour which passes through all required nodes and edges at least once [26]. Required nodes and edges are subset of all nodes and edges respectively. When the required node set is empty GRP reduces to Arc Routing Problem (ARP). If there is no edge required to be traversed then GRP reduces to Node Routing Problem (NRP). Different types of these problems are examined in the following subsections.

2.1.1 Arc Routing Problems

In ARP, the aim is to determine a least-cost traversal of a required subset of arcs/edges, which starts and ends at the same node under possible constraints [27]. ARP arises in a variety of areas and different types are defined and studied in the literature according to the definition of the required set, the characteristics of the network and additional constraints. The primary ARPs are Chinese postman problem (CPP), rural postman problem (RPP) and capacitated arc routing problem (CARP). In CPP all arcs/edges are required to be traversed whereas in RPP a subset of them is required. Different than CPP and RPP, there is a set of vehicles with capacities in CARP and in addition to the cost, demand or weight is defined for each required edge.

2.1.1.1 Chinese Postman Problem

CPP was first defined as finding a minimum cost tour which traverses all the arcs of a connected graph at least once. Applications of CPP include waste collection, street sweeping, and snow plowing operations.

The problem is polynomially solvable when the graph is directed or undirected. If the graph is mixed, having both arcs and edges, then the problem is NP-hard [27]. There is another variation of CPP, windy postman problem, which is defined on an undirected graph but the cost of edge is different for each travel direction [28] [29] [30]. In hierarchical postman problem priority relationships among the arcs are taken into consideration by grouping the arcs according to their priority levels. The arcs with higher priority must be serviced before the lower ones but they can re-traversed [27]. In priority constrained Chinese postman problem nodes have different priorities and the aim is to find minimum cost route which traverses all edges at least once and visits the higher priority nodes as early as possible [31].

There are other variations of CPP where there is a fixed k number of postmen performing totally k tours and each edge should be traversed by at least one

tour. If the graph is mixed and the aim is to minimize total cost by all tours the problem is called k-CPP [32]. If the objective is to minimize the longest tour then it is called min-max k-CPP [33].

Two main differences of our problem from CPP are the existence of the required node set and not being obliged to traverse all arcs.

2.1.1.2 Rural Postman Problem

RRP aims to find a least-cost traversal of required subset of edges [26]. Both directed and undirected versions of RPP is proved to be NP-hard [34]. When RPP is defined on a mixed graph and the required subset consist of directed arcs it is called stacker crane problem which is also NP-hard [32]. By defining a profit function for each edge that can be collected on the first traversal, another version of RPP is introduced. It is called privatized RRP where the purpose is to find a cycle with maximum profit and least cost and it is also NP-hard [35]. When there is a specified time until each edge should be served, the problem is called RRP with deadline classes [36].

When the required set is a subset of all edges in windy postman problem, the problem is called windy RRP and several algorithms are suggested by Benavent et al. [37]. Another variation of this problem is min-max k-vehicles windy RRP where the aim is to minimize the longest tour while maintaining a balanced tour for the vehicles [38]. RRP has applications in street sweeping, snow plowing, garbage collection, mail delivery and school bus routing.

2.1.1.3 Capacitated Arc Routing Problem

CARP aims to minimize traversal of all arcs by vehicles with same capacities and the total demand or weight of all arcs served by any vehicle cannot exceed the capacity. It is applicable to areas including winter gritting, refuse collection and police patrolling. There are many variations of CARP and review on them can be found in the study by Şahin [3].

2.1.2 Node Routing Problems

NRP contains traveling salesman problem (TSP)-like problems where there is required set of nodes to be visited. Vehicle routing is one the most famous node routing problems and it has a broad literature because of the many variants and their application areas. Multiple vehicle, time windows, pick and delivery, vehicle capacities are some of the common constraints added the standard VRP.

Our problem can be seen as a VRP with blocked arcs which can used after blockages are cleaned. Therefore, in the VRP literature we focus on the studies on blocked networks, namely Canadian traveler problem (CTP).

2.1.2.1 Canadian Traveler Problem

CTP is a kind of shortest path problem in which some edges of the graph are blocked and they are not known in advance by the traveler. It is assumed that if the blocked edges are removed from the graph, the network is still connected. In original problem each edge is blocked with some probability known by traveler however the status is not known until visiting an adjacent node. Furthermore, when an arc is blocked it remains blocked forever in the first definition of the problem [39].

There are some variants of CTP in which some of the constraints in the classical version is relaxed. For example in recoverable CTP, blocked edges adjacent to same node have the same recovery times. In the stochastic version each edge has a blockage probability whereas in the deterministic version there is limit on the total number of blocked edges. If there is a parameter k defined as the maximum number of blocked edges and they cannot be opened, then the problem is called k-CTP [40]. In CTP with sensing, the traveler can obtain information about an edge by incuring a cost. The cost can be dependent on the edge and/or the current node [41]. In repeated CTP there are multiple travelers but one cannot start his tour until the previous tour ends [42]. However in multi-agent CTP, travelers start together and they can communicate about the status of the edges [43].

One of major differences of our problem with CTP is the presence of required node set in contrast to the single node targeted in CTP. Moreover, our problem is defined under the assumption that the blocked edges are known in advance.

To the best of authors' knowledge there is no variant of general routing problem which reflects characteristics of *Debris Removal Problem in Response Phase*.

2.2 Relief Transportation/Distribution

In their review on optimization models used in emergency logistics Caunhye et al. [2] point out the difference of business and emergency logistics. They emphasize the lack of suggestions on future research directions in the related studies and also the need of focused reviews. In their review they categorize the operations as pre-disaster and post-disaster, and they illustrate the relations between operations and facilities as in the Figure 2.1.

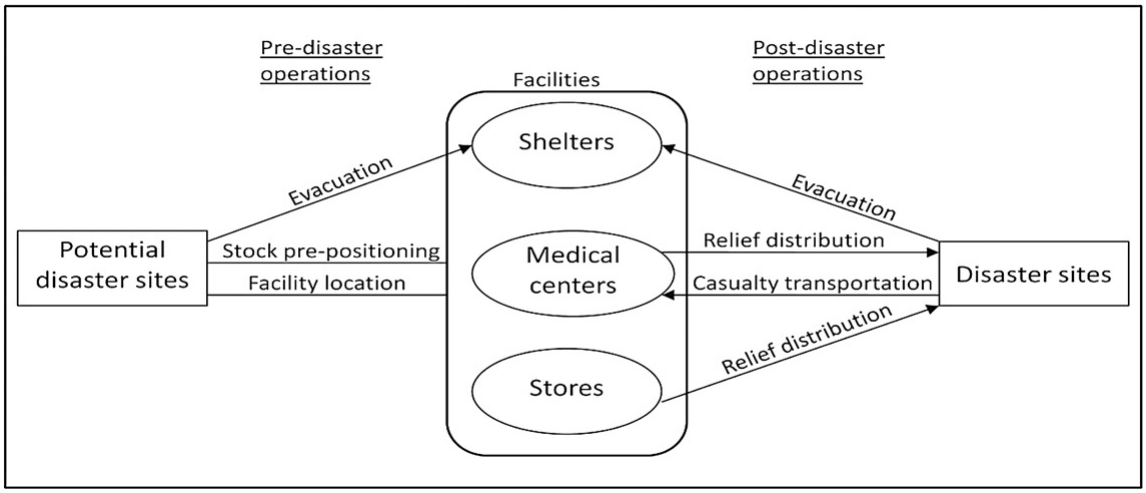


Figure 2.1: Framework for disaster operations and associated facilities and flow [2]

They classify these operations in two main categories which are facility location, and relief distribution and casualty transportation. In the literature there are location models that include operations on evacuation, stock pre-positioning and relief distribution. Relief distribution models involve resource allocation which is basically assigning tasks and equipment, and commodity flow which requires

determining the quantity of commodities and the roads used for the flow. Since our problem aims to enable access to critical disaster-affected area in order to deliver relief we focus on the studies belong to the second category which is relief distribution.

One of the studies included in this review by Caunhye et al. [2] is done by Viswanath and Peeta [44]. The aim of the study is to determine critical routes during response phase of an earthquake and they formulate a multi-commodity maximal covering network design problem with two objectives; minimizing total time traveled and maximizing total population covered with a limited budget. This budget is spent on using a link, possible damaged and can be repaired. There is a demand associated with demand centers in the network and there is a set links from which a demand center can be reached. The problem is tested on a network from southwest Indiana and solved using branch and cut algorithm.

Another multi-objective model for relief distribution is developed by Tzeng et al. [45] with objectives minimizing total cost, travel time and and maximizing the minimal satisfaction. They only consider disaster affected areas which can be accessed through current road network. The formulation has a periodic structure and satisfaction is calculated by parameters depend on the location and the commodity.

Yan and Shih [46] divide roadway network repair after a disaster into two ,i.e, long and short term. Their study is on the short term in which the time constraint is stronger and they focus on urban areas. A multi-objective, multiple commodity network flow model is developed in order to repair necessary roads and to transport relief items as soon as possible. They define two networks and two flow variables for emergency repair and relief distribution but the repair and relief operations are considered together. They define repair points as the damaged roads that cannot be bypassed by using another road. Hence one of the important assumptions of the study is that a road is only repaired to reconnect roadway network, and these repair points are known. Demand points are the areas that need relief and a minimum percentage of their demand must be satisfied. Although the existence of the demand points are similar to our problem,

the knowledge about the repair points creates a significant difference because in our problem the model decides which roads to clean.

Another review on disaster relief routing is written by Torre et al. [47]. They examined the models in terms of their objectives, characteristic of the information on supply and demand, type of the commodity, depot and vehicles. The possible damage in transportation network is handled by stochasticity in travel times in studies of Shen et al. [48], Mete and Zabinsky [49], Rawls and Turnquist [50], Van Hentercyk et al. [51].

2.3 Debris Removal

As stated in the Introduction there are different operations carried on in the different phases of disaster and we are interested in the response phase. When we examine the literature on humanitarian and emergency logistics, although there are many studies on the activities which take place in the response phase, studies including debris removal are not common. The studies on debris removal literature is mostly focused on recovery phase of the disaster management cycle and below we give some recent studies focused on debris removal.

Fetter and Rakes [52] highlight the difference of managing disaster debris with daily solid waste and point out that the disposal of debris constitutes big part of disaster costs. They state that the disaster debris cleanup operations are commonly divided in two phases. The first phase aims to clear debris to ensure access to the disaster-affected area as in our problem and the second phase includes all operations related to debris collection, separation and recycling. They mention that with a change in the disaster disposal policies by the U.S. Federal Emergency Management Agency (FEMA), the recycling of debris is encouraged and parallel to that policy, a facility location model which aims to maximize recycling with minimum cost is suggested in their study. The model decides where to locate temporary disposal and storage reduction (TDSR) facilities among a set of possible locations. TDSRs may posses different technologies and they incur fixed

and technological cost which are minimized together with cost of collecting and transporting debris. Revenue obtained from the sales of the reduced debris is also included in the objective together with or without the fixed and variable costs.

Different than the other studies, Hu and Sheu [53] incorporate psychological effects of debris. They state that studies on the waste management focuses mostly on physical health, the socio-economic and psychological impacts are paid seldom attention. They also point out that the post-disaster debris management literature lacks quantitative studies. They develop a multi-objective model which includes three conflicting costs; logistical, risk-induced and psychological. Logistical costs consist of operational costs related to transportation and recycling of debris. Risk-induced cost includes environmental risks associated with uncollected debris, storages and transportation. In psychological cost both disaster victims and people working in the recovery operations are considered. The proposed system is applied to a case study on Wenchuan Earthquake.

Pramudita et al. [54] summarize the important issues on the debris collection operations as having appropriate disposal sites, providing necessary equipment especially vehicles and transportation cost. For debris collection after disaster they suggest a model which is variant of Location-Capacitated Vehicle Routing Problem by transforming arc routing to vehicle routing. The aim is to service all required arcs, after the transformation they become nodes and the objective function minimizes total distance traveled together with opening cost of intermediate depots where vehicles unload. A matrix called access possibility is defined and it takes value one if vehicle can go from one node to another. The values of this matrix should be updated each time a required node is visited and it is referred as a dynamic constraint. The flow is defined between depot, required nodes and the shortest path distances between these nodes include travel and service costs and assumed to be known. Our problem differs in terms of the main goal which is reaching a set of nodes with a possibility of cleaning blocked arcs whereas in their study Pramudita et al. [54] aim to clean all blocked roads and in their test data all arcs are blocked. They also assume that the shortest path between two nodes are the only path that exists in the network. To solve the problem they develop an algorithm which uses the mathematical model.

Although debris removal is commonly stated among the operations in the response phase or short-term recovery, there are not many studies on these phases. For example Holguín-Veras et al. [55] mention debris removal among the operations which take place in short-term recovery or in transitional stage between response and long-term recovery [56]. However to the authors' knowledge, the only study which considers debris removal in the response or short-term recovery phase is done by Şahin [3] who also define *Debris Removal Problem in Response Phase*.

In her study three mathematical models are suggested under the assumption that the blocked arcs and the time required to clean them are known. All models aim to reach some required or critical nodes as soon as possible. The first model minimizes visiting times of critical nodes whereas the second model aims to minimize total distance traveled under a given time limit. This time constraint is included in the second model by setting an upper bound to the visiting times which correspond to the objective function of the first model. Thus, the second model is a variation of the first one. To decrease computational time, a third mathematical model, called Minimize Total Effort is introduced and analysis are performed with this model using two data sets. To suggest fast solution methodologies which provide near optimal solutions, constructive and improvement heuristics are also developed in this study [3].

To the best of our knowledge, our study is the second one which focuses on debris removal in the response phase to enable access to a set of nodes. In our problem setting, although the blocked arcs are known they are not obliged to be cleaned and the decision on which arcs to clean is made by the model. These are the most important differences of our study from the others on debris removal and road repair. Moreover this study is first one which incorporates weights of the nodes into the problem. In studies with multi-commodity models, the differences among the critical nodes or demand points are reflected in terms of their demand amount. In our study we assign weights to the critical nodes and these weights are not related to the commodities, and they directly affect the time that a critical node is reached.

Chapter 3

Model Development

Experiences in the past disasters sadly show that reaching disaster affected areas in a short period of time is crucial to reduce the loss of lives. To the best of the authors' knowledge there is no systematic way utilized by governments to determine the paths to visit critical areas and decide which roads to clean immediately after an earthquake. As stated in the previous chapter the studies on debris removal in the post disaster phase mostly focus on complete removal of debris and recycling operations. In order to suggest solutions to the problems *Debris Removal in Response* and *Prioritized Debris Removal in Response* we develop two mathematical models for each. The first model treats each critical node equally in terms of importance and its objective is to minimize the total time spent to visit all the critical nodes. This model is called Minimize Total Time (MTT) and the objective value also corresponds to the visiting time of the last visited node. The second model, called Minimize Weighted Sum of Visiting Times (MWS)is developed for PDRR. Its objective is minimizing the sum of weighted visiting times so as to take priority relationship among the critical nodes into consideration. These models are developed under the setting explained below.

Let $G = (N, A)$ be a complete and symmetric graph where N is the node set, including critical nodes set C and noncritical node set NC, and A constitutes the arc set of the network. s denotes the supply node and $s \in C$. Time required for traversing arc $(i, j) \in A$ is t_{ij} and parameter I_{ij} takes value 0 if the arc (i, j)

is blocked. The arcs are blocked because of the wreckages caused by the disaster and they must be cleaned in order to be used. The effort spent on cleaning a blocked arc is measured in terms of time and it is denoted by c_{ij} for arc (i, j). Thus the time required to traverse a blocked arc (i, j) for the first time is $t_{ij} + c_{ij}$. Since the network is symmetric if (i, j) is blocked so is (j, i) and removing debris on one of them makes both of them clean. Furthermore, it is assumed that an unblocked arc cannot be blocked again so for the next usages of the arc only t_{ij} time is spent.

Critical nodes and arcs adjacent to these nodes are duplicated for the mathematical models. This is required to allow revisiting a critical node as an intermediate node and the necessity of these artificial nodes and arcs will be explained in detail after we introduce the model. Hence, each critical node $k \in C$ has a duplicated version k' and these artificial nodes are represented by set C'.

The arcs adjacent to the critical nodes are duplicated as well and included in set A' defined as $A' = A \cup \{(k', j),(j, k') : k' \in C', j \in N\} \cup (k', l') : k', l' \in C', k' \neq l'\}$. These artificial arcs have the same parameter values with the original ones so if the original arc (k, j) is blocked then all of them are blocked. However cleaning one of the original or artificial arcs once is sufficient to use these arcs. The set NC' contains these artificial critical nodes and original noncritical nodes; $NC' = NC \cup C'$. N' is the set of all original and artificial nodes, i.e., $N' = C \cup NC'$ so the new network is $G' = (N', A')$.

In the figure below an example of duplication of nodes and arcs in a small network is illustrated. The nodes k and l are critical where node i is noncritical. The dotted nodes and arcs are the artificial ones included in the new network.

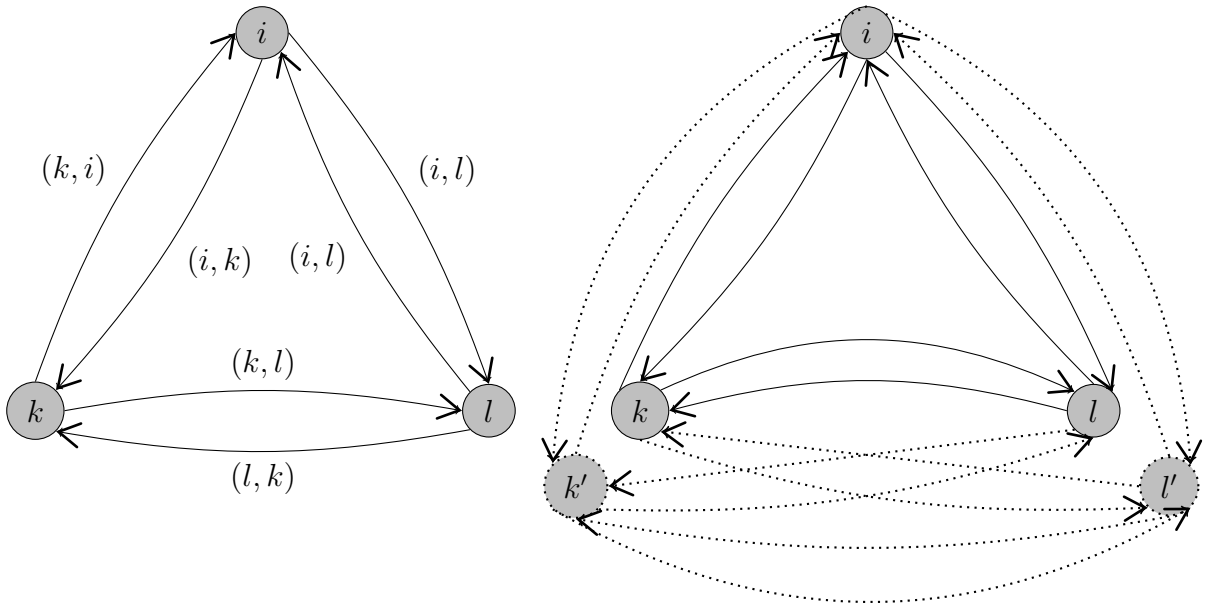


Figure 3.1: Original network $G = (N, A)$ (left), new network (right) $G' = (N', A')$ where dotted nodes and arcs are artificial.

As indicated earlier, *Debris Removal Problem in Response Phase* is studied by Şahin [3] and she suggested three mathematical models for the problem. Due to the periodic structure they possess, the first two models have found to be computational intractable so a third mathematical model, called Minimize Total Effort (MTE) is introduced. Since this model is found to be more efficient compared to the first two, the computational analyses are performed using this model. Yet, for some instances optimal solutions cannot be reached within hours. In this thesis, we first provide a more efficient mathematical model for the problem proposed by Şahin. Our model has a higher efficiency enabled by changing the decision variables. Before introducing our models, in order to clarify this alteration, the third model developed by Şahin [3] is represented below.

The same sets and parameters which are defined on the original network $G = (N, A)$ are used in this formulation and the decision variables are as follows:

y_{kl} =1 if $l \in C$ is visited right after $k \in C$, and 0 otherwise

x_{ij}^{kl} =1 if $(i,j) \in A$ is traversed while going from node $k \in C$ to $l \in C$, and 0 otherwise

C_{kl} = travel time spent to reach critical node $l \in C \backslash \{s\}$ from the critical node $k \in C$ if l is visited right after k (time required for debris removal not included)

B_{ij} =1 if $(i,j) \in A$ is cleaned, and 0 otherwise

p_k = time that node $k \in C$ is reached (time required for debris removal not included)

TT = total travel time spent to visit all critical nodes (time required for debris removal not included)

The model Minimize Total Effort (MTE) [3] is as follows:

$$\min \quad TT + \sum_{i,j \in N: i<j} B_{ij} c_{ij} \tag{3.1}$$

s.t.

$$\sum_{l \in C:\ l \neq k} y_{lk} = 1 \qquad \forall k \in C \backslash \{s\} \tag{3.2}$$

$$\sum_{l \in C:\ l \neq k} y_{kl} = 1 \qquad \forall k \in C \backslash \{s\} \tag{3.3}$$

$$\sum_{l \in C \backslash \{s\}} y_{sl} = 1 \tag{3.4}$$

$$\sum_{j \in N} x_{kj}^{kl} - \sum_{j \in N} x_{jk}^{kl} = y_{kl} \qquad \forall k, l \in C \tag{3.5}$$

$$\sum_{j \in N} x_{lj}^{kl} - \sum_{j \in N} x_{jl}^{kl} = -y_{kl} \qquad \forall k, l \in C \tag{3.6}$$

$$\sum_{j \in N} x_{ij}^{kl} - \sum_{j \in N} x_{ji}^{kl} = 0 \qquad \forall k, l \in C \quad \forall i \in N, i \neq k, i \neq l \tag{3.7}$$

$$p_s = 0 \tag{3.8}$$

$$p_l \geq p_k + C_{kl} - (1 - y_{kl})M \qquad \forall k \in C, l \in C \backslash \{s\} \tag{3.9}$$

$$TT \geq p_k \qquad \forall k \in C \backslash \{s\} \tag{3.10}$$

$$\sum_{i,j \in N} x_{ij}^{kl} \leq |N| y_{kl} \qquad \forall k, l \in C \quad (3.11)$$

$$C_{kl} = \sum_{i,j \in N} x_{ij}^{kl} t_{ij} \qquad \forall k, l \in C \quad (3.12)$$

$$\sum_{k,l \in C} x_{ij}^{kl} + \sum_{k,l \in C} x_{ji}^{kl} \leq (B_{ij} + I_{ij}) |C \setminus \{s\}| \qquad \forall i, j \in N i < j \quad (3.13)$$

$$TT \geq 0 \qquad (3.14)$$

$$p_k \geq 0 \qquad \forall k \in C \quad (3.15)$$

$$C_{kl} \geq 0 \qquad \forall k, l \in C \quad (3.16)$$

$$x_{ij}^{kl} \in \{0, 1\} \qquad \forall i, j \in N \; \forall k, l \in C \quad (3.17)$$

$$y_{kl} \in \{0, 1\} \qquad \forall k, l \in C \quad (3.18)$$

$$B_{ij} \in \{0, 1\} \qquad \forall i, j \in N \quad (3.19)$$

This model minimizes total time spent to visit all the critical nodes which is also the objective of our first model. Constraints 3.2 and 3.3 ensure that each critical node except the supply node has exactly one predecessor and successor critical node in order to form a visiting order. Constraint 3.4 guarantees that supply node is predecessor of exactly one of the critical nodes. These three assignment constraints construct a closed tour. Since constraint 3.8 makes the visiting time of the supply node equal to zero, it ensures that the tour starts from the supply node. Although the constraints imply that the vehicle returns to the supply node, the time spent to return to the supply node is not included in the objective function.

Constraints 3.5, 3.6 and 3.7 are flow balance constraints between each critical node. If critical node l is visited right after critical node k, constraint 3.5 ensures that the total flow leaving k minus entering k equals to 1. Similarly constraint 3.6 implies that total flow entering l minus leaving l equals to 1 if l comes right after k. Total flow entering and leaving is forced to be zero for any node other k and l by constraint 3.7.

Constraint 3.12 calculates time spent to go from critical node k to l only in terms of traveling time. Constraint 3.9 assigns visiting times of critical nodes again

excluding the time spent on debris removal. This constraint also prevents subtours among critical nodes. Constraint 3.10 and the objective together force TT to be equal to the visiting time of the last visited critical node.

Constraint 3.11 guarantees that no arc is traversed to go from critical node k to critical node l if l is not visited right after k. Constraint 3.13 ensures that an edge can be used between any critical node pair if it is already open or debris is cleaned.

Instead of binary variable x_{ij}^{kl} used in this model, we define binary variable x_{ij}^{k} which takes value 1 if arc (i, j) is traversed while going to critical node k from the predecessor critical node of k. Also the variable C_l is introduced to replace C_{kl} with the same definition which is the travel time spent to reach node l from the predecessor critical node of l. This reduction in number of indices is the main factor in the efficiency of our first model since it reduces the number of variables and constraints. This is also the reason of creating artificial nodes and arcs, which will be explained in detail after the introduction of our first model and discussion of the constraints.

In the next section we introduce formulation of our first model. Then our second model called Minimize Weighted Sum of Visiting Times is explained. The decision variables that are used in both mathematical models we developed are as follows:

y_{kl} =1 if $l \in C$ is visited right after $k \in C$, and 0 otherwise

x_{ij}^{k} =1 if $(i, j) \in A'$ is traversed while going to critical node k from the previous critical node

C_k = time spent to reach critical node $k \in C \setminus S$ from the previous critical node (the time required for debris removal not included)

3.1 First Model: Minimize Total Time

The additional decision variables used in the first model which minimizes total time required to visit all critical nodes are as follows:

B_{ij} =1 if $(i,j) \in A'$ is cleaned, and 0 otherwise

cb_{ij} =1 if $(i,j) \in A$ is cleaned, and 0 otherwise

p_k = time that node $k \in C$ is reached (debris removal not included)

TT = total travel time spent to visit all critical nodes (debri removal not included)

The formulation of the first model (MTT) is as follows:

$$\min \; TT + \sum_{i,j \in N: i<j} c_{ij} cb_{ij} \tag{3.20}$$

s.t.

$$\sum_{l \in C:\, l \neq k} y_{lk} = 1 \qquad \forall k \in C \backslash \{s\} \tag{3.21}$$

$$\sum_{l \in C:\, l \neq k} y_{kl} = 1 \qquad \forall k \in C \backslash \{s\} \tag{3.22}$$

$$\sum_{l \in C \backslash \{s\}} y_{sl} = 1 \tag{3.23}$$

$$\sum_{j \in NC' \cup \{l\}} x^l_{kj} = y_{kl} \qquad \forall k, l \in C \; k \neq l \tag{3.24}$$

$$\sum_{i \in N'} x^k_{ij} - \sum_{h \in NC' \cup \{k\}} x^k_{jh} = 0 \qquad \forall k \in C \; \forall j \in NC' \tag{3.25}$$

$$\sum_{i \in N'} x^l_{il} = 1 \qquad \forall l \in C \backslash \{s\} \tag{3.26}$$

$$C_l = \sum_{i,j \in N'} x^l_{ij} t_{ij} \qquad \forall l \in C \backslash \{s\} \tag{3.27}$$

$$p_s = 0 \tag{3.28}$$

$$p_l \geq p_k + C_l - (1 - y_{kl})\mu \qquad \forall k \in C, l \in C \backslash \{s\} \tag{3.29}$$

$$TT \geq p_k \qquad \forall k \in C \tag{3.30}$$

$$y_{kl} + y_{lk} \leq 1 \qquad \forall k, l \in C, k \neq l \tag{3.31}$$

$$B_{ij} \leq 1 - I_{ij} \qquad \forall i, j \in N' : i < j \tag{3.32}$$

$$\sum_{l \in C \setminus S} (x_{ij}^l + x_{ji}^l) \leq |C|(B_{ij} + I_{ij}) \qquad \forall i, j \in N' : i < j \quad (3.33)$$

$$B_{ij} + B_{ij'} + B_{i'j'} + B_{ji'} \leq 4cb_{ij} \qquad \forall i, j \in C \quad (3.34)$$

$$B_{ij} + B_{ij'} \leq 2cb_{ij} \qquad \forall i \in NC, j \in C : i < j \quad (3.35)$$

$$B_{ji} + B_{ij'} \leq 2cb_{ij} \qquad \forall i \in NC, j \in C : i > j \quad (3.36)$$

$$B_{ij} \leq cb_{ij} \qquad \forall i, j \in N : i < j \quad (3.37)$$

$$x_{ij}^k \in (0, 1) \qquad \forall i, j \in N', \ \forall k \in C \quad (3.38)$$

$$y_{kl} \in (0, 1) \qquad \forall k, l \in C \quad (3.39)$$

$$B_{ij} \in (0, 1) \qquad \forall i, j \in N' \quad (3.40)$$

$$cb_{ij} \in (0, 1) \qquad \forall i, j \in N \quad (3.41)$$

$$TT \geq 0 \quad p_k, C_k \geq 0 \qquad \forall k \in C \quad (3.42)$$

Constraints 3.21-3.23 and 3.28 have the same meaning with constraints 3.2-3.4 and 3.15 in MTE so they construct a route starting and ending at the supply node by ensuring each critical node has a predecessor and successor critical node. Again the time spent while returning to the supply node is not included in the objective function.

In both of the problems in this study, the critical nodes are allowed to be used as intermediate nodes. For example while going from critical node k to critical node l, another critical node m can be visited. In MTE, the flow balance constraints allow other critical nodes to be used on the path between two consecutive critical node. Thus, nodes i and j in constraint 3.5-3.7 can be critical.

$$\sum_{j \in N} x_{kj}^{kl} - \sum_{j \in N} x_{jk}^{kl} = y_{kl} \qquad \forall k, l \in C \quad 3.5$$

$$\sum_{j \in N} x_{lj}^{kl} - \sum_{j \in N} x_{jl}^{kl} = -y_{kl} \qquad \forall k, l \in C \quad 3.6$$

$$\sum_{j \in N} x_{ij}^{kl} - \sum_{j \in N} x_{ji}^{kl} = 0 \qquad \forall k, l \in C \ \ \forall i \in N, i \neq k, i \neq l \quad 3.7$$

When y_{kl} equals to 1 for some critical node k and l, it means l is visited right after k and for both them it is the first time that vehicle reaches them. Thus, when a critical node m is traversed on the path between k and l, it means m is visited earlier and it is revisited while going from k to l.

Flow balance constraint in our formulation does not allow a critical node to be revisited if we do not create artificial critical nodes. Without duplication of the critical nodes, the flow balance constraint in MTT become as follows:

$$\sum_{j \in N} x_{kj}^{l} = y_{kl} \qquad \forall k, l \in C \ k \neq l \quad 3.24'$$

$$\sum_{i \in N} x_{ij}^{k} - \sum_{h \in N} x_{jh}^{k} = 0 \quad \forall k \in C \ \forall j \in N \quad 3.25'$$

$$\sum_{i \in N} x_{il}^{l} = 1 \qquad \forall l \in C \backslash \{s\} \quad 3.26'$$

In the constraints above, for critical nodes k and l such that $y_{kl} = 1$, let node j be a node traversed while going from k to l. Node j cannot be a critical node because then x_{kj}^{l} would be equal to 1 and constraint 3.25′ forces x_{jh}^{l} to be 1 for some node h. Then due to constraint 3.24′ y_{jl} would be equal to 1 which means critical node l is visited right after critical node j and this is not possible since critical node l already has a predecessor which is critical node k.

To be able to use the critical nodes as intermediate nodes, we create artificial critical nodes that behave like noncritical nodes. Constraint 3.24 ensures that if critical node l is visited right after critical node k then an arc (k, j) is traversed to go from k to l where node j can be a regular noncritical node, an artificial critical node or critical node l. Thus either vehicle goes to critical node l directly or it goes to an intermediate node.

$$\sum_{j \in NC' \cup \{l\}} x_{kj}^{l} = y_{kl} \qquad \forall k, l \in C \ k \neq l \quad 3.24$$

$$\sum_{i \in N'} x_{ij}^{k} - \sum_{h \in NC' \cup \{k\}} x_{jh}^{k} = 0 \quad \forall k \in C \ \forall j \in NC' \quad 3.25$$

$$\sum_{i \in N'} x_{il}^{l} = 1 \qquad \forall l \in C \backslash \{s\} \quad 3.26$$

Constraint 3.25 ensures that for each intermediate node j traversed while going to critical node k, total flow entering and leaving j must be equal. Node j might be reached from any node, that is why $i \in N'$ in the first sum. In other words i can be a critical node, meaning that it is the predecessor of k or it can be an intermediate node. In the second sum h can be an intermediate node or the

destination itself, i.e, critical node k. Constraint 3.26 guarantees that there is exactly one entering arc (i, l) to each critical node l where i can be any node.

Constraint 3.27 ensures that if an arc is traversed to reach critical node l then its travel time but not debris cleaning time is added to C_l value, similar to constraint 3.12 in MTE. Constraint 3.29 eliminates sub-tours between critical nodes and assigns visiting times again excluding the time spent on debris removal. Constraint 3.30 is same as constraint 3.10 in MTE. Constraint 3.31 means that if $k \in C$ is visited before $l \in C$ then the opposite cannot be true. This is a valid inequality and it is implied by the sub-tour elimination constraint.

Constraint 3.32 ensures that only a blocked arc can be cleaned. Constraint 3.33 guarantees that an edge can be used only if it is clean or cleaned, similar to 3.13 in MTE. Constraints 3.34-3.37 assure that debris on $(i, j) \in A$ is removed if the original arc or one of the corresponding artificial arcs is cleaned. The actual variable indicating whether an arc is cleaned or not is cb_{ij} where $i, j \in A$. Therefore when one or more artificial arcs are cleaned it actually means the original arc is cleaned. By defining and using variable cb_{ij} in the objective function instead of B_{ij} we prevent spending cleaning time for the same arc more than once.

To explain the necessity of artificial nodes and arcs, assume that optimal solution has a partial path as shown in the figure where $k, l, m \in C$, $i, j \in NC$ and the arcs are numbered with respect to the order of travel.

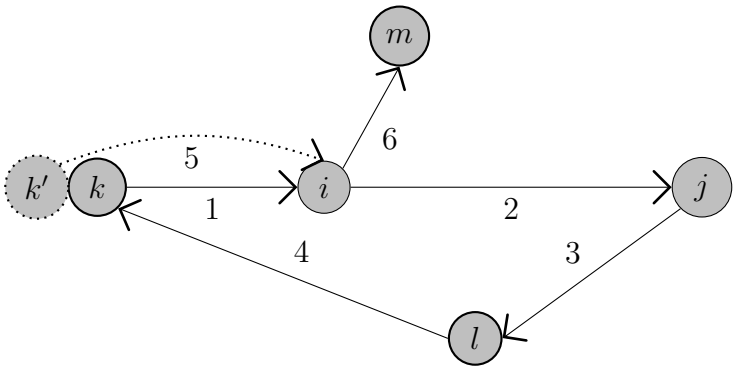


Figure 3.2: An example of revisiting a critical node where $k, l, m \in C, i, j \in NC$

The visiting order between critical nodes is k, l, m but node k is revisited to reach node m. Thus y_{kl} and y_{lm} take value 1 while y_{km} is 0. If we do not duplicate the critical nodes, since arc (k, i) is used to reach node m, x_{ki}^m should take value 1. However due to the constraint 3.24 and y_{km} being equal to 0, x_{ki}^m cannot take value 1. As stated in the problem definition in the first chapter, a critical node can be used as an intermediate node as in this path and without artificial nodes we cannot have this type of paths in the solution. To go from l to m, artificial node k' is used so instead of x_{ki}^m, $x_{k'i}^m$ takes value 1. Furthermore assuming arc (k, i) is blocked and cleared while going from k to l, the variable B_{ki} takes value 1 due to constraint 3.33. Since artificial arc $(k'i)$ is also used, $B_{k'i}$ has to be equal to 1 because of the same constraint. However (k, i) and $(k'i)$ correspond to the same arc so we need to consider only the first debris removal. This is guaranteed by variable cb_{ki} which is linked with B_{ki} and $B_{k'i}$ by constraints 3.34-3.37 for all possible node sets.

3.2 Second Model: Minimize Weighted Sum of Visiting Times

Treating each critical node equally might not be realistic since the characteristics of the nodes differ. It is reasonable to give some nodes priority if they are highly populated or more vulnerable. When the amount of debris, number of blocked arcs and number of critical nodes are high, time spent to reach all nodes may take hours. Therefore considering the weights or priorities of critical nodes and reaching the higher weighted ones sooner increase the overall benefit. For that purpose we have developed a second model which minimizes weighted sum of visiting times. The weights of the critical nodes are denoted by w_k for $k \in C$ for this model.

In the first model we have variable p_k which is the visiting time of node k but only considering the traveling times. In order to minimize weighted visiting times we need the actual time that a critical node is reached so if an arc (i, j) is cleaned while going to critical node k, the time required for debris removal should be

included. Therefore we need to know which arc is cleaned to reach a specific critical node. Since arc remains open once it is cleaned, spending debris removal time on that arc in the next usages must be prevented. To ensure that we need to know whether a critical node is visited earlier or later than another critical node and guarantee that a blocked arc is cleaned on its first usage.

Additional decision variables used in this model are as follows:
a_{kl} =1 if $l \in C$ is visited after $k \in C$, and 0 otherwise
v_{ij}^{k} =1 if (i, j) is cleaned to reach node $k \in C$, and 0 otherwise
r_k = time that node $k \in C$ is reached

The variable a_{kl} is different than y_{kl} since it takes value 1 if critical node k is visited any time before critical node l, not only when they are consecutively visited. Because of this variable, the vehicle starts from the supply node but does not return to it so the path finishes when the last critical node is visited. The second model is as follows:

$$\min \sum_{k \in C \setminus \{s\}} w_k r_k \tag{3.43}$$

s.t.

$$\sum_{k \in C:\ k \neq l} y_{kl} = 1 \qquad \forall l \in C \setminus \{s\} \tag{3.44}$$

$$\sum_{k \in C, l \in C \setminus \{s\}:\ k \neq l} y_{kl} = |C \setminus \{s\}| \tag{3.45}$$

$$\sum_{l \in C:\ l \neq k} y_{kl} \leq 1 \qquad \forall k \in C \tag{3.46}$$

$$a_{kl} \geq y_{kl} \qquad \forall k, l \in C \tag{3.47}$$

$$a_{kl} + a_{lk} = 1 \qquad \forall k, l \in C, k \neq l \tag{3.48}$$

$$a_{ml} \geq a_{mk} + y_{kl} - 1 \qquad \forall k, l, m \in C, k \neq l \tag{3.49}$$

$$a_{sl} = 1 \qquad \forall l \in C \setminus \{s\} \tag{3.50}$$

$$\sum_{j \in NC' \cup \{l\}} x_{kj}^{l} = y_{kl} \qquad \forall k, l \in C \setminus \{s\}\ k \neq l \tag{3.51}$$

$$\sum_{i \in N'} x_{ij}^{k} - \sum_{h \in NC' \cup \{k\}} x_{jh}^{k} = 0 \qquad \forall k \in C \setminus \{s\}\ \forall j \in NC' \tag{3.52}$$

$$\sum_{i \in N'} x_{il}^{l} = 1 \qquad \forall l \in C \backslash \{s\} \qquad (3.53)$$

$$C_l = \sum_{i,j \in N'} x_{ij}^{l} t_{ij} \qquad \forall l \in C \backslash \{s\} \qquad (3.54)$$

$$r_s = 0 \qquad (3.55)$$

$$r_l \geq r_k + C_l + \sum_{i,j \in N : i<j} v_{ij}^{k} c_{ij} + (1 - y_{kl})M \qquad \forall k \in C \;\; \forall l \in C \backslash \{s\} \qquad (3.56)$$

$$y_{kl} + y_{lk} \leq 1 \qquad \forall k, l \in C, k \neq l \qquad (3.57)$$

$$\sum_{l \in C \backslash \{s\}} v_{ij}^{l} \leq 1 - I_{ij} \qquad \forall i, j \in N : i < j \qquad (3.58)$$

$$2 - v_{ij}^{l} \geq x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{ji'}^{k} + x_{j'i}^{k} + x_{ij'}^{k} + x_{i'j'}^{k} + x_{j'i'}^{k} + a_{kl} \qquad \forall i, j, k, l \in C : i < j, I_{ij} = 0, k \neq l \qquad (3.59)$$

$$2 - v_{ij}^{l} \geq x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{ji'}^{k} + a_{kl} \qquad \forall i, k, l \in C, j \in NC : i < j, I_{ij} = 0, k \neq l \qquad (3.60)$$

$$2 - v_{ji}^{l} \geq x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{i'j}^{k} + a_{kl} \qquad \forall i, k, l \in C, j \in NC : i > j, I_{ij} = 0, k \neq l \qquad (3.61)$$

$$2 - v_{ij}^{l} \geq x_{ij}^{k} + x_{ji}^{k} + a_{kl} \qquad \forall k, l \in C, i, j \in NC : i < j, I_{ij} = 0, k \neq l \qquad (3.62)$$

$$|C \backslash \{s\}| \sum_{k \in C \backslash \{s\}} v_{ij}^{k} \geq \sum_{k \in C \backslash \{s\}} (x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{ji'}^{k} + x_{j'i}^{k} + x_{ij'}^{k} + x_{i'j'}^{k} + x_{j'i'}^{k}) \qquad \forall i, j \in C \backslash \{s\} : I_{ij} = 0, i < j \qquad (3.63)$$

$$|C \backslash \{s\}| \sum_{k \in C \backslash \{s\}} v_{ij}^{k} \geq \sum_{k \in C \backslash \{s\}} (x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{ji'}^{k}) \qquad \forall i \in C \backslash \{s\}, j \in NC : I_{ij} = 0, i < j \qquad (3.64)$$

$$|C \backslash \{s\}| \sum_{k \in C \backslash \{s\}} v_{ji}^{k} \geq \sum_{k \in C \backslash \{s\}} (x_{ij}^{k} + x_{ji}^{k} + x_{i'j}^{k} + x_{ji'}^{k}) \qquad \forall i \in C \backslash \{s\}, j \in NC : I_{ij} = 0, i > j \qquad (3.65)$$

$$|C \backslash \{s\}| \sum_{k \in C \backslash \{s\}} v_{ij}^{k} \geq \sum_{k \in C \backslash \{s\}} (x_{ij}^{k} + x_{ji}^{k}) \qquad \forall i, j \in NC : I_{ij} = 0, i < j \qquad (3.66)$$

$$x_{ij}^{k} \in (0, 1) \qquad \forall i, j \in N', \; \forall k \in C \qquad (3.67)$$

$$v_{ij}^{k} \in (0, 1) \qquad \forall i, j \in N, \; \forall k \in C \qquad (3.68)$$

$$a_{kl}, y_{kl} \in (0, 1) \qquad \forall k, l \in C \qquad (3.69)$$

$$r_k, C_k \geq 0 \qquad \forall k \in C \qquad (3.70)$$

Constraint 3.44 ensures that each critical node except the supply node has a predecessor critical node same as in MTE and MTT. Constraint 3.45 limits the

total number of assignments to the number of critical nodes that we need to reach. Constraint 3.46 implies that a critical node may have a successor critical node or not. These are different than the assignment constraints of the previous formulation because the vehicle does not return to the supply node. This is needed because of the variable a_{kl} and constraint 3.48 which assures either $k \in C$ is visited before $l \in C$ or vice versa.

Constraint 3.47 implies that if $k \in C$ is visited just before $l \in C$ then k is visited before l and with constraint 3.49 we satisfy that any critical node m which is visited before k is also visited before l. The supply node is guaranteed to be the start node with constraints 3.50 and 3.55.

Constraints 3.50-3.53 are flow balance constraints identical to 3.25-3.27 and constraint 3.54 is same as 3.27 in MTT. Constraint 3.56 eliminates sub-tours between critical nodes and assigns visiting times including the time spent on debris removal. Thus if an arc (i, j) is cleaned while going to critical node k, its cleaning time c_{ij} is added to r_k. Constraint 3.57 is the same valid inequality at 3.31.

Constraint 3.58 implies that an arc is cleaned only once and only if it is blocked. Constraints 3.41-3.44 prevent a blocked arc from being cleaned in latter usage. For example if a blocked arc (i, j) or one of its artificial version have traversed while going to critical node k and if k is visited before critical node l, then (i, j) cannot be cleaned while going to critical node l. Constraints 3.63-3.66 ensure that a blocked arc is cleaned while going to a critical node in order to be traversed to reach any critical node. Hence a blocked arc (i, j) is cleaned if it is used at least once. These last constraints 3.58-3.66 together guarantee that a blocked arc is cleaned once if it is used and debris is removed on its first usage.

Chapter 4

Heuristic Algorithms

The number of noncritical and critical nodes, their locations and the severity of the earthquake cause variations in the solution times of both mathematical models. With higher dimensions and different parameter values, reaching optimality may take several hours. Since the aim of the problem is finding a route to reach critical nodes and visiting them as soon as possible, waiting for an optimal solution for hours conflicts with the essence of the problem. Therefore, for the cases where finding an optimal solution takes longer than a reasonable amount of time, in order to get a feasible route sooner, little deviations from optimality can be bearable.

In order to obtain near optimal solutions quickly we have developed two constructive heuristic algorithms for both of the problems we defined earlier. Hence, the first constructive heuristic aims to minimize the total time where the second aims to find a route with minimum sum of weighted visiting times. To decrease the optimality gaps, improvement heuristic is applied to solutions obtained from these constructive heuristics. The constructive heuristics utilize Dijkstra's algorithm and the improvement heuristic is based on 2-opt algorithm [57]. In the following subsections the constructive heuristics and the use of 2-opt algorithm are described in detail.

4.1 Constructive Heuristics

We have developed two algorithms, called Minratio and Weighted Shortest Distance for the first and second problem respectively. The aim of the first one is to find a predecessor and successor for each critical node so as to form a route with minimum total time as a solution to the problem *Debris Removal in Response*. With the second algorithm we try to find a route which gives the minimum weighted sum of visiting times for the problem *Prioritized Debris Removal in Response*.

4.1.1 Minratio Heuristic

This heuristic first finds the shortest paths between critical nodes with Dijkstra's algorithm. Then for each critical node k it calculates a ratio dividing the distance from k to its closest critical node by average distance from k to all other critical nodes. We denote the distance of shortest path from k to each critical node l as $c(k, l)$ and min_k is the distance of the closest critical node to k. If we call the average shortest path distances from k to all eligible critical nodes avg_k then the ratio for k is $ratio_k = min_k$ / avg_k. For a critical node other than supply node, eligibility means having a degree less than 2 in the current subgraph. The supply node is eligible if it has a degree zero in other words if it is not paired with any critical node. We define set E which consist of eligible critical nodes so $ratio_k$ is calculated $\forall k \in E$ considering all distances $c(k, l)$ where $l \in E$ and l is not connected k.

After calculating these ratios for all critical eligible nodes, the node which gives the minimum ratio is chosen. Say critical node k is chosen and assume that closest critical node to k is l. Then the algorithm connects nodes k and l using the shortest path with the value min_k, which equals to $c(k, l)$, meaning that either k is visited just before l or vice versa. Until there is no unconnected critical node, the algorithm continues to calculate the ratios, pick the one giving the minimum and connects it to the closest critical node.

Smaller ratio for a critical node k indicates that the difference between min_k and the other distances to the critical nodes except the closest one to k, is larger with respect to the other critical nodes. By choosing the node with minimum ratio we aim to find and benefit from most advantageous path. In other words, we try to consider all critical nodes together and connect the most distant one to its closest. This makes our algorithm less myopic compared to Nearest Neighbor (NN) because NN algorithm starts from the supply node, visits the closest critical node until all nodes are visited so it considers one critical node at each iteration.

By a simple example we can illustrate how Minratio works and its difference to NN algorithm. In Figure 4.1 there are 4 critical nodes where s is the supply node and the dashed edges show the shortest paths among the critical nodes so they might be sharing some arc which can be blocked. For simplicity, we have only one blocked arc with 1 unit cleaning time and it is used in the shortest paths between $k-l$ and $k-m$. When we apply NN algorithm to construct a route, the vehicle visits k first since it is closest one to s, then it visits m and since we clean the blocked arc we update the shortest path distance between $k-l$. Thus $c(k,l)$ drops to 6. Since the current source node is m, node l is visited after m and route is completed with a total cost of $4+4+10=18$.

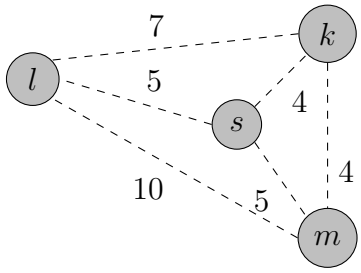


Figure 4.1: A simple network where dashed edges show the shortest path between each node

Instead of forming a route starting from the supply node, Minratio chooses critical nodes to connect and construct sub-routes. Average distances from a critical node to all other eligible critical node are calculated at each iteration to find the pair giving the minimum ratio.

As seen in Table 4.1, in each iteration the critical node which gives the smallest ratio is chosen and connected to the closest critical node so node m is chosen in the first iteration and paired with node k. Then the shortest paths between eligible nodes recalculated by considering cleaned arcs. A blocked arc is cleaned between the path m and k, and as stated earlier it is also used in the shortest path between nodes k and l. Thus we need to update the shortest distance between these node because a blocked arc does not require cleaning more than once. Therefore, $c(k, l)$ (and $c(l, k)$) is reduced 1 unit and becomes 6. After computing new ratios node m is chosen again in the next iteration. After connecting node m with s, both of them become ineligible. Then l is chosen and connected to node k or vice versa in the last iteration. Thus the route is constructed with total cost of $4 + 5 + 6 = 15$.

Table 4.1: An example of Minratio Algorithm

Iteration 1	s	k	l	m	min	avg	ratio
s	-	4	5	5	4	4.67	0.86
k	4	-	7	4	4	5	0.80
l	5	7	-	10	5	7.33	0.68
m*	5	**4**	10	-	4	6.33	**0.63**
	Node m connected to node k						
Iteration 2	s	k	l	m	min	avg	ratio
s	-	4	5	5	4	4.67	0.86
k	4	-	6	-	4	5	0.80
l	5	6	-	10	5	7	0.71
m*	**5**	-	10	-	5	7.5	**0.67**
	Node m connected to node s						
Iteration 3	s	k	l	m	min	avg	ratio
s	-	-	-	-	-	-	-
k	-	-	6	-	6	6	1
l*	-	**6**	-	-	6	6	**1**
m	-	-	-	-	-	-	-
	Node l connected to node k						

The flowchart of our first constructive algorithm, Minratio, is given in Figure 4.2 and the pseudo-code of the algorithm is presented below.

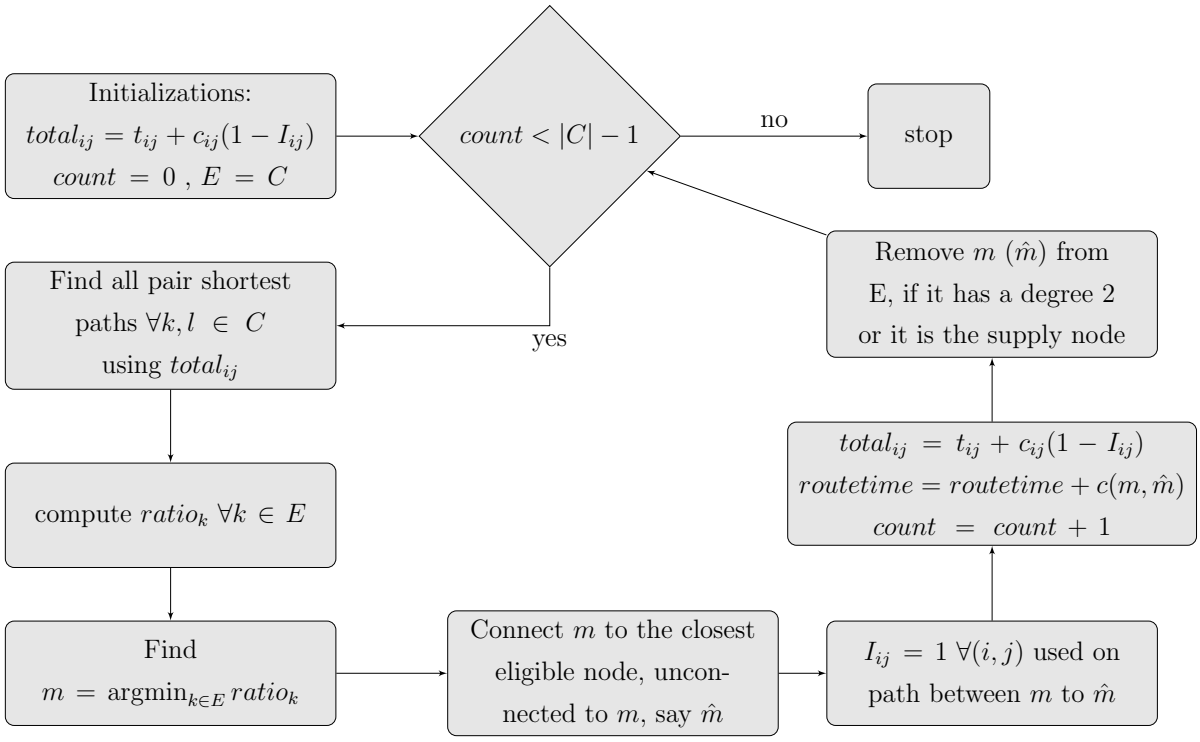


Figure 4.2: Flowchart of Algorithm Minratio

Algorithm 1: Minratio Algorithm

Data: Sets N, C, parameters t_{ij}, c_{ij}, I_{ij}

initialize:

$total_{ij} = t_{ij} + c_{ij}(1 - I_{ij})$ for all $(i, j) \in A$;

Set of eligible nodes $E = C$;

$count = 0$, $routetime = 0$, $route = \emptyset$

while $count < |C| - 1$ **do**

  forall the *critical nodes* k *and* l **do**

    Find shortest path using $total_{ij}$;

    Keep shortest distances $c(k, l)$, $c(l, k)$ and paths, $path(k, l)$, $path(l, k)$

  end

  Compute $ratio_k$ $\forall k \in E$, considering only eligible and unconnected nodes to k;

  Find $m = \text{argmin}_{k \in E}\, ratio_k$;

  Connect m to $\hat{m}$ such that $\hat{m} = \text{argmin}_{\hat{m} \in E}\, c(m, \hat{m})$ and not connected to m ;

  $routetime = routetime + c(m, \hat{m})$;

  $I_{ij} = 1$ $\forall$ (i, j) in $path(m, \hat{m})$ and $path(\hat{m}, m)$;

  $total_{ij} = t_{ij} + c_{ij}(1 - I_{ij})$ for all $(i, j) \in A$;

  Connect m and $\hat{m}$;

  $count = count + 1$

  if *m has degree 2 in the current constructed subgraph or $m = s$* **then**

    $E = E \setminus m$

  end

  if *$\hat{m}$ has degree 2 in the current constructed subgraph or $\hat{m} = s$* **then**

    $E = E \setminus \hat{m}$

  end

end

Output: *route*, *routetime*

4.1.2 Weighted Shortest Distance Heuristic

For the second objective which is minimizing sum of weighted visiting times, as introduced in the previous chapter, we have developed another mathematical model. Since we need to know the exact visiting times of the critical nodes and blocked arcs cleaned to reach these nodes, we need extra variables and constraints which increase the computational times. Therefore, we propose another constructive heuristic which aims to minimize the sum of weighted visiting times.

The previous model connects critical nodes without considering the times they are reached since the objective is minimizing total time. However, for our second objective, we should focus on the individual visiting times for each critical node. Therefore, we develop Weighted Shortest Distance Heuristic which is basically a weighted version of NN algorithm with some modifications. The algorithm starts from the supply node, choose one critical node to visit, travels there and treating this critical node as the current source node, finds the next critical node until all of them are reached.

To choose which critical node to visit next, first we apply Dijkstra's algorithm and find the shortest paths to the current source node. Having these shortest distances for all critical nodes that are not visited yet, we find a ratio dividing these values by the weights of the nodes.

Say min_k is the shortest path from k to the current source node, dividing it by w_k we obtain a ratio, say $wratio_k$. The critical node giving the minimum value among these ratios is picked and visited next. The aim of choosing the one with the minimum $wratio$ value is to visit relatively close and high priority critical nodes first. The flowchart of the algorithm is given in Figure 4.3 and the pseudo-code of the algorithm can be seen below.

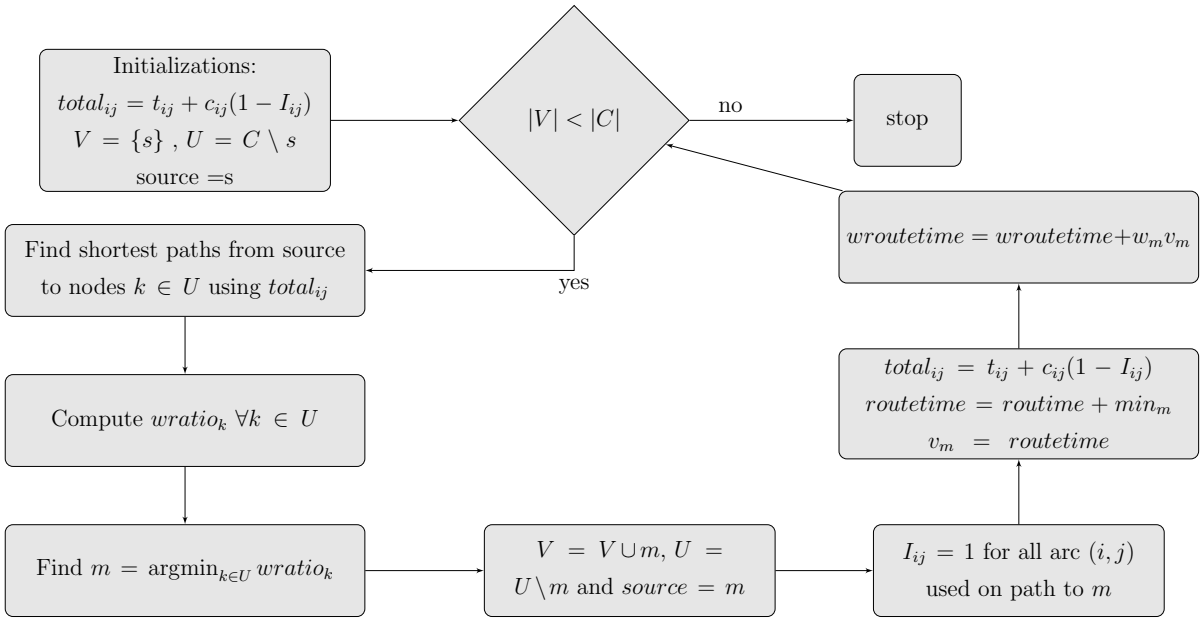


Figure 4.3: Flowchart of Algorithm Weighted Shortest Distance

Algorithm 2: Weighted Shortest Distance Algorithm

Data: Sets N, C, parameters t_{ij}, c_{ij}, I_{ij}, w_k

initialize:

$total_{ij} = t_{ij} + c_{ij}(1 - I_{ij}) \; \forall (i,j) \in A$;

Set of visited nodes $V = \{s\}$, set of unvisited nodes $U = C \setminus s$;

$source = s$, $wroutetime = 0$, $route = \emptyset$

while $|V| < |C|$ **do**

  forall the *critical nodes in* U **do**

    find the shortest paths and distance to node using $total_{ij}$;

    Keep paths $path_k$ and shortest distances $min_k \; \forall k \in U$;

    $wratio_k = min_k \; / \; w_k \; \forall k \in U$

  end

  Find $m = \text{argmin}_{k \in U} \, wratio_k$;

  $V = V \cup m$, $U = U \setminus m$;

  $source = m$;

  add $path_m$ to $route$;

  $I_{ij} = 1 \; \forall \; (i,j)$ in $path_m$;

  $total_{ij} = t_{ij} + c_{ij}(1 - I_{ij}) \; \forall (i,j) \in A$;

  $routetime = routime + min_m$;

  visiting time of node m, $v_m = routetime$;

  $wroutetime = wroutetime + w_m v_m$

end

Output: $route$, $wroutetime$

4.2 Improvement Heuristics

In order to increase the quality of the solution, we apply the 2-opt algorithm to the solutions obtained from the constructive heuristics. 2-opt is a local search algorithm proposed for traveling salesman problem and it is applicable to many routing problems.

The algorithm basically takes a feasible route, chooses a pair of nodes and reverses the path between them, and repeats this process until a better path cannot be found. For a complete search all possible pairs are swapped and results are compared. For our problem, a swap pair can only be chosen among the critical nodes except the supply node. In the figure below, an iteration applied on a feasible route is illustrated where k_n s are critical nodes and i_n s are intermediate nodes. Critical nodes k_2 and k_4 are chosen as a pair and the path between them is reversed.

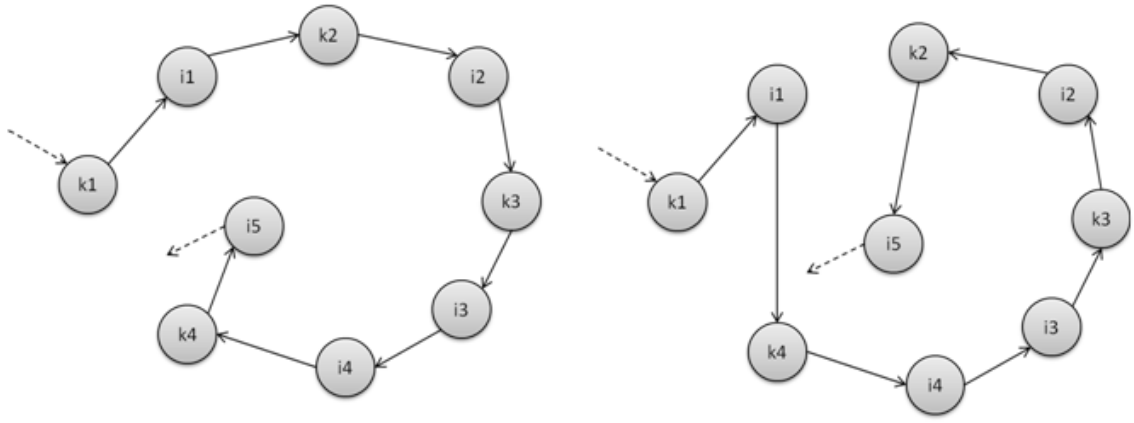


Figure 4.4: An example of 2-opt applied on a feasible route

In order to compare the routes obtained by applying 2-opt with each other and with the starting solution we need to calculate the objective values. While calculating the total time or weighted sum of visiting times we need to check the usage of the blocked arcs. For example, in the figure above, if arc (i_1, k_2) is blocked and not cleaned earlier, then the debris on it is removed while going from k_1 to k_2 in the first route. However the arc is not used in the second route obtained by 2-opt so to calculate the times correctly we need to subtract this cleaning

time. Similarly, if arc (k_2, i_5) is blocked and not cleaned before, it is required to include the debris removal time in the second route. If it is cleaned in the earlier visitations then debris removal time should not be considered while going from k_2 to some other critical node.

The pseudo-code of improvement algorithms applied to the constructions heuristics are presented below. The *swap* procedure is the same for both improvement algorithms but the calculation of times are different. In the first improvement algorithm *RouteTime* calculates the total time required to complete the new route formed by *swap* procedure as the purpose is minimizing total time. In the second improvement algorithm since we try to find a route with minimum sum of weighted visiting times, *RouteTimeWeighted* calculates this time after each *swap* procedure. These procedures can be seen in Figure 4.5.

The pseudo-code of improvement heuristics *2-opt* and *2-optWeighted* is as follows:

Algorithm 3: 2-opt/2-optWeighted Algorithms

Data: Sets N, C, parameters t_{ij}, c_{ij}, I_{ij}
initialize:
$newroutetime = 0$
Input: route,routetime
while $newroutetime \leq routetime$ **do**
  forall the $k, l \in C \setminus \{s\}$ **do**
    Find *newroute* by *swap(k,l)* ;
    Compute *newroutetime* by *RouteTime*/*RouteTimeWeighted* ;
    if $newroutetime < routetime$ **then**
      $routetime = newroutetime$;
      $newroute = route$;
    end
  end
end
Output: *newroute*, *newroutetime*

Procedure swap(k,l).

Input: route

if *k comes before l in route* **then**
  $i = k$, $j = l$
end
else
  $i = l$, $j = k$
end
add nodes to *newroute* up to node i excluding i;
add nodes between i and j to *newroute* in reverse order ;
add nodes after j to *newroute*

Output: *newroute*

Procedure RouteTime(route).

Input: route

$i = 0$, $j = 1$;
$l =$ *route length* -1 ;
while $i < l$ **do**
  $newroutetime = newroutetime + t_{ij} + c_{ij} * (1 - I_{ij})$;
  Update $I_{ij} = 1$;
  $i = i + 1$, $j = j + 1$
end

Output: newroutetime

Procedure RouteTimeWeighted(route).

Input: route

$i = 0$, $j = 1$, $p_0 = 0$;
$l =$ *route length* -1 ;
while $i < l$ **do**
  $routetime = routetime + t_{ij} + c_{ij} * (1 - I_{ij})$;
  $p_j = routetime$;
  Update $I_{ij} = 1$;
  $i = i + 1$, $j = j + 1$
end
$newroutetime = \sum_{k \in C} p_k * w_k$;

Output: *newroutetime*

Figure 4.5: Procedures used in improvement algorithms

Chapter 5

Data and Computational Analysis

In this study, we define two problems and develop two mathematical models both of which aims to find a route to visit some set of required nodes. While the first problem, *Debris Removal in Response*, focuses on the total time, the second, *Prioritized Debris Removal in Response*, aims to minimize sum of weighted visiting times. Although the mathematical models provide optimal solutions, when the dimensions increase and the parameters change, it may take too much time to solve the problems. Therefore we propose two constructive heuristics followed by an improvement algorithm to obtain near optimal solutions quickly. For the test of these solution methodologies and to be able to compare the performances we use two different networks under different settings. In the following sections the information about these data sets is given. The computational studies on mathematical models and heuristics are presented in the subsequent sections.

5.1 Data

We test our models and algorithms using two different data sets based on two districts of İstanbul, Turkey. The first set, Kartal, has 45 nodes, 7 of which

are critical. The second set, Bakırköy, has 73 nodes including 15 critical nodes. Hereafter by critical nodes we refer to the critical nodes excluding the supply node. In the data sets the critical nodes correspond to the neighborhoods which are close to schools and hospitals. Detailed information about these data sets can be found in the studies by Kılcı [58] and Şahin [3]. The features of the data sets are summarized in Table 5.1.

Table 5.1: Features of the data set

	Kartal	Bakırköy
# of nodes	45	73
# of critical nodes	7	15
Supply Node	16-Marmara Region Disaster Center of Turkish Red Cresent	7-Disaster Coordination Center
Node numbers of critical nodes	14,21,22,26,33,41,43	5,15,16,17,18,19,20,21,22,34,36,47,55,65,67
Node numbers of only schools	14,21,22	5,15,34,36,47,55,65,67
Node numbers of only hospitals	26,33,41,43	16,17,18,19,20,21,22

The maps in Figure 5.1 and 5.2 show the locations of supply nodes and critical nodes in Kartal and Bakırköy, respectively. In both, red triangle represents the supply node, yellow circles correspond to schools and green circles are the locations of the nodes near hospitals.

For the computational analyses same parameters and instance are used as in the study by Şahin [3]. Travel times between each node are calculated using distance matrices and assuming that the vehicle has a speed of 20km/hour. These time matrices are symmetric and satisfy triangular inequality. Both of the networks are complete.

Four degrees of earthquake severity (SOE) are used where 4 is the most severe one. The blocked arc ratio (BAR) corresponding to these severity degree is accepted as in Table 5.2 and blocked arcs are randomly assigned by taking value 0 in I matrix according to the blocked arc ratios.

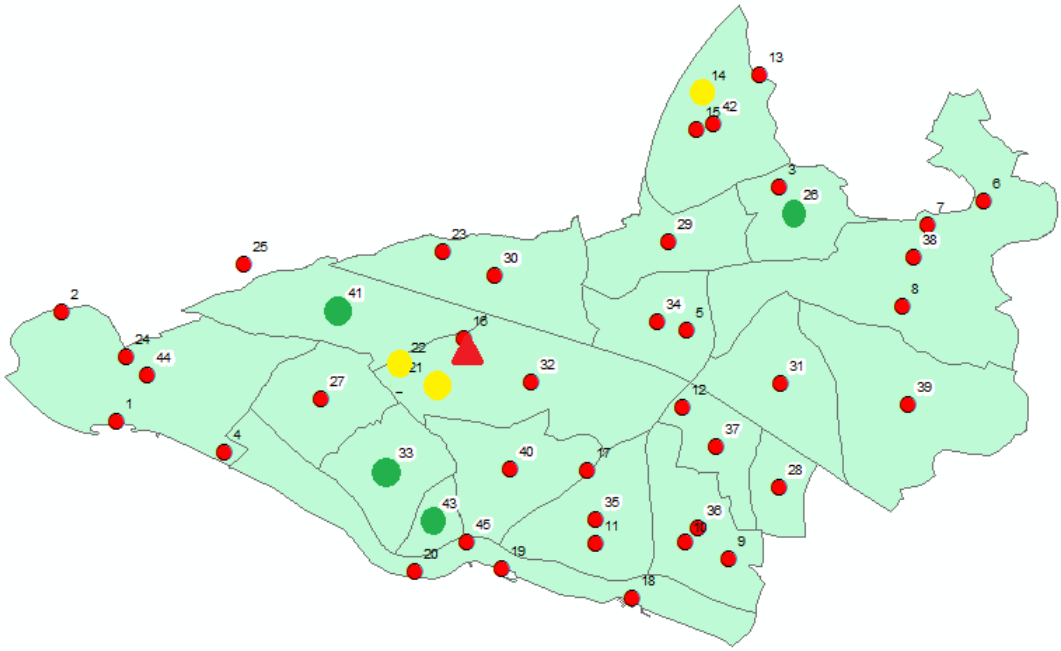


Figure 5.1: The location of supply node and critical nodes in Kartal

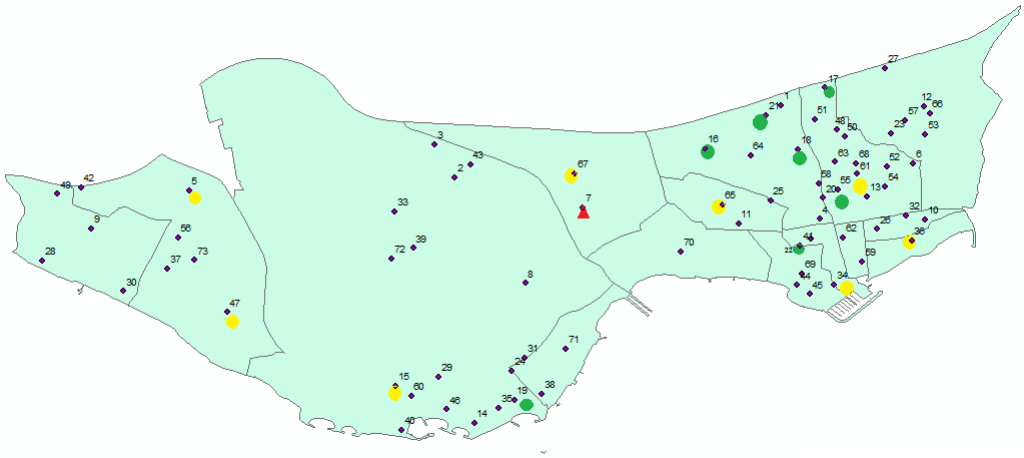


Figure 5.2: The location of supply node and critical nodes in Bakırköy

Table 5.2: Severity of earthquake, corresponding BAR values and BAR values used in Kartal and Bakırköy instances.

SOE	BAR	BAR value for Kartal	BAR value for Bakırköy
1	0-20%	12.5%	19%
2	20-50%	44.5%	23%
3	50-80%	58%	54%
4	80-100%	81.9%	82%

The time required to remove debris on an arc is calculated in two different ways, both depend on the severity of the earthquake and the travel time of the arc. c_{ij} denotes the higher cleaning time and c'_{ij} denotes the lower. They are calculated as follows:

$$c_{ij} = SOE * t_{i,j} + U[0, \underset{(i,j)\in A}{\operatorname{argmax}} t_{ij}]$$

$$c'_{ij} = SOE * t_{i,j}$$

For each degree of severity, 5 instances are generated with the same BAR value so they have the same number of blocked arcs but the location of the blocked arcs are different. These instances are solved using two different cleaning times given above. Hence, for Kartal data set for both cleaning times we have 20 instances, 5 for each SOE. For example K1, K2, K3, K4, K5 has the same number of blocked arc but in different locations and the cleaning time c_{ij} is used. K5 and K5$'$ use the same I matrix but different debris removal times. For Bakırköy instances 3 different critical node sets are used as only schools, only hospitals and all critical nodes.

For the second model, to determine the weights assigned to each critical node, population of neighborhoods to which each critical node belongs is used. The population of the neighborhoods are obtained form Turkish Statistical Institute and they are normalized so that each critical node has a weight value out of 100, and sum of all weights is equal to 100. The normalization is done only for the nodes included in the set for the related instance. Thus when the critical nodes are only schools in Bakırköy instances, the hospitals are not considered in the calculation of the weights.

5.2 Computational Analysis

Using the instances explained in the previous section, we test both of our models and heuristics. The experiments on mathematical models are conducted using CPLEX 12.6 and Gurobi 5 on a 4XAMD Opreton Interlagos 16C 6282SE 2.6G 16M 6400MT computer. The heuristic algorithms are coded in Java and ran on a personal computer with a processor Intel Core i5 CPU at 2.4 GHz and 4 GB of RAM. Since the heuristics take less than a second, CPU times for them are not reported.

5.2.1 Analyses on the First Model

For each earthquake severity from 1 to 4, and for higher and lower cleaning efforts, 5 different instances in terms of blocked arcs' locations are used. For Kartal instances we consider all the critical nodes and since Bakırköy is a larger network the experiments are conducted with different critical node sets; only schools, only hospitals and all critical nodes. In Tables 5.3 and 5.4 results of the first model are illustrated with higher and lower cleaning efforts respectively. In these tables, instance features, optimal values and the CPU times of our first model (MTT) and the third model (MTE) developed by Şahin [3] can be seen and in the last column number of cleaned arcs is given for each instance. As summarized in Table 5.5 CPU times are lowered drastically for Kartal instances by our model MTT.

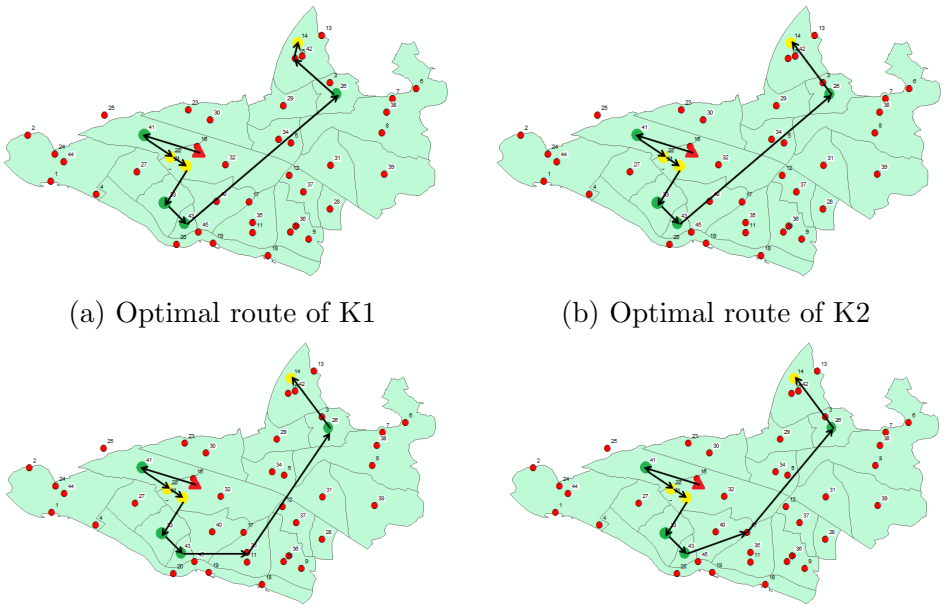
Parallel to our intuitions, as the severity of the earthquake increases, the CPU times increase as well. This is due to increase in the number of blocked arcs. In other words, deciding which arcs to clean becomes more important when their number rises. When we focus on the instances with the same severity level, we observe that the locations of the blocked arcs also have an impact on the path and CPU times. For example instances K17 and K20 have the same number of blocked arcs however the blocked arcs are not identical. This causes differences in optimal values, number of cleaned arcs and also CPU times.

Table 5.3: Performance of the first model on Kartal instances with higher cleaning times

Instances Features	Instance #	optimal value	CPU time (Cplex)		# of cleaned arcs
			MTT	MTE [3]	
SOE=1, # of critical nodes=7, cleaning effort : c_{ij}	K1	44	39.07	221.41	0
	K2	43	50.25	186.73	0
	K3	44	51.13	190.82	0
	K4	43	43.04	178.4	0
	K5	43	42.63	183.2	0
SOE=2, # of critical nodes=7, cleaning effort : c_{ij}	K6	48	49.25	223.26	0
	K7	50	56.34	235.96	0
	K8	51	58.22	270.41	0
	K9	49	68.92	225.46	0
	K10	48	41.44	261.39	0
SOE=3, # of critical nodes=7, cleaning effort : c_{ij}	K11	53	71.55	315.37	0
	K12	63	162.14	495.63	0
	K13	68	114.85	381.4	0
	K14	46	57.72	196.96	0
	K15	47	58.17	186.72	0
SOE=4, # of critical nodes=7, cleaning effort : c_{ij}	K16	109	616.1	5167.18	1
	K17	82	372.72	3319.8	0
	K18	110	551.87	9136.87	1
	K19	90	348.14	2915.86	1
	K20	101	490.14	4541.84	3

When we look at the number of cleaned arcs for each instance, we observe that cleaning is needed when the number of blocked arcs increases. Since the cleaning times are also higher for the instances where the blocked arcs are many, the optimal value increases as it can be easily seen when SOE equals to 4.

In the first 5 instances of Kartal, no arcs are cleaned and the optimal paths have slight differences. In the Figure 5.3 these paths are illustrated. From these optimal paths we see that cleaning is not needed since the number of blocked arcs are low and a path which does not include any blocked arcs with a low total time can be found. For example in the solution of instance K2, the vehicle goes from node 26 to node 14 directly but in K1 since that arc is blocked the vehicle uses node 15 as an intermediate node between 26 and 14. This change in the path increases the total time only by one unit as it can be seen in the optimal values of instances K1 and K2 in Table 5.3.



(a) Optimal route of K1

(b) Optimal route of K2

(c) Optimal route of K3

(d) Optimal route of K4 and K5

Figure 5.3: Optimal routes of instances K1,K2,K3,K4,K5 for MTT

When we compare instances for higher and lower debris cleaning times, we observe that when the cleaning times are lower, it is more preferable to clean arcs. For example, instances K16 and K16′ have the same severity level, number and location of blocked arcs, the only difference is the time required to clean blocked arcs. This difference increases the number of blocked arcs from 1 to 3 between K16 and K16′ while the total time drops from 109 to 97. The effect of cleaning times can also be realized by comparing instances K18 and K18′. When the cleaning is lower the optimal value decreases from 110 to 95 while the number of blocked arcs increase from 1 to 3 for these instances. Furthermore, time needed to reach optimal solution for K18′ is less than half of the time needed for K18. Although the average CPU time of the instances with lower cleaning times is less than the average CPU time of the ones with higher cleaning times, the main decrease is observed when the SOE is high.

Table 5.4: Performance of the first model on Kartal instances with lower cleaning times

Instances Features	Instance #	optimal value	CPU time(Cplex) MTT	CPU time(Cplex) MTE [3]	# of cleaned arcs
SOE=1, # of critical nodes=7, cleaning effort : c'_{ij}	K1′	44	48.37	219.89	0
	K2′	43	45.05	213.25	0
	K3′	44	43.73	192.61	0
	K4′	43	51.62	156.37	0
	K5′	43	51.45	152.9	0
SOE=2, # of critical nodes=7, cleaning effort : c'_{ij}	K6′	48	55.27	231.4	0
	K7′	49	67.33	210.34	1
	K8′	51	70.75	282.31	0
	K9′	49	79.59	259.62	0
	K10′	48	66.77	210.25	0
SOE=3, # of critical nodes=7, cleaning effort : c'_{ij}	K11′	51	91.29	214.86	1
	K12′	63	125.43	316.91	0
	K13′	67	145.93	336.34	1
	K14′	46	66.84	198.85	0
	K15′	47	62.44	184.67	0
SOE=4, # of critical nodes=7, cleaning effort : c'_{ij}	K16′	97	488.17	4864.74	3
	K17′	78	119.8	2422.78	1
	K18′	95	513.36	3791.44	3
	K19′	81	120.31	2794.34	2
	K20′	80	373.64	3258.78	3

The performance of the two mathematical models, MTT and MTE are summarized in Table 5.5 where it can be seen that for all instances from Kartal, same optimal solutions are reached 8 times faster on the average by our model MTT.

Table 5.5: CPU times of models Minimize Total Effort [3] and Minimize Total Time (in seconds)

	Minimize Total Effort [3]			Minimize Total Time		
Instances	min	avg	max	min	avg	max
K1 ... K20	178.4	1441.73	9136.87	39.07	167.18	616.1
K1′ ... K20′	152.9	1025.63	4864.74	43.73	134.36	513.36

The analyses on Bakırköy instances with the first model, MTT, lead to similar conclusions. When the severity of the earthquake increases, and consequently the number of blocked arcs and time required for debris removal increase, CPU times and optimal values rise. As seen in Tables 5.6, 5.7, 5.8 and 5.9, the amount of increase in the optimal values and CPU times vary according to the number of critical nodes and their locations. For example when the severity rises from 1 to 4, the increase in the objective function is higher for the instances where the critical node set consist of schools compared to the instances where only hospitals are critical.

When we compare the instances concerned with the schools and hospitals, we see that the location of the schools results in higher CPU time. For example, with instance B6 the optimal is reached in 454.17 seconds for schools and in 242.55 seconds for hospital with MTT formulation as seen in Table 5.6. This difference in the CPU times increases when SOE becomes 4 as in the instance B16-B20. The average CPU for these instances where SOE equals to 4 is 4058 seconds for schools and 677 seconds for hospitals. Therefore the location of the critical nodes has a significant effect on the CPU times.

The decrease in the cleaning times has relatively inconsistent effect on the CPU times for Bakırköy instances compared to Kartal. For example from Table 5.6 we see that when the critical set only includes schools, CPU times of B2′ and B4′ are lower than B2 and B4, however B1, B3 and B5 are lower than B1′, B3′ and B5′ respectively. Furthermore not just the number of critical nodes but the size and features of network have an effect on CPU times. When we look at the instance of Kartal and Bakırköy with 7 critical nodes, namely instances K1-K5 and B1-B5 for hospitals, we see that the average CPU time of these Bakırköy instances is 4 times of that of Kartal.

Table 5.6: Performance of the first model on Bakırköy instances with higher and lower cleaning times with SOE=1

Instance Features	# of critical nodes	Instance #	Optimal value	CPU times (Cplex) MTT	CPU times (Cplex) MTE [3]	# of cleaned arcs
SOE=1 cleaning effort c_{ij}	8 (schools)	B1	52	454.17	1981.32	-
		B2	61	766.38	3433.06	1
		B3	52	463.86	2586.69	-
		B4	54	528.97	2235.72	-
		B5	52	480.68	1753.38	-
	7 (hospitals)	B1	41	242.55	921.03	-
		B2	38	215.5	761.61	-
		B3	39	235.79	646.14	-
		B4	40	293.67	821.21	-
		B5	40	338.96	728.84	-
SOE=1 cleaning effort c'_{ij}	8 (schools)	B1′	52	775.45	1920.36	-
		B2′	58	517.7	2834.09	1
		B3′	52	563.33	1849.6	-
		B4′	54	357.18	2194.44	-
		B5′	52	883.73	1638.53	-
	7 (hospitals)	B1′	41	297.92	736.43	-
		B2′	38	214.37	706.57	-
		B3′	39	245.09	768.61	-
		B4′	40	238.23	649.6	-
		B5′	40	224.49	845.63	-

Results of models MTT and MTE are illustrated when SOE equals to 1 and critical node sets are schools and hospitals separately in Table 5.6. Similar to Kartal results we see that no blocked arc is cleaned when the severity of the earthquake is low except the instances B2 and B2′ where the critical nodes are only schools.

As the severity increases the number of cleaned arcs increases in these instances similar to Kartal. For the cases where hospitals are critical, blocked arcs are cleaned only when the severity is at the highest level. However, in Bakırköy instances where the critical nodes are only schools, we see that cleaning occurs at each SOE level from the Tables 5.6, 5.7, 5.8 and 5.9.

When SOE becomes 2, as seen in Table 5.7 MTE reports a gap for instance B7 where critical nodes are only schools. The best objective found by MTE is actually the optimal solution but the optimality cannot be assured by this model in 4 hours. Our model, MTT, finds optimal solution for this instance in less than an hour.

Table 5.7: Performance of the first model on Bakırköy instances with higher and lower cleaning times with SOE=2

		Minimize Total Time				Minimize Total Effort [3]			
Instance features	Instance #	Best obj.	Gap %	CPU time	# of cleaned arcs	Best obj.	Gap %	CPU time	# of cleaned arcs
SOE=2 cleaning effort c_{ij}, 8 (schools)	B6	52	0.00	1389.95	-	52	0.00	1647.38	-
	B7	60	0.00	2549.11	-	60	0.00	13108.3	-
	B8	52	0.00	1050.84	-	52	0.00	2126.97	-
	B9	60	0.00	1877.6	-	60	0.00	12207.8	-
	B10	52	0.00	425.77	-	52	0.00	12236.8	-
SOE=2 cleaning effort c_{ij}, 7 (hospitals)	B6	39	0.00	207.74	-	39	0.00	762.91	-
	B7	39	0.00	165.41	-	39	0.00	686.03	-
	B8	40	0.00	197.05	-	40	0.00	889.07	-
	B9	42	0.00	294.85	-	42	0.00	647.82	-
	B10	38	0.00	173.02	-	38	0.00	600.08	-
SOE=2 cleaning effort c'_{ij}, 8 (schools)	B6′	52	0.00	501.47	-	52	0.00	2785.6	-
	B7′	56	0.00	2432.34	1	56	26.79	14400	1
	B8′	52	0.00	437.61	-	52	0.00	2245.7	-
	B9′	56	0.00	2175.61	1	56	0.00	12997	1
	B10′	52	0.00	1502.76	-	52	0.00	1971.1	-
SOE=2 cleaning effort c'_{ij}, 7 (hospitals)	B6′	39	0.00	189.75	-	39	0.00	744.54	-
	B7′	39	0.00	180.87	-	39	0.00	708.93	-
	B8′	40	0.00	349.11	-	40	0.00	794.42	-
	B9′	42	0.00	239.42	-	42	0.00	729.86	-
	B10′	38	0.00	218.22	-	38	0.00	760.88	-

When SOE increases to 3, MTE reports gaps for all instances where schools are the critical nodes and our model MTT reaches optimal solution in less than an hour as seen in Table 5.8. These results again prove the effectiveness of our model

MTT and also highlight the importance of the critical nodes' locations with the great difference in the CPU times of the instances where the critical node sets are different.

Table 5.8: Performance of the first model on Bakırköy instances with higher and lower cleaning times with SOE=3

			Minimize Total Time			Minimize Total Effort [3]			
Instance features	Instance #	Best obj.	Gap %	CPU time	# of cleaned arcs	Best obj.	Gap %	CPU time	# of cleaned arcs
SOE=3 cleaning effort c_{ij}, 8 (schools)	B11	71	0.00	1439.26	-	71	0.00	12711	-
	B12	80	0.00	2770.89	-	80	43.77	14400	-
	B13	74	0.00	1038.03	-	74	0.00	13736.36	-
	B14	71	0.00	731.99	-	71	0.00	12507.97	-
	B15	77	0.00	2460.8	-	77	33.77	14400	-
SOE=3 cleaning effort c_{ij}, 7 (hospitals)	B11	39	0.00	136.22	-	39	0.00	461.74	-
	B12	42	0.00	190.2	-	42	0.00	550.53	-
	B13	46	0.00	206.05	-	46	0.00	580.23	-
	B14	48	0.00	203.78	-	48	0.00	744.75	-
	B15	43	0.00	172.18	-	43	0.00	540.02	-
SOE=3 cleaning effort c'_{ij}, 8 (schools)	B11′	67	0.00	3112.28	1	67	34.0	14400	1
	B12′	74	0.00	3456.95	2	75	60.0	14400	2
	B13′	73	0.00	2717.89	1	73	57.5	14400	1
	B14′	69	0.00	1747.06	1	69	27.5	14400	1
	B15′	75	0.00	1128.25	1	75	50.7	14400	1
SOE=3 cleaning effort c'_{ij}, 7 (hospitals)	B11′	39	0.00	227.11	-	39	0.00	618.33	-
	B12′	42	0.00	203.41	-	42	0.00	644.19	-
	B13′	46	0.00	239.16	-	46	0.00	818.27	-
	B14′	48	0.00	291.72	-	48	0.00	1370.15	-
	B15′	43	0.00	252.89	-	43	0.00	703.36	-

Similar to the results in Kartal instances when SOE is at the highest level, the CPU times are also the highest. As seen in Table 5.9, MTE again reports gaps while our model MTT finds optimal solutions in 2 hours for the instances where the critical nodes are only schools. For all instances where the critical nodes are hospitals, both models are able to reach optimality in 2 hours. When the severity is the highest, the cleaning effort is low and the critical nodes are only hospitals, the average CPU time of MTE is 4963 seconds where our model solves instances optimally in 580 seconds on the average.

Table 5.9: Performance of the first model on Bakırköy instances with higher and lower cleaning times with SOE=4

			Minimize Total Time			Minimize Total Effort [3]			
Instance features	Instance #	Best obj.	Gap %	CPU time	# of cleaned arcs	Best obj.	Gap %	CPU time	# of cleaned arcs
SOE=4 cleaning effort c_{ij}, 8 (schools)	B16	96	0.00	5271.01	-	96	71.01	14400	-
	B17	78	0.00	1971.88	-	78	31.71	14400	-
	B18	112	0.00	6974.84	-	112	74.64	14400	-
	B19	84	0.00	2974.57	-	84	45.24	14400	-
	B20	87	0.00	3098.48	-	87	63.22	14400	-
SOE=4 cleaning effort c_{ij}, 7 (hospitals)	B16	61	0.00	1197.11	-	61	0.00	7279.04	-
	B17	58	0.00	1437.28	-	58	0.00	7653.28	-
	B18	51	0.00	211.69	-	51	0.00	1162.77	-
	B19	59	0.00	329.28	-	59	0.00	6006.05	-
	B20	52	0.00	208.13	-	52	0.00	848.49	-
SOE=4 cleaning effort c'_{ij}, 8 (schools)	B16′	87	0.00	5118.85	3	91	69.71	14400	1
	B17′	78	0.00	2633.39	-	78	37.18	14400	-
	B18′	104	0.00	6676.95	4	105	73.69	14400	3
	B19′	79	0.00	3649.5	1	79	44.07	14400	1
	B20′	83	0.00	2826.31	2	83	67.85	14400	2
SOE=4 cleaning effort c'_{ij}, 7 (hospitals)	B16′	59	0.00	444.2	1	59	0.00	7433.39	1
	B17′	57	0.00	497.27	1	57	0.00	7577.35	1
	B18′	51	0.00	375.97	-	51	0.00	1352.84	-
	B19′	55	0.00	1189.86	1	55	0.00	7261.47	1
	B20′	52	0.00	393.92	-	52	0.00	1191.86	-

For the instances where the critical nodes are only schools or hospitals, the improvement of MTT in the CPU times compared to MTE is obvious. However when all the critical nodes are included, both of the models cannot find optimal solutions in a reasonable amount of time as seen in Table 5.10. When the number of critical nodes is 15, for most of the instances MTT finds better objective values however there is no consistency. Even when only schools are concerned, if the severity of the earthquake is high, reaching optimal may take almost 2 hours as in the instance B18 and B18′. Therefore, for Bakırköy instances with 15 critical nodes, we only use the ones with the lower cleaning effort and see that optimality cannot be reached in 2 hours by MTT.

Table 5.10: Performance of the first model on Bakırköy instances with lower cleaning times and 15 critical nodes

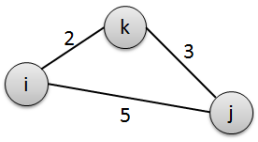
Instance features	Instance #	Best obj.	Minimize Total Time			Minimize Total Effort [3]			
			Gap %	CPU time	# of cleaned arcs	Best obj.	Gap %	CPU time	# of cleaned arcs
SOE=1 cleaning effort cij. critical nodes 15	B1	76	90.79	7200	-	74	91.9	14400	-
	B2	78	92.31	7200	1	84	92.9	14400	2
	B3	74	90.54	7200	-	78	92.3	14400	-
	B4	73	89.04	7200	-	80	92.5	14400	-
	B5	74	90.54	7200	-	73	91.8	14400	-
SOE=2 cleaning effort cij. critical nodes 15	B6	82	85.37	7200	-	75	92.0	14400	-
	B7	79	93.67	7200	-	80	90.0	14400	1
	B8	90	95.56	7200	-	77	92.2	14400	-
	B9	109	100.00	7200	-	81	92.6	14400	1
	B10	80	88.75	7200	-	76	92.1	14400	-
SOE=3 cleaning effort cij. critical nodes 15	B11	97	93.81	7200	-	95	91.6	14400	-
	B12	97	94.85	7200	-	98	91.8	14400	1
	B13	96	95.44	7200	-	100	92.0	14400	-
	B14	104	97.12	7200	-	98	90.8	14400	1
	B15	95	94.74	7200	1	99	90.9	14400	1
SOE=4 cleaning effort cij. critical nodes 15	B16	115	100.00	7200	1	135	92.6	14400	-
	B17	122	100.00	7200	-	131	100.0	14400	-
	B18	125	96.00	7200	2	165	95.2	14400	-
	B19	114	94.52	7200	-	134	99.3	14400	-
	B20	114	94.64	7200	1	328	99.7	14400	6

Analyses on the Transportation Network and Solvers

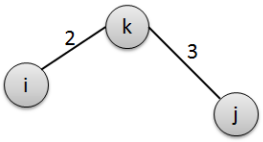
As stated in Data section complete networks are used in the test of our first model MTT. Because of the completeness there are more than one path with the same distance between same nodes. While keeping the original path we eliminate the arc with the same distances to find the transportation network. For example if the shortest path between nodes i and j is $i-k-l-j$ with distance d, we eliminate the arc (i,j) if its travel distance is d and we keep it if it less than d. The aim is to decrease the size of the data set and compare the effect of using complete network and transportation network.

To decide which arcs to eliminate and which to keep first we find the minimum spanning tree of the complete network by Prim algorithm. Then, shortest path

among all nodes are found by Floyd-Warshall algorithm. We compare the distance matrix obtained from the shortest path with the distance matrix of the complete network to construct an incidence matrix T. If the distance between i-j in the first matrix equal to the distance between i-j in the complete network and if arc (i, j) is not in the spanning tree we make $T_{ij} = 0$ and else $T_{ij} = 0$. This way we eliminate arc (i, j) and (j, i) in the transportation network. A small example on a network with 3 nodes can be seen in Figure 5.4.



(a) Complete network



(b) Transportation network

$$T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

(c) T matrix

Figure 5.4: Example of constructing transportation network and T matrix

To test our first model with the transportation network we add the following constraints to the model so that an arc which is not included in T matrix cannot be traversed and cleaned.

$$\sum_{k \in C} x_{ij}^k \leq |C| T_{ij} \qquad \forall i, j \in N' \tag{5.1}$$

$$B_{ij} \leq T_{ij} \qquad \forall i, j \in N' \tag{5.2}$$

$$cb_{ij} \leq T_{ij} \qquad \forall i, j \in N \tag{5.3}$$

In order to see the effect of using the transportation network, we solve Kartal instances with higher cleaning effort, K1-K20, with the modified MTT. Furthermore, we test original MTT and modified version with the transportation network both with Cplex 12.6 and Gurobi 5. As seen in Table 5.11 a clear improvement cannot be observed in the CPU times by using transportation network. Moreover for the first model Cplex 12.6 reaches optimality faster than Gurobi therefore all analyses are conducted with the complete network using Cplex 12.6 for the first problem.

Table 5.11: Performances of Cplex 12.6 and Gurobi 5 in terms of CPU times on Kartal instances with higher cleaning times with complete and transportation networks

Instance	Complete Network		Transportation Network	
#	Cplex 12.6	Gurobi 5	Cplex 12.6	Gurobi 5
K1	39.07	73.15	42.05	258.36
K2	50.25	99.6	48.69	80.3
K3	51.13	90.35	52.98	129.52
K4	43.04	102.52	52.67	121.23
K5	42.63	101.1	55.94	119.47
K6	49.25	91.41	55.81	112.85
K7	56.34	74.78	93.9	118.76
K8	58.22	98.11	62.45	132.24
K9	68.92	124.66	73.6	125.01
K10	41.44	140.03	53.05	93.57
K11	71.55	97.92	66.74	109.87
K12	162.14	190.53	130.2	230.07
K13	114.85	203.25	126.82	199.63
K14	57.72	122.43	84.93	149.74
K15	58.17	66.82	59.04	127.61
K16	616.1	582.98	709.7	804.06
K17	372.72	290.53	157.77	327.16
K18	551.87	525.37	597.62	425.53
K19	348.14	393.91	394.65	500
K20	490.14	623.4	227.77	410.32

5.2.2 Analyses on the Second Model

For the second model same data sets and instances are used by assigning appropriate weights to the critical nodes as explained in the Data section.

Our preliminary analyses show that for the second model Gurobi gives better CPU times than Cplex. The instance K1-K20 are solved for 2 hours using both solvers. Out of 20 instances Cplex finds 18 optimal solutions in 2 hours while Gurobi reaches all optimal solutions with average CPU time of 683 seconds as

seen in Table 5.12. Therefore only Gurobi 5 is used for the test of the second model.

Table 5.12: Performances of Cplex 12.6 and Gurobi 5 on Kartal instances with higher cleaning times for the second model

Instance	Cplex 12.6		Gurobi 5	
#	Gap %	CPU time	Gap %	CPU time
K1	0.00	70.25	0.00	77.9
K2	0.00	94.15	0.00	79.92
K3	0.00	88.21	0.00	170.13
K4	0.00	60.27	0.00	180.57
K5	0.00	60.57	0.00	173
K6	0.00	476.06	0.00	381.84
K7	0.00	314.06	0.00	336.59
K8	0.00	332.62	0.00	340.94
K9	0.00	741.95	0.00	423.35
K10	0.00	430.96	0.00	384.81
K11	0.00	1460.1	0.00	490.67
K12	0.00	311.29	0.00	503.7
K13	0.00	1246.04	0.00	721.46
K14	0.00	379.09	0.00	654.75
K15	0.00	245.1	0.00	709.89
K16	0.00	5874.1	0.00	1668.84
K17	0.00	4067.4	0.00	1536.69
K18	90.12	7200	0.00	2187.08
K19	91.59	7200	0.00	1334.81
K20	0.00	2286.1	0.00	1298.63

The results of the second model, MWSVT, on Kartal instances with higher and lower cleaning efforts are presented in Tables 5.13 and 5.14 respectively. From these tables we see that debris removal occurs when the severity of the earthquake is higher. When the cleaning effort is high, blocked arcs are cleaned only when SOE equals to 4. When the cleaning effort is low, more blocked arcs are cleaned. For example in the solution of instance K19 only 1 arc is cleaned where 3 arcs are cleaned in K19′. Furthermore, similar to the previous results, when SOE increases, the CPU times and optimal values increase as well as seen in the test

of the second model on Kartal instances.

Table 5.13: Performance of the second model on Kartal instances with higher cleaning times

Instances Features	Instance #	optimal value	CPU time Gurobi	# of cleaned arcs
SOE=1, # of critical nodes=7, cleaning effort : c_{ij}	K1	2035	77.9	-
	K2	1949	79.92	-
	K3	2035	170.13	-
	K4	1949	180.57	-
	K5	1949	173	-
SOE=2, # of critical nodes=7, cleaning effort : c_{ij}	K6	2123	381.84	-
	K7	2135	336.59	-
	K8	2258	340.94	-
	K9	2197	423.35	-
	K10	2123	384.81	-
SOE=3, # of critical nodes=7, cleaning effort : c_{ij}	K11	2698	490.67	-
	K12	2862	503.7	-
	K13	3225	721.46	-
	K14	2169	654.75	-
	K15	2128	709.89	-
SOE=3, # of critical nodes=7, cleaning effort : c_{ij}	K16	5376	1668.84	1
	K17	4684	1536.69	-
	K18	5273	2187.08	1
	K19	4909	1334.81	1
	K20	4301	1298.63	1

In the previous subsection from Tables 5.3 and 5.4 we see that when SOE is 1 and 2, the optimal values are the same for the same instances where only the cleaning effort differs, i.e, objective values of instances K1-K10 are the same with K1′-K10′. The optimal values start to differ when SOE equals to 3 and become completely different when SOE is 4. We see the same pattern in the results of the second model with Kartal instances from Tables 5.13 and 5.14, i.e., with the first two SOE value the objective values are identical and they are completely different when SOE is highest.

Table 5.14: Performance of the second model on Kartal instances with lower cleaning times

Instances Features	Instance #	optimal value	CPU time Gurobi	# of cleaned arcs
SOE=1, # of critical nodes=7, cleaning effort : c'_{ij}	K1′	2035	95.83	-
	K2′	1949	124.7	-
	K3′	2035	102.15	-
	K4′	1949	99.36	-
	K5′	1949	137.65	-
SOE=2, # of critical nodes=7, cleaning effort : c'_{ij}	K6′	2123	387.9	-
	K7′	2135	426.63	-
	K8′	2258	431.03	-
	K9′	2197	358.51	-
	K10′	2123	422.57	-
SOE=3, # of critical nodes=7, cleaning effort : c'_{ij}	K11′	2564	722.86	1
	K12′	2860	806.2	1
	K13′	3225	624.41	1
	K14′	2169	605.93	-
	K15′	2128	513.3	-
SOE=3, # of critical nodes=7, cleaning effort : c'_{ij}	K16′	4775	1741.48	2
	K17′	4320	1495.75	2
	K18′	4759	1956.3	3
	K19′	4321	1717.03	1
	K20′	3502	1615.06	2

To be able to see the effect of the weights, we compare the total times and routes given by two mathematical models for instances K1-K20. The weights of the critical nodes for Kartal instances can be seen in Table 5.15.

Table 5.15: Weights of the critical nodes in Kartal data set

critical nodes	14	21	22	26	33	41	43
weights	22	15	15	8	17	17	5

The total route times of the optimal solutions found by both models are presented in Table 5.16. From this table we see that for each level of SOE some values are the same with the first model and some are higher. This implies that the change in the route and total route time in the weighted case, significantly depends on

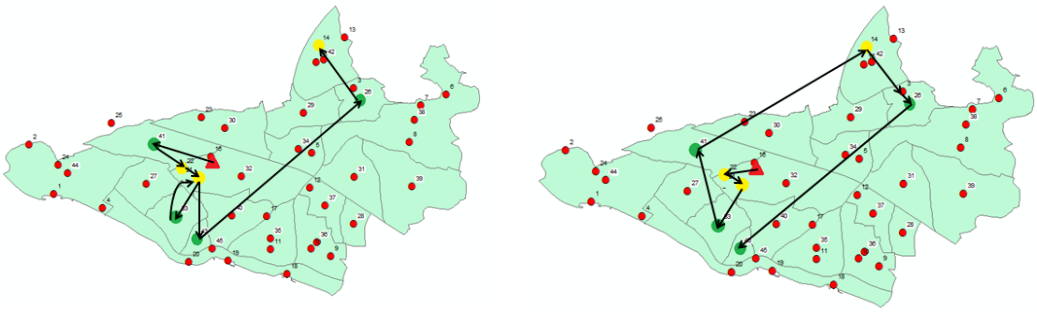
the locations of the blocked arcs together with the weights. For example, the only difference in instance K11 and K13 are the location of the blocked arcs but for the first one both of the models reach optimal solutions with total route time equal to 53 while for instance K13, the total route times are 68 and 72 for first and second problem respectively.

Table 5.16: Total route times of optimal solutions obtained from the mathematical models using Kartal instances with higher cleaning times

Instances Features	Instance #	Total time of the route by MTT	Total time of the route by MWSVT
SOE=1, # of critical nodes=7, cleaning effort : c_{ij}	K1	44	44
	K2	43	44
	K3	44	47
	K4	43	44
	K5	43	44
SOE=2, # of critical nodes=7, cleaning effort : c_{ij}	K6	48	48
	K7	50	63
	K8	51	52
	K9	49	49
	K10	48	48
SOE=3, # of critical nodes=7, cleaning effort : c_{ij}	K11	53	53
	K12	63	75
	K13	68	82
	K14	46	46
	K15	47	52
SOE=4, # of critical nodes=7, cleaning effort : c_{ij}	K16	109	117
	K17	82	85
	K18	110	110
	K19	90	93
	K20	101	115

When we look at the instances K1 and K3, from Table 5.16 we see that the total route times are different; 44 and 47 respectively. However, from Table 5.14 it can be seen that the optimal values for the weighted case for these instances are the same which is 2035. This shows that same objective value for the second model does not necessarily imply that the optimal solutions have the same route and/or total route time.

Out of 20 instances of Kartal with high cleaning effort, optimal routes found by the first and second model are the same for 7 of them. Instance K7 is one of them with different optimal routes and its solutions are presented in Figure 5.5. Node 14 is relatively far from other critical nodes and from the supply node, and it is usually the last visited node for Kartal instance. However in the weighted version of the problem, node 14 is visited in the 5th order instead of 7th for instance K7. Hence, the weights, together with the other features of the network, can change the optimal route.



(a) Optimal route of K7 for MTT

(b) Optimal route of K7 for MWSVT

Figure 5.5: Optimal route of instance K7 for both of the mathematical models

The second model is also tested by Bakırköy data set. In the analyses on the first model, we see that taking both schools and hospitals as critical increases the complexity and optimal solutions cannot be reached in reasonable time. Since the second model is larger than first one, hospitals and schools are taken in the critical node set separately for the analyses on the second model. The weights of the critical nodes for Bakırköy instances are represented in Tables 5.17.

Table 5.17: Weights of the critical nodes in Bakırköy (hospitals on the left, schools on the right)

critical nodes	16	17	18	19	20	21	22
weights	13	22	13	13	22	13	4

critical nodes	5	15	34	36	47	55	65	67
weights	17	15	4	5	17	24	5	14

Tables 5.18 and 5.19 show optimal value and CPU times of Bakırköy instances in which only hospitals are critical. We see that in the most of cases the CPU times are lower when the cleaning effort is lower; when SOE is less than 4, all of the instances with lower cleaning times are solved faster than the ones with higher cleaning times.

The optimal values between high and low cleaning times only differ when the severity of the earthquake is at the highest value. Out of 20 instances, only 3 of them are affected from the change in the cleaning effort, namely instance B16, B17, B19 have different objective values than instance B16′, B17′, B19′ respectively.

Table 5.18: Performance of the second model on Bakırköy instances (hospitals) with higher cleaning times

Instance Features	Instance #	Optimal value	CPU time Gurobi	# of cleaned arcs
SOE=1 cleaning effort c_{ij}, 7 (hospitals)	B1	1979	338	-
	B2	1876	355.64	-
	B3	1884	531.34	-
	B4	1862	534.37	-
	B5	1920	318.56	-
SOE=2 cleaning effort c_{ij}, 7 (hospitals)	B6	1884	403.22	-
	B7	1907	541.26	-
	B8	1924	642.64	-
	B9	1897	722.25	-
	B10	1876	704.38	-
SOE=3 cleaning effort c_{ij}, 7 (hospitals)	B11	1875	2241.83	-
	B12	1884	1663.46	-
	B13	2169	1925.84	-
	B14	2362	2419.83	-
	B15	1970	2080.29	-
SOE=4 cleaning effort c_{ij}, 7 (hospitals)	B16	2430	4135.24	-
	B17	2671	4890.86	-
	B18	2361	4833.86	-
	B19	2672	6275.62	-
	B20	2483	4944.91	-

When the cleaning times are higher, no blocked arc is cleaned in the optimal paths as seen in Table 5.18. When the cleaning times are lower, one blocked arc is cleaned in the optimal solutions of the instances B16′, B17′, B19′ as seen in Table 5.19.

Table 5.19: Performance of the second model on Bakırköy instances (hospitals) with lower cleaning times

Instance Features	Instance #	Optimal value	CPU time Gurobi	# of cleaned arcs
SOE=1 cleaning effort c'_{ij}, 7 (hospitals)	B1′	1979	316.54	-
	B2′	1876	294.52	-
	B3′	1884	335.47	-
	B4′	1862	255.17	-
	B5′	1920	287.19	-
SOE=2 cleaning effort c'_{ij}, 7 (hospitals)	B6′	1884	348.97	-
	B7′	1907	444.67	-
	B8′	1924	410.58	-
	B9′	1897	438.99	-
	B10′	1876	371.99	-
SOE=3 cleaning effort c'_{ij}, 7 (hospitals)	B11′	1875	1394.25	-
	B12′	1884	1330.03	-
	B13′	2169	1254.56	-
	B14′	2362	1145.38	-
	B15′	1970	1210.83	-
SOE=4 cleaning effort c'_{ij}, 7 (hospitals)	B16′	2426	4963.48	1
	B17′	2655	4981.34	1
	B18′	2361	4518.54	-
	B19′	2570	5621.99	1
	B20′	2483	6060.65	-

When we compare the total route times of Bakırköy instances where critical nodes are only hospitals and the cleaning effort is high, we see that 5 out of 20 instances have the same total route time for models MTT and MWSVT. Thus we can say that the weights have an impact in most of the instances where the critical set consists of hospitals.

When the critical nodes become only schools it gets harder to reach optimality. As seen in Table 5.20, optimal solutions are found for 9 out of 20 instances when the cleaning effort is high and from Table 5.21 we see that half of instances are solved to optimality when the cleaning effort is low. As stated in the analyses of the first model, the locations of schools have a tremendous effect on the solutions times. For the second model, the optimal solutions cannot be reached in 2 hours especially when the severity level is high.

Table 5.20: Performance of the second model on Bakırköy instances (schools) with higher cleaning times

Instance Features	Instance #	Best objective	Gap %	CPU time Gurobi	# of cleaned arcs
SOE=1 cleaning effort c_{ij}, 8 (schools)	B1	2779	0.00	1721.08	-
	B2	3439	0.00	3738.91	1
	B3	2779	0.00	2442.78	-
	B4	2847	0.00	3051.61	-
	B5	2779	0.00	6052.22	-
SOE=2 cleaning effort c_{ij}, 8 (schools)	B6	2779	0.00	3605.48	-
	B7	3591	77.28	7200	-
	B8	2779	0.00	4499.47	-
	B9	3556	0.00	5646.93	-
	B10	2779	0.00	3676.88	-
SOE=3 cleaning effort c_{ij}, 8 (schools)	B11	3802	90.90	7200	-
	B12	4376	89.90	7200	-
	B13	4131	91.43	7200	-
	B14	4017	89.15	7200	-
	B15	4199	92.99	7200	-
SOE=4 cleaning effort c_{ij}, 8 (schools)	B16	5125	95.82	7200	-
	B17	3928	97.66	7200	-
	B18	6105	98.33	7200	-
	B19	4426	100.00	7200	-
	B20	4030	96.87	7200	-

We investigate the routes and the total route times of the instances for which the optimal solutions are found and we compare them with the solutions of the first model. This comparison shows that the total route times are the same for

the instances we solve to optimality. When the cleaning effort is low the optimal solutions of the first 10 instances by both models have the same total route time. However this does not necessarily mean that the reported solutions are the same. For example in the solution of instance B2′ with low cleaning time by the first model there is one cleaned arc as presented in Table 5.6, however, in the solution of the same instance by the second model there are two cleaned arcs as seen in Table 5.21. Although the number of cleaned arcs are different, both solutions have a total route time which equal to 58 so there are alternative optimal solutions for the first model. These solutions are also alternative optimal solutions for the second model if the visiting times of the critical nodes are the same in these paths, and they are for instance B2′. These alternative optimal paths are shown in Figure 5.6 and 5.7.

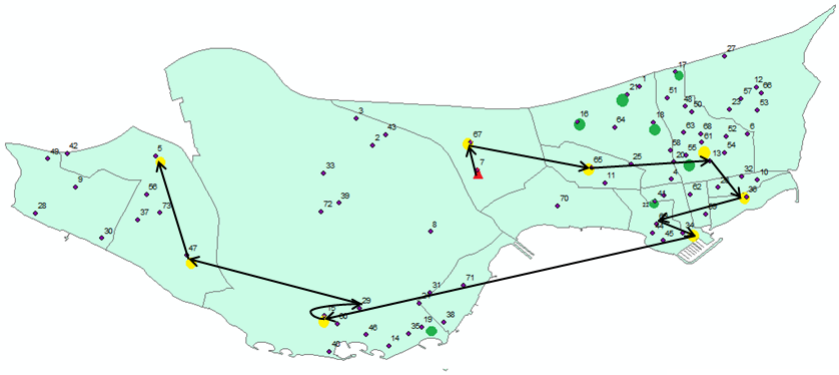


Figure 5.6: Optimal route of B2′ (schools) by MTT

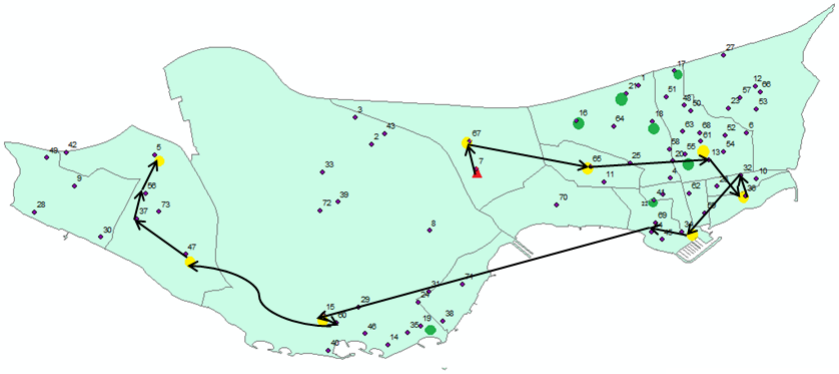


Figure 5.7: Optimal route of B2′ (schools) by MWSVT

Table 5.21: Performance of the second model on Bakırköy instances (schools) with lower cleaning times

Instance Features	Instance #	Best objective	Gap %	CPU time Gurobi	# of cleaned arcs
SOE=1 cleaning effort c'_{ij}, 8 (schools)	B1′	2779	0.00	1629.69	-
	B2′	3136	0.00	2778.12	2
	B3′	2779	0.00	2711.83	-
	B4′	2847	0.00	1793.75	-
	B5′	2779	0.00	2862.37	-
SOE=2 cleaning effort c'_{ij}, 8 (schools)	B6′	2779	0.00	1946.02	-
	B7′	3183	0.00	3475.90	1
	B8′	2779	0.00	3490.22	-
	B9′	3183	0.00	3325.31	1
	B10′	2779	0.00	3347.79	-
SOE=3 cleaning effort c'_{ij}, 8 (schools)	B11′	3802	83.90	7200	-
	B12′	4232	86.51	7200	2
	B13′	4045	88.23	7200	1
	B14′	3898	81.35	7200	1
	B15′	4028	84.51	7200	2
SOE=4 cleaning effort c'_{ij}, 8 (schools)	B16′	4927	97.16	7200	1
	B17′	3725	99.19	7200	1
	B18′	5303	98.08	7200	3
	B19′	4452	97.87	7200	3
	B20′	4194	96.76	7200	2

A similar example occurs in the instance B9 in which the critical nodes are schools and cleaning effort is high. The optimal solutions of this instance by first and second model have the same total route time but the routes are different. Since the visiting time of the critical nodes are different in these route, it implies that the routes are alternative optimal solutions for the first model but not for the second model. When we calculate the objective function of MWSVT using the optimal route for MTT, we get 3697 while the optimal solution for MWSVT is 3556 as seen in Table 5.20.

5.2.3 Performance of Minratio Heuristic

As the network enlarges and the number of critical nodes increase it is not possible to reach optimal solution in a reasonable amount of time for some cases. Computational studies on Bakırköy show that the number of critical nodes and their locations have a direct effect on the CPU times. Moreover especially with the higher severity degrees solvers cannot find optimal solutions and they report large gaps in 2 hours for some of the instances.

For *Debris Removal Problem in Response Phase* in order to find near-optimal solutions quickly we develop an algorithm called Minratio. As explained in the previous chapter, the algorithm calculates a ratio for each critical node k, by dividing min_k by avg_k; min_k is the distance of the closest critical node to k and avg_k is the average distance from k to all eligible critical nodes. Minratio chooses the critical node giving the minimum ratio and connects it to its closest critical node. To decide which critical node to choose, different rules can be generated and in addition to Minratio, two different algorithms are generated and tested with Kartal data set. The first one is Maxratio which picks the critical node giving the maximum ratio and the other one is Avgratio which picks the critical node giving the minimum or maximum ratio whichever is the more far from the average of the ratios. Performances of these heuristics are presented in Table 5.22 using Kartal instances with higher cleaning effort. Since Minratio gives better solutions compared to the others, the rest of the analyses are conducted with Minratio heuristic.

Table 5.22: Comparison of Minratio, Maxratio and Avgratio on Kartal instances with higher cleaning times

Instance #	Optimal value	Minratio value	Maxratio value	Avgratio Value
K1	44	44	63	44
K2	43	43	43	43
K3	44	44	60	61
K4	43	43	43	43
K5	43	43	43	43
K6	48	48	53	51
K7	50	50	50	50
K8	51	51	69	59
K9	49	49	74	74
K10	48	48	49	58
K11	53	53	59	55
K12	63	66	83	83
K13	68	68	82	74
K14	46	46	64	64
K15	47	47	50	50
K16	109	112	134	127
K17	82	82	115	99
K18	110	110	139	118
K19	90	90	102	102
K20	101	109	136	136

Table 5.23 illustrates the performance of Minratio algorithm for all Kartal instances. From this results we see that as SOE increases, the solutions found by the heuristics go far from the optimal. Except instance K20 and especially K20′ the heuristics either find the optimal solution or a near one with gaps less than 3%.

As summarized in Table 5.24 the algorithm finds the optimal solutions for 85% and 70% of Kartal instances with higher and lower cleaning efforts respectively. From the same table it can be seen that Minratio heuristic finds one more optimal solution compared to the heuristic developed by Şahin [3]. The maximum gaps are the same for both heuristics but Minratio outperforms in terms of average gaps.

Table 5.23: Performance of Minratio heuristic on Kartal instances with higher and lower cleaning times

	Optimal	Minratio Heuristic			Optimal	Minratio Heuristic	
Instance #	Value	Value	Gap %	Instance #	Value	Value	Gap %
K1	44	44	0.00	K1′	44	44	0.00
K2	43	43	0.00	K2′	43	43	0.00
K3	44	44	0.00	K3′	44	44	0.00
K4	43	43	0.00	K4′	43	43	0.00
K5	43	43	0.00	K5′	43	43	0.00
K6	48	48	0.00	K6′	48	48	0.00
K7	50	50	0.00	K7′	49	49	0.00
K8	51	51	0.00	K8′	51	51	0.00
K9	49	49	0.00	K9′	49	51	4.08
K10	48	48	0.00	K10′	48	48	0.00
K11	53	53	0.00	K11′	51	51	0.00
K12	63	66	4.76	K12′	63	63	0.00
K13	68	68	0.00	K13′	67	68	1.49
K14	46	46	0.00	K14′	46	46	0.00
K15	47	47	0.00	K15′	47	47	0.00
K16	109	112	2.75	K16′	97	99	2.06
K17	82	82	0.00	K17′	78	78	0.00
K18	110	110	0.00	K18′	95	97	2.10
K19	90	90	0.00	K19′	81	82	1.23
K20	101	109	7.92	K20′	80	95	18.75

When the cleaning effort is higher, heuristic by Şahin [3] finds 16 optimal solutions while Minratio finds 17. If these heuristics are used together, 19 instances would be solved optimally out of 20. When the cleaning effort is lower, these heuristics together solves 15 instances optimally. Hence the optimal ratios would rise to 95% and 75% if the heuristics are combined.

Table 5.24: Performance summary and comparison of Minratio heuristics with the heuristic by Şahin [3] on Kartal instances

	Heuristic by Şahin [3]			Minratio heuristic		
Instances	avg gap	max gap	optimum ratio %	avg gap	max gap	optimum ratio %
K1 ... K20	1.28	7.92	80	0.77	7.92	85
K1′ ... K20′	2.2	18.75	65	1.49	18.75	70

Similar results are observed with Bakırköy instances in which the critical nodes are only schools. In Table 5.25, we see that when SOE is 1 or 2, all instances are solve to optimality by Minratio heuristic. With the increase in SOE, the solutions found by the heuristics start to go far from the optimal.

Table 5.25: Performance of Minratio heuristic on Bakırköy instances with higher and lower cleaning times, critical nodes are schools

	Optimal	Minratio Heuristic			Optimal	Minratio Heuristic	
Instance #	Value	Value	Gap %	Instance #	Value	Value	Gap %
B1	52	52	0.00	B1′	52	52	0.00
B2	61	61	0.00	B2′	58	58	0.00
B3	52	52	0.00	B3′	52	52	0.00
B4	54	54	0.00	B4′	54	54	0.00
B5	52	52	0.00	B5′	52	52	0.00
B6	52	52	0.00	B6′	52	52	0.00
B7	60	60	0.00	B7′	56	56	0.00
B8	52	52	0.00	B8′	52	52	0.00
B9	60	60	0.00	B9′	56	56	0.00
B10	52	52	0.00	B10′	52	52	0.00
B11	71	72	1.41	B11	67	72	7.46
B12	80	85	6.25	B12′	74	75	1.35
B13	74	74	0.00	B13′	73	73	0.00
B14	71	72	1.41	B14′	69	71	2.9
B15	77	78	1.3	B15′	75	76	1.34
B16	96	103	7.3	B16′	87	92	5.75
B17	78	78	0.00	B17′	78	78	0.00
B18	112	120	7.1	B18′	104	110	5.77
B19	84	89	5.95	B19′	79	86	8.86
B20	87	88	1.15	B20′	83	83	0.00

For Bakırköy instances where the critical nodes are only schools, we again compare our heuristic with the one developed by Şahin [3]. As seen from Table 5.26, for both cleaning levels Şahin's heuristic finds one more optimal solutions compared to Minratio. However, both average and maximum gaps of the solutions found by Minratio are lower than Şahin's. Hence we can say that Minratio outperforms in terms of gaps. Furthermore, if these heuristics are combined they would report 14 and 15 optimal solutions for high and low cleaning efforts respectively.

Table 5.26: Performance summary and comparison of Minratio heuristics with the heuristic by Şahin [3] on Bakırköy instances (schools)

	Heuristic by Şahin [3]			Minratio heuristic		
Instances schools	avg gap	max fap	optimum ratio %	avg gap	max fap	optimum ratio %
B1 ... B20	1.95	19.05	65	1.60	7.29	60
B1′ ... B20′	2.25	24.05	70	1.67	8.86	65

The results of Minratio heuristics on Bakırköy instances in which the critical nodes are only hospitals is shown in Table 5.27.

Table 5.27: Performance of Minratio heuristic on Bakırköy instances with higher and lower cleaning times, critical nodes are hospitals

	Optimal	Minratio Heuristic			Optimal	Minratio Heuristic	
Instance #	Value	Value	Gap %	Instance #	Value	Value	Gap %
B1	41	41	0.00	B1′	41	41	0.00
B2	38	38	0.00	B2′	38	38	0.00
B3	39	39	0.00	B3′	39	39	0.00
B4	40	41	2.50	B4′	40	41	2.50
B5	40	40	0.00	B5′	40	40	0.00
B6	39	39	0.00	B6′	39	39	0.00
B7	39	39	0.00	B7′	39	39	0.00
B8	40	40	0.00	B8′	40	40	0.00
B9	42	42	0.00	B9′	42	42	0.00
B10	38	38	0.00	B10′	38	38	0.00
B11	39	41	5.13	B11′	39	41	5.13
B12	42	42	0.00	B12′	42	42	0.00
B13	46	46	0.00	B13′	46	46	0.00
B14	48	48	0.00	B14′	48	48	0.00
B15	43	48	11.63	B15′	43	48	11.63
B16	61	62	1.64	B16′	59	62	5.08
B17	58	58	0.00	B17′	57	58	1.75
B18	51	52	1.96	B18′	51	52	1.96
B19	59	61	3.39	B19′	55	58	5.45
B20	52	55	5.77	B20′	52	55	5.77

For Bakırköy instances where critical nodes are hospitals Minratio finds 12 optimal solutions out of 20 for both cleaning times while the heuristic developed by Şahin reaches 8 optimal solutions for both cleaning times as seen in Table 5.28. For these sets Minratio outperforms in terms of the number of optimal solutions found and also in terms of the average and maximum gaps.

Table 5.28: Performance summary and comparison of Minratio heuristics with the heuristic by Şahin [3] on Bakırköy instances (hospitals)

	Heuristic by Şahin [3]			Minratio heuristic		
Instances hospitals	avg gap	max fap	optimum ratio %	avg gap	max fap	optimum ratio %
B1 ... B20	3.74	16.28	40	1.60	11.63	60
B1′ ... B20′	3.94	16.28	40	1.96	11.63	60

5.2.4 Performance of Weighted Shortest Distance Heuristic

In order to minimize weighted sum of visiting times, we need additional variables and constraint which increase the CPU times. To find near optimal solutions faster we develop Weighted Shortest Distance heuristic. Its performance on Kartal instances can be seen in Table 5.29. Average gap for Kartal instances with higher cleaning effort is 4% and with lower is 5%.

6 and 5 cases are solved optimally by the algorithm from Kartal instances with high and low cleaning times respectively. With higher SOE values the quality of the solution obtained from WSD decreases. The gap from optimal is larger for instances K17, K19 which indicates impact of the blocked arcs' location.

Table 5.29: Performance of Weighted Shortest Distance(WSD) heuristic on Kartal instances with higher and lower cleaning times

	Optimal	WSD Heuristic			Optimal	WSD Heuristic	
Instance #	Value	Value	Gap %	Instance #	Value	Value	Gap %
K1	2035	2040	0.25	K1′	2035	2040	0.25
K2	1949	1949	0.00	K2′	1949	1949	0.00
K3	2035	2040	0.25	K3′	2035	2040	0.25
K4	1949	1949	0.00	K4′	1949	1949	0.00
K5	1949	1949	0.00	K5′	1949	1949	0.00
K6	2123	2240	5.51	K6′	2123	2228	4.95
K7	2135	2171	1.67	K7′	2135	2171	1.69
K8	2258	2273	0.66	K8′	2258	2273	0.66
K9	2197	2197	0.00	K9′	2197	2197	0.00
K10	2123	2240	5.51	K10′	2123	2240	5.51
K11	2698	2826	4.74	K11′	2564	2726	6.32
K12	2862	2862	0.00	K12′	2860	2860	0.00
K13	3225	3225	0.00	K13′	3225	3225	0.00
K14	2169	2169	0.00	K14′	2169	2169	0.00
K15	2128	2275	6.91	K15′	2128	2275	6.91
K16	5376	5376	0.00	K16′	4775	4775	0.00
K17	4684	5324	13.66	K17′	4320	5088	17.78
K18	5273	5482	3.96	K18′	4759	4935	3.70
K19	4909	5622	14.52	K19′	4321	5307	22.82
K20	4301	4301	0.00	K20′	3502	3713	6.03

As shown in Table 5.20, for Bakırköy instances where critical nodes are only hospitals, WSD finds 4 and 5 optimal solutions for the instances with high and low cleaning efforts respectively. The average gap for the high cleaning effort is 3.9 and for the low one it is 3.87. Results of these instances are better than Kartal's in terms of average gaps but the optimal ratio is low for both data sets.

Table 5.30: Performance of Weighted Shortest Distance(WSD) heuristic on Bakırköy instances (hospitals) with higher and lower cleaning times

Instance	Optimal	WSD Heuristic		Instance	Optimal	WSD Heuristic	
#	Value	Value	Gap %	#	Value	Value	Gap %
B1	1979	2108	6.52	B1′	1979	2108	4.70
B2	1876	1923	2.51	B2′	1876	2009	2.51
B3	1884	1884	0.00	B3′	1884	2009	0.00
B4	1862	1923	3.28	B4′	1862	2009	3.28
B5	1920	1961	2.14	B5′	1920	2073	2.14
B6	1884	1892	0.42	B6′	1884	1953	0.42
B7	1907	1907	0.00	B7′	1907	2009	0.00
B8	1924	2048	6.44	B8′	1924	2109	6.44
B9	1897	1897	0.00	B9′	1897	2013	0.00
B10	1876	1923	2.51	B10′	1876	2009	2.51
B11	1875	1875	0.00	B11′	1875	1927	0.00
B12	1884	1945	3.24	B12′	1884	1945	3.24
B13	2169	2264	4.38	B13′	2169	2297	2.40
B14	2362	2373	0.47	B14′	2362	2373	0.47
B15	1970	1970	0.00	B15′	1970	1970	0.00
B16	2430	2782	14.49	B16′	2426	2794	15.17
B17	2671	2923	9.43	B17′	2655	2923	10.09
B18	2361	2613	10.67	B18′	2361	2613	10.67
B19	2672	2829	5.88	B19′	2570	2769	7.74
B20	2483	2621	5.56	B20′	2483	2621	5.56

In Tables 5.31, results of WSD heuristic on Bakırköy instances where the critical nodes are only schools are presented. The second model, MWSVT, cannot find optimal solutions for half of the instance in 2 hours. This is why there are some negative values in gaps which means that WSD heuristic finds better objective values for these instances.

Table 5.31: Performance of Weighted Shortest Distance(WSD) heuristic on Bakırköy instances (schools) with higher and lower cleaning times

Instance	Best objective	WSD Heuristic		Instance	Best objective	WSD Heuristic	
#	Value	Value	Gap %	#	Value	Value	Gap %
B1	2779	3103	11.66	B1′	2779	3103	11.66
B2	3439	3705	7.73	B2′	3136	3434	9.50
B3	2779	3103	11.66	B3′	2779	3103	11.66
B4	2847	3189	12.01	B4′	2847	3189	12.01
B5	2779	3103	11.66	B5′	2779	3103	11.66
B6	2779	3103	11.66	B6′	2779	3103	11.66
B7	3591	3834	6.77	B7′	3183	3834	20.45
B8	2779	3103	11.66	B8′	2779	3103	11.66
B9	3556	3591	0.98	B9′	3183	3834	2.168
B10	2779	3103	11.66	B10′	2779	3103	11.66
B11	3802	4271	12.34	B11′	3802	4271	12.34
B12	4376	4765	8.89	B12′	4232	4610	8.93
B13	4131	4195	1.55	B13′	4045	4195	3.71
B14	4017	4017	0.00	B14′	3898	4017	3.05
B15	4199	4202	0.07	B15′	4028	4202	4.32
B16	5125	5656	10.36	B16′	4927	4998	1.44
B17	3928	4408	12.22	B17′	3725	4321	16.00
B18	6105	5525	-9.50	B18′	5303	5463	3.02
B19	4426	4426	0.00	B19′	4452	4414	-0.85
B20	4030	4343	7.77	B20′	4194	4252	1.38

Chapter 6

Conclusion

In this study, we propose solution methodologies for *Debris Removal Problem in Response Phase* defined by Şahin [3]. We investigate the problem with two different objectives therefore we define two problems. The aim of the first problem, *Debris Removal in Response* (DRR), is to find a route which starts from a supply node and visits a predetermined set of critical nodes as soon as possible. The second problem, *Prioritized Debris Removal in Response* (PDRR), takes weights of the critical nodes into account and the objective is to minimize weighted sum of visiting times.

The critical nodes consist of areas close to schools and hospitals which are densely populated and have an urgent need of relief items. Although relief transportation is one of the most popular areas in emergency logistics, debris removal is not an operation commonly incorporated in relief transportation/distribution problem. Debris removal is mostly studied on late post-disaster phase with recycling incentives. In recovery and reconstruction phases the aim is to remove all debris and recycle properly while in the response phase blocked roads are cleaned from debris in order to reach critical areas. Since the purpose is to reach disaster affected people and deliver relief as soon as possible, time is of the essence during response phase. Therefore debris removal in the response phase requires different solution methodologies than the ones applied in the recovery phase.

For the defined problems, we develop two MIP models. The first one, Minimize Total Time (MTT), treats each node as equally important and minimizes the visiting time of the last visited node. A model with the same objective is proposed by Şahin [3] and in the computational analysis we show that our model is more efficient in terms of CPU consumption. For the second problem another MIP model, Minimize Weighted Sum of Visiting Times (MWSVT), is developed. This model calculates the visiting time of each critical node considering all traveling and cleaning times and minimizes weighted sum of them. Because of the additional variables and constraints, the second model results in larger CPU times.

The preliminary analysis on the mathematical models indicates that reaching optimal solution may take hours for data sets with higher dimensions. Waiting for a solution for hours conflicts with the essence of the problem since the goal is to reach people as soon as possible. Therefore we propose heuristic solution methodologies for both of the problems.

The first heuristic, Minratio, constructs a tour by connecting two critical nodes in each iteration. It decides which nodes to connect by a ratio and uses the shortest paths between the critical nodes. The ratio helps to benefit from the most advantageous shortest path by connecting the most distant critical node to its closest. An improvement algorithm based on 2-opt is applied to the solution obtained from the first part. The second heuristic, Weighted Shortest Distance, is weighted version of Nearest Neighbor Algorithm and same improvement algorithm is applied to the solution obtained from the constructive part.

Using two data sets belonging to the districts of İstanbul, we conduct computational study for all the solution methodologies. The analyses show that when the number of blocked arcs increases in the case of severe earthquakes, cleaning blocked arcs becomes inevitable. Together with the number of blocked arcs and cleaning times, the location of these blocked arcs has an important effect on the solution times for the mathematical models. Heuristic algorithms however, give solutions less than a second. For the first problem Minratio algorithm finds optimal solutions more than half of the instances but the gap from the optimal is higher for the second problem especially when the severity of the earthquake is

higher.

Our main contribution to the literature is the second problem and corresponding model. To the best of the author' knowledge, debris removal during response phase is an under-researched area and there is no study considering node priorities in this problem.

A possible future direction would be having more than one vehicle departing from several supply nodes. Moreover by conducting parametric analysis with different data sets, different functions related the the weights can be generated and analyzed for the heuristic solutions methodologies of the second problem.

Bibliography

[1] B. Özmen, M. Nurlu, and H. Güler, "Analysis of earthquake zones with geographical information system," Republic of Turkey Ministry of Public Works and Settlement, 1997.

[2] A. M. Caunhye, X. Nie, and S. Pokharel, "Optimization models in emergency logistics: A literature review," *Socio-Economic Planning Sciences*, vol. 46, no. 1, pp. 4–13, 2012.

[3] H. Şahin, "Debris Removal During Disaster Response Phase: a Case for Turkey," Master's thesis, Bilkent University, Ankara, 2013.

[4] N. Altay and W. G. Green III, "Or/ms research in disaster operations management," *European Journal of Operational Research*, vol. 175, no. 1, pp. 475–493, 2006.

[5] L. N. Van Wassenhove, "Humanitarian aid logistics: supply chain management in high gear," *Journal of the Operational Research Society*, vol. 57, no. 5, pp. 475–489, 2006.

[6] J.-B. Sheu, "Challenges of emergency logistics management," *Transportation research part E: logistics and transportation review*, vol. 43, no. 6, pp. 655–659, 2007.

[7] B. Balcik and B. M. Beamon, "Facility location in humanitarian relief," *International Journal of Logistics*, vol. 11, no. 2, pp. 101–121, 2008.

[8] D. Murphy, "Haiti earthquake: Small port-au-prince airport strained by aid demand." `http://www.csmonitor.com/World/Global-News/2010/0115/`

Haiti-earthquake-Small-Port-au-Prince-airport-strained-by-aid-demand, 2010. Visited February 2014.

[9] J. Holguín-Veras, "The donations sandy's victims don't need." http://articles.latimes.com/2012/nov/03/opinion/la-oe-holguin-veras-hurricane-donations-20121104, 2012. Visited February 2014.

[10] R. Samii, L. N. Van Wassenhove, K. Kumar, and I. Becerra-Fernandez, "Choreographer of disaster management: preparing for tomorrows disasters," *INSEAD, Fontainebleau,France*, 2002. No.06 /2002-5039.

[11] T. Lush, "98 % of haiti quake debris remains." http://www.chinapost.com.tw/international/americas/2010/09/14/272484/98-of.htm, 2010. Visited April 2014.

[12] FEMA, "Public assistance debris monitoring guide." http://www.fema.gov/pdf/government/grant/pa/fema_327_debris_monitoring.pdf, 2010. Visited June 2014.

[13] J. Stephenson, *Hurricane Katrina: continuing debris removal and disposal issues*. DIANE Publishing, 2008.

[14] O. Norio, T. Ye, Y. Kajitani, P. Shi, and H. Tatano, "The 2011 eastern Japan great earthquake disaster: Overview and comments," *International Journal of Disaster Risk Science*, vol. 2, no. 1, pp. 34–42, 2011.

[15] J. Xiao, H. Xie, and C. Zhang, "Investigation on building waste and reclaim in wenchuan earthquake disaster area," *Resources, Conservation and Recycling*, vol. 61, pp. 109–117, 2012.

[16] L. Luther, "Disaster debris removal after hurricane Katrina: status and associated issues," Congressional Research Service, Library of Congress, 2006.

[17] M. Bjerregaard, "Msb/undp debris management guidelines," *Disaster Waste Recovery*, 2009.

[18] FEMA, "Hurricane charley recovery by the numbers." http://www.fema.gov/news-release/2009/08/03/hurricane-charley-recovery-numbers, 2009. Visited June 2014.

[19] F. Baycan, "Emergency planning for disaster waste: A proposal based on the experience of the Marmara earthquake in Turkey," in *2004 International Conference and Student Competition on post-disaster reconstruction" Planning for reconstruction" Coventry, UK*, 2004.

[20] C.-P. Yang, "Composition of demolition wastes from chi-chi earthquake-damaged structures and the properties of their inert materials," *Canadian Geotechnical Journal*, vol. 46, no. 4, pp. 470–481, 2009.

[21] F. Baycan and M. Petersen, "Disaster waste management-c&d waste," in *Annual conference of the international solid waste association*, pp. 8–12, 2002.

[22] H. Hayashi and T. Katsumi, "Generation and management of disaster waste," *Soils and foundations*, pp. 349–358, 1996.

[23] M. Çelik, O. Ergun, and P. Keskinocak, "Post disaster debris clearance with incomplete information," *International IIE Conference. İstanbul,Turkey*, 2013.

[24] U. o. L. B. EM-DAT, CRED, "Disasters in numbers." /http://www.cred.be/sites/default/files/Disasters-in-numbers-2013.pdf, 2013. Visited June 2014.

[25] Tübitak, "Türkiye ulusal deprem araştırmaları programı." http://www.tubitak.gov.tr/tubitak_content_files/ARDEB/kamag/Turkiye_Ulusal_Deprem_Arastirmalari_Programi.pdf, 2005. Visited June 2014.

[26] C. Orloff, "A fundamental problem in vehicle routing," *Networks*, vol. 4, no. 1, pp. 35–64, 1974.

[27] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem," *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

[28] Z. Win, "On the windy postman problem on eulerian graphs," *Mathematical Programming*, vol. 44, no. 1-3, pp. 97–112, 1989.

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

[30] M. Guan, "On the windy postman problem," *Discrete Applied Mathematics*, vol. 9, no. 1, pp. 41–46, 1984.

[31] T. Kramberger and J. Žerovnik, "Priority constrained chinese postman problem," *Logistics & Sustainable Transport*, vol. 1, no. 1, 2007.

[32] G. N. Frederickson, M. Hecht, and C. Kim, "Approximation algorithms for some routing problems," *SIAM Journal on Computing*, vol. 7, no. 2, pp. 178–193, 1978.

[33] D. Ahr and G. Reinelt, "A tabu search algorithm for the min–max¡ i¿ k¡/i¿-chinese postman problem," *Computers & Operations Research*, vol. 33, no. 12, pp. 3403–3422, 2006.

[34] J. K. Lenstra and A. Kan, "On general routing problems," *Networks*, vol. 6, no. 3, pp. 273–280, 1976.

[35] J. Aráoz, E. Fernández, and C. Zoltan, "Privatized rural postman problems," *Computers & operations research*, vol. 33, no. 12, pp. 3432–3449, 2006.

[36] A. Letchford and R. Eglese, "The rural postman problem with deadline classes," *European Journal of Operational Research*, vol. 105, no. 3, pp. 390–400, 1998.

[37] E. Benavent, A. Carrotta, A. Corberán, J. M. Sanchis, and D. Vigo, "Lower bounds and heuristics for the windy rural postman problem," *European journal of operational research*, vol. 176, no. 2, pp. 855–869, 2007.

[38] E. Benavent, A. Corberán, I. Plana, and J. M. Sanchis, "Min-max k-vehicles windy rural postman problem," *Networks*, vol. 54, no. 4, pp. 216–226, 2009.

[39] C. H. Papadimitriou and M. Yannakakis, "Shortest paths without a map," in *Automata, Languages and Programming*, pp. 610–620, Springer, 1989.

[40] A. Bar-Noy and B. Schieber, "The canadian traveller problem," in *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pp. 261–270, Society for Industrial and Applied Mathematics, 1991.

[41] Z. Bnaya, A. Felner, and S. E. Shimony, "Canadian traveler problem with remote sensing.," in *IJCAI*, pp. 437–442, 2009.

[42] D. Fried, S. E. Shimony, and A. Felner, "Optimal policies for special cases of the canadian traveler problem,"

[43] Z. Bnaya, A. Felner, D. Fried, O. Maksin, and S. E. Shimony, "Repeated-task canadian traveler problem," in *Fourth Annual Symposium on Combinatorial Search*, 2011.

[44] K. Viswanath and S. Peeta, "Multicommodity maximal covering network design problem for planning critical routes for earthquake response," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1857, no. 1, pp. 1–10, 2003.

[45] G.-H. Tzeng, H.-J. Cheng, and T. D. Huang, "Multi-objective optimal planning for designing relief delivery systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 6, pp. 673–686, 2007.

[46] S. Yan and Y.-L. Shih, "Optimal scheduling of emergency roadway repair and subsequent relief distribution," *Computers & Operations Research*, vol. 36, no. 6, pp. 2049–2065, 2009.

[47] L. E. de la Torre, I. S. Dolinskaya, and K. R. Smilowitz, "Disaster relief routing: Integrating research and practice," *Socio-economic planning sciences*, vol. 46, no. 1, pp. 88–97, 2012.

[48] Z. Shen, M. M. Dessouky, and F. Ordóñez, "A two-stage vehicle routing model for large-scale bioterrorism emergencies," *Networks*, vol. 54, no. 4, pp. 255–269, 2009.

[49] H. O. Mete and Z. B. Zabinsky, "Stochastic optimization of medical supply location and distribution in disaster management," *International Journal of Production Economics*, vol. 126, no. 1, pp. 76–84, 2010.

[50] C. G. Rawls and M. A. Turnquist, "Mark a. turnquist. pre-position of emergency supplies for disaster response," *Transportation Research Part E*, 2009.

[51] P. Van Hentenryck, R. Bent, and C. Coffrin, "Strategic planning for disaster recovery with stochastic last mile distribution," in *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems*, pp. 318–333, Springer, 2010.

[52] G. Fetter and T. Rakes, "Incorporating recycling into post-disaster debris disposal," *Socio-Economic Planning Sciences*, vol. 46, no. 1, pp. 14–22, 2012.

[53] Z.-H. Hu and J.-B. Sheu, "Post-disaster debris reverse logistics management under psychological cost minimization," *Transportation Research Part B: Methodological*, vol. 55, pp. 118–141, 2013.

[54] A. Pramudita, E. Taniguchi, and A. G. Qureshi, "Location and routing problems of debris collection operation after disasters with realistic case study," *Procedia-Social and Behavioral Sciences*, vol. 125, pp. 445–458, 2014.

[55] J. Holguín-Veras, M. Jaller, L. N. Van Wassenhove, N. Pérez, and T. Wachtendorf, "On the unique features of post-disaster humanitarian logistics," *Journal of Operations Management*, vol. 30, no. 7, pp. 494–506, 2012.

[56] J. Holguín-Veras, N. Pérez, M. Jaller, L. N. Van Wassenhove, and F. Aros-Vera, "On the appropriate objective function for post-disaster humanitarian logistics models," *Journal of Operations Management*, vol. 31, no. 5, pp. 262–280, 2013.

[57] G. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.

[58] F. Kılcı, "A decision support system for shelter site selection with gis integration: Case for Turkey," Master's thesis, Bilkent University, Ankara, 2012.