

Minimizing latency in post-disaster debris removal

by

Meraj Ajam

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

Doctor of Philosophy

in

Industrial Engineering and Operations Management



July 2019

Minimizing latency in post-disaster debris removal

Koç University

Graduate School of Sciences and Engineering

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

Meraj Ajam

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

Committee Members:

Prof. F.Sibel Salman

Prof. Ceyda Oğuz

Assistant Prof. Bahriye Cesaret

Associate Prof. Melih Çelik

Assistant Prof. Vahid Akbari

Date: _____



To my family

ABSTRACT

After a natural disaster, roads and bridges can be damaged or blocked by debris, causing inaccessibility between critical locations such as hospitals, disaster response centers, shelters and disaster-struck areas. To facilitate immediate response operations, it is essential to clear debris from the roads in the shortest time possible. Moreover, in the longer term, management of waste and restoration of damaged roads are important activities for recovery of the region. In this study, we focus on the immediate response phase, such as the first 72 hours after the disaster and aim to optimize debris clearance decisions.

First, we study the post-disaster road clearing problem with the aim of providing a fast and effective method to determine the route of a work troop responsible for clearing blocked roads. The problem is to find a route for the troop that starts at the depot and visits all of the critical locations. The objective is to minimize the total latency of critical nodes, where the latency of a node is defined as the travel time from the depot to that node. A mathematical model for this problem has already been developed in the literature. However, for real-life instances with more than seven critical nodes, this exact formulation cannot solve the problem optimally in a 3-hour limit. To find a near-optimal solution in a short running time, we develop a MIP model based-heuristic that solves a mixed integer program on a transformed network and a lower bounding method to evaluate the optimality gaps. Alternatively, we develop a metaheuristic based on a combination of Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neighborhood Search (VNS). We test both the MIP model based-heuristic and the metaheuristic on Istanbul data and show that optimal or near-optimal solutions are obtained within seconds. We also compare our algorithms with existing work in the literature. Finally, we conduct an analysis to

observe the trade-off between total and maximum latency.

Second, we incorporate multiple work troops, multiple depots and visiting a subset of nodes (instead of visiting all nodes) in our problem. To the best of our knowledge, this problem with this generalization (and with the objective function of minimum latency) has not been investigated in the literature. We introduce a novel multi-level network model. We test the mathematical model with two categories of data sets: 1) Kartal data which is based on a real data set, located in Istanbul, Turkey. 2) random data sets in which all parameters are considered randomly to verify the strength of the proposed mathematical model. Our findings testify that the mathematical model is able to solve all instances optimally within reasonable computational times (less than 15 minutes in average).

Since these algorithms should be executed in the post-disaster stage, having short computational times is crucial and enables the problem to be resolved in real-time as new information is obtained over the response stage. Note that given the small and reasonable computational times in all proposed algorithms, any organization associated with disaster management can employ these methodologies to mitigate the negative impacts caused by a disaster.

ÖZETÇE

Doğal bir afetten sonra, yol ve köprüler zarar görebilir veya enkazdan dolayı tıkanabilirler, bu da hastaneler, afet müdahale merkezleri, sığınaklar ve afet bölgeleri gibi kritik yerler arasında ulaşımın kesilmesine neden olur. Acil müdahale işlemlerini kolaylaştırmak için, yollardan enkazları mümkün olan en kısa sürede temizlemek önemlidir. Ayrıca, uzun vadede, atık yönetimi ve hasarlı yolların restorasyonu bölgenin geri kazanılması adına önemli faaliyetlerdir. Bu tezde, afetten sonraki ilk 72 saat gibi acil müdahale aşamasına odaklanıyoruz ve enkaz temizleme kararlarını optimize etmeyi amaçlıyoruz.

Öncelikle, afet sonrası yol temizleme problemini, engellenen yolların temizlenmesinden sorumlu kişi ya da kurumların rotasını belirlemek için hızlı ve etkili bir yöntem sağlamak amacıyla inceliyoruz. Sorun, depoda başlayan ve tüm kritik yerleri ziyaret eden birlikler için bir rota bulmak. Amaç, bir düğümün gecikmesinin depodan o düğüme giden seyahat süresi olarak tanımlandığı kritik düğümlerin toplam gecikmesini en aza indirmektir. Literatürde bu problem için matematiksel bir model geliştirilmiştir. Bununla birlikte, yediden fazla kritik düğüme sahip gerçek hayattaki durumlar için, bu kesin formülasyon sorunu 3 saatlik bir limitte en iyi şekilde çözemez. Kısa bir çalışma süresinde optimale yakın bir çözüm bulmak için, dönüştürülmüş bir ağdaki karma bir tamsayı programını çözen ve optimalite farkını yorumlamak için daha düşük sınırlayıcı bir yöntem olan bir sezgisel yöntem geliştiriyoruz. Alternatif olarak, Greedy Randomized Adaptive Search Procedure (GRASP) ve Variable Neighborhood Search (VNS) kombinasyonuna dayanan bir metasezgisel algoritma geliştirdik. Hem sezgisel hem de metasezgisel algoritmaları İstanbul verileri üzerinde test ediyor ve en iyi veya en iyiye yakın çözümlerin saniyeler içinde elde edildiğini gösteriyoruz. Algoritmalarımızı literatürdeki mevcut çalışmalarla da karşılaştırıyoruz.

Son olarak, toplam gecikme ile maksimum gecikme arasındaki dengeyi gözlemlemek adına bir analiz yapıyoruz.

İkinci olarak, sorunumuza birden fazla iş birliği, birden fazla depo ve bir düğüm alt grubunu ziyaret etmeyi (tüm düğümleri ziyaret etmek yerine) dahil ediyoruz. Bildiğimiz kadarıyla, bu sorun (ve asgari gecikmenin amaç fonksiyonu olmasıyla) bu genellemeyle literatürde incelenmemiştir. Yeni bir çok seviyeli ağ modelini tanıtıyoruz. Matematiksel modeli iki veri seti kategorisiyle test ediyoruz: 1) İstanbul, Türkiye’de bulunan gerçek bir veri setine dayanan Kartal verileri. 2) önerilen matematiksel modelin gücünü doğrulamak için tüm parametrelerin rastgele olarak değerlendirildiği rastgele veri setleri. Bulgularımız matematiksel modelin tüm örnekleri makul hesaplama zamanlarında (ortalama 15 dakikadan az) en iyi şekilde çözebildiğini kanıtlamaktadır.

Bu algoritmalar afet sonrası aşamada çalıştırılması gerektiğinden, hesaplama zamanlarının kısa olması çok önemlidir ve bu müdahale aşamasında yeni bilgiler elde edilirken sorunun gerçek zamanlı olarak çözülmesini sağlar. Ayrıca, önerilen tüm algoritmalarındaki küçük ve makul hesaplama zamanları göz önüne alındığında, afet yönetimi ile ilgili herhangi bir kuruluşun, bir felaketten kaynaklanan olumsuz etkileri azaltmak için bu metodolojileri kullanabileceği unutulmamalıdır.

ACKNOWLEDGMENTS

Foremost, I would like to express my gratitude to my adviser Prof. F. Sibel Salman for her patience, motivation, enthusiasm, immense knowledge and most importantly her endless support especially in tough times. I could not have imagined having a better adviser and mentor for my PhD study. She is truly a dedicated professor. My sincere thanks must also go to the members of my thesis advisory committee: Prof. Ceyda Oğuz and Dr. Bahriye Cesaret. They generously gave their time to offer me valuable comments in the entire process of improving my work. I would also like to thank my other thesis committee members: Dr. Melih Çelik and Dr. Vahid Akbari for their time and support by reading my thesis, listening to my work, and giving great feedback on it.

I would especially like to thank my family for their continued support and encouragement. Without such a team behind me, I doubt that I would be in this place today. I am indebted to my many student colleagues for providing a stimulating and fun environment in which to learn and grow. I am sincerely thankful to Vahid Akbari and Mahdi Mostajabdaveh ,my closest friends, for encouraging and supporting each other during the process of our undergraduate and graduate studies. I am especially grateful to Sina Khoshsima, Davood Shiri, Milad Mokhtardoost, Nadia Tahmasbi, Katy Soleimanpour, Gelareh Shabab, Maziar Kasaei, Abdullah Coşkun, Baturhan Bayraktar and all my other friends at Koç University. In the end, I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting this thesis by the grant 114M373.

TABLE OF CONTENTS

List of Tables	xi
List of Figures	xiii
Nomenclature	xv
Chapter 1: Introduction	1
1.1 Our contributions	4
1.1.1 Contributions in the first problem (single work troop)	4
1.1.2 Contributions in the second problem (multiple work troops)	5
Chapter 2: Literature review	7
2.1 Road clearance	7
2.2 Minimizing total latency	11
2.3 Comparison to our work	14
Chapter 3: Problem definition and methodologies	15
3.1 Problem definition associated with ML-RCP	15
3.1.1 A lower bound for ML-RCP	17
3.1.2 Solution methods	23
3.2 Problem definition associated with MML-RCP	32
3.2.1 Mathematical model	34
Chapter 4: Computational results associated with ML-RCP	49
4.1 Data sets	49
4.1.1 Kartal data	49

4.1.2	Istanbul data	50
4.2	Kartal data results	54
4.3	Istanbul network results	56
4.4	Comparison with the related study in the literature	58
4.5	Sensitivity analysis	62
4.6	Total and Maximum Latency Tradeoffs	64
Chapter 5:	Computational results associated with MML-RCP	67
5.1	Data sets	68
5.1.1	Random data	68
5.2	Kartal data results	68
5.3	Kartal data results with multiple depots	70
5.4	A graphical analysis for a Kartal instance	73
5.5	Random data results	79
5.6	Tradeoff between total and maximum latency in MML-RCP	83
Chapter 6:	Conclusion and remarks	86
	Appendices	95
Appendix A:	Existing formulation associated with MLP	96

LIST OF TABLES

3.1	Decision variables used in the model associated with synchronization	42
4.1	Subsets of critical nodes in Kartal network	50
4.2	Kartal instances and the corresponding SOE and number of blocked-edge settings	51
4.3	Results of Kartal data sets for the mathematical model, CMLB and the MIP model based-heuristic	56
4.4	Results of Kartal data sets for CMLB, MIP model based-heuristic and metaheuristic	57
4.5	Results for simplified Istanbul data for 15 and 20 critical nodes	59
4.6	Results for simplified Istanbul data for 25 and 30 critical nodes	59
4.7	Results for southwestern Istanbul data for 15 and 20 critical nodes . .	59
4.8	Results for southwestern Istanbul data for 25 and 30 critical nodes . .	60
4.9	Heuristic results of Berktas et al. (2016) for Kartal data sets	61
4.10	Results of Simplified Istanbul data for 25 critical nodes	62
4.11	Best objective value found by metaheuristic in Simplified Istanbul data for 25 critical nodes	63
4.12	Best objective value found in Simplified Istanbul data for 25 critical nodes	63
5.1	Results of Kartal data sets for 7, 11 and 15 critical nodes with a total number of 45 nodes	72
5.2	Results of Kartal data sets for 15 critical nodes and 3 depots	74

5.3	Results of random data sets for 7 critical nodes with a total number of nodes of 20, 30, 40 and 50	80
5.4	Results of random data sets for 11 critical nodes with total number of 20 and 30 nodes	81
5.5	Results of random data sets for 11 critical nodes with total number of 40 and 50 nodes	81
5.6	Results of random data sets for 15 critical nodes with total number of 30 and 40 nodes	82
5.7	Results of random data sets for 15 critical nodes with total number of 50 nodes	83

LIST OF FIGURES

1.1	The 2011 earthquake and tsunami in Japan	2
1.2	Example with different total latency for the same maximum latency value	4
3.1	Example of ML-RCP	16
3.2	CMLB example	21
3.3	An example illustrating the shaking procedure	29
3.4	Example of MML-RCP	33
3.5	Graphical representation of the multi-level network in (Angel-Bello et al. 2017)	35
3.6	A feasible example of the multi-level network in (Angel-Bello et al. 2017)	36
3.7	An example of the multi-level network	37
3.8	Demonstration of the flow formulation	44
3.9	An example of the multi-level network with multiple depots	48
4.1	Kartal Network	51
4.2	Istanbul Network	52
4.3	VNS analysis for simplified and southwestern Istanbul	60
4.4	Results of ML-RCP under maximum latency constraint	65
5.1	Graphical representation of instance k1 with three vehicles	69
5.2	Graphical analysis associated with Table 5.2	75
5.3	Route of instance k1 with 2 vehicles	76
5.4	Route of instance k1 with 3 vehicles	77
5.5	Route of instance k1 with 4 vehicles	78

5.6	Results of MML-RCP under maximum latency constraint	84
-----	---	----



NOMENCLATURE

CMLB: Consecutive Makespan Lower Bound

CPU: Central Processing Unit

DP: Dynamic Programming

GRASP: Greedy Randomized Adaptive Search Procedure

Gurobi: An optimization solver for various types of optimization problems

ICN: Intermediate Critical Node

MIP: Mixed Integer Programming

MLP: Minimum Latency Problem

ML-RCP: Minimum Total Latency in Road Clearance Problem

MML-RCP: Multi-vehicle ML-RCP

MM-MIP: Makespan Minimization MIP

NP-hard: Non-deterministic Polynomial-time hard

SP: Shortest Path

TDP: Traveling Deliveryman Problem

TRP: Traveling Repairman Problem

TSP: Traveling Salesman Problem

VNS: Variable Neighborhood Search

Chapter 1

INTRODUCTION

Severe disasters, such as earthquakes or hurricanes, destroy the infrastructure in the affected areas and cause massive amounts of debris to accumulate, including construction materials (e.g., concrete, bricks, wood), vehicles, vegetation (e.g., fallen trees) and rubble from road infrastructure. To facilitate immediate response operations, it is essential to clear debris from the roads in the shortest time possible. Moreover, in the longer term, management of waste and restoration of damaged roads are important activities for recovery of the region. In this thesis, we focus on the immediate response phase, such as the first 72 hours after the disaster and aim to optimize debris clearance decisions.

Adverse road conditions in the aftermath of a disaster prevent access between critical locations such as hospitals, relief aid sources, shelters and casualty locations. For instance, the 2011 earthquake and tsunami in Japan (Fig. 1.1) left tens of thousands of people stranded in the devastation zone (Ranghieri and Ishiwatari (2014)). Typically, to clear the roads and possibly restore the moderately damaged road segments, one or more work troops equipped with the necessary machinery are dispatched immediately after an initial damage assessment phase (Note that prior to dispatching the vehicles, information regarding the blocked roads should be obtained by drones, helicopters or satellites. Then, the routes of the vehicles can be optimized to yield effective operations.). In most cases, clearing all the affected roads in the time-constrained immediate response stage would be impossible. Thus, it is necessary to select the most beneficial clearing tasks so that critical locations will be rendered reachable within a short time frame. Here we note that since we investigate this



Figure 1.1: The 2011 earthquake and tsunami in Japan

post-disaster problem in the **immediate** response stage, we consider the reachability of only critical nodes. The sooner each critical location is reached, the better chances are to prevent loss of lives and suffering. In addition, there are several studies (i.e., Berktaş et al. (2016) and Şahin et al. (2016)) associated with immediate response stage, developing methodologies to find optimal or near-optimal routes to ensure that all critical locations are reachable through each other.

The goal of this study is to develop an effective and applicable method to decide which roads should be cleared (unblocked) and in which order they should be cleared or repaired, by constructing the route of a clearing team. We note that in this study, we use road clearance and unblocking interchangeably for the activities that enable access on a blocked edge and may as well include repairs.

We investigate this problem in two aspects. The first one includes only one work troop positioned in a depot, while the second one is studied in a more generalized way which considers multiple work troops positioned in a depot(s). In our problem

definition, the roadway is represented by a network and the critical locations constitute a subset of the nodes. Each edge has a traversal time and, in addition, a subset of the edges are known to be blocked with given estimated clearing times. If an edge is cleared, it may be traversed again at a later time (It is assumed only in the first problem). The total travel time includes the traversal time of all edges in the route and their clearing times when blocked. The route should visit all critical nodes to ensure that they are reachable. The objective of these both routing problems are to minimize the total latency of all critical nodes. The latency of a critical node is defined as the time elapsed until it is visited. That is, it is the travel time spent from the beginning of the operation, which starts at the origin (depot node), until that critical node is reached.

Alternative objectives may be considered in the response context. For instance, minimizing the maximum latency over the critical nodes, which is the same as the total time of the route. However, when we minimize the latency of only the last visited critical node, the solution may keep people unnecessarily waiting. In the simple example in Fig. 1.2, suppose the clearing team is dispatched from node zero and nodes 1, 2 and 3 are critical. All of the edges are blocked. The numbers on the edges are the traversal times plus the additional unblocking times. There are two paths (0-1-3-2 and 0-2-3-1) that connect the critical nodes where the latency of the last visited node for both paths is 15. For the first path, the latency of nodes 1, 3 and 2 are 5, 8 and 15, respectively, so that the total latency is equal to 28. Alternatively, for the second path, the latency of nodes 2, 3 and 1 are 5, 12 and 15, respectively. The total latency of this path is 32. Although the maximum latency is the same for both paths, their corresponding total latency values are different. This simple example shows why we prefer to minimize total latency instead of maximum latency. However, we also provide an analysis where an upper bound can be set on maximum latency while minimizing total latency.

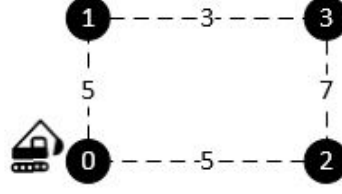


Figure 1.2: Example with different total latency for the same maximum latency value

1.1 Our contributions

1.1.1 Contributions in the first problem (single work troop)

The first problem (single work troop) was previously studied by Berktaş et al. (2016), as we explain in the literature review section. Our main contributions are the heuristics that we develop as well as the lower bounding method. We show that our heuristics largely outperform the heuristic method proposed in Berktaş et al. (2016) in terms of computational times and are able to solve larger instances. For large data sets, especially as the number of critical nodes increases, the exact formulation presented by Berktaş et al. (2016) yields loose upper and lower bounds. Thus, we suggest a novel method to generate strong lower bounds for cases with a high number of critical nodes, which is based on solving a number of mixed integer programming (MIP) models. In order to find high quality solutions for the problem, we propose a MIP model based-heuristic that solves an MIP on a transformed network that is reduced in size and hence runs in a short time. We note that the MIPs used in the lower bounding scheme and the one in the MIP model based-heuristic are different. Considering that the practitioners may not have access to a commercial solver, we propose an alternative approach, which is a metaheuristic. The metaheuristic is a hybrid algorithm that runs the constructive step of a Greedy Randomized Adaptive Search Procedure (GRASP), followed by Variable Neighborhood Search (VNS) improvement steps repeatedly. Both of our heuristic approaches run in short times in realistic-sized data sets, justifying their applicability in the post-disaster response stage. To summarize, both the MIP model based-heuristic and the metaheuristic have significantly smaller

computational times in comparison to the MIP model based-heuristic proposed by Berktaş et al. (2016) that solves the same problem. In addition, our optimality gaps are better than theirs when the number of critical nodes gets larger. We test our algorithms with larger networks and increased number of critical nodes and still maintain small execution times. Moreover, our heuristic methods work for any incomplete network and thus covers more realistic cases. We report the optimality gaps of the heuristic solutions by means of a novel lower bounding scheme.

We call this problem as the Minimum Total Latency in Road Clearance Problem (ML-RCP).

1.1.2 Contributions in the second problem (multiple work troops)

In the disaster response phase, in order to provide the connectivity of the critical nodes in road networks, using multiple vehicles can be more effective in terms of reducing the people's waiting time. Using multiple vehicles in the routes gives an opportunity to rehabilitate the blocked edges sooner in comparison to the single vehicle case, hence accelerating the response phase. Therefore, it is important to solve the more challenging multi-vehicle problem. To the best of our knowledge, the multi-vehicle road clearance problem with the total latency objective has not been investigated before. We address this problem both when all the vehicles are pre-positioned in the same depot and the case where multiple depots exist and each vehicle can initiate its route from a different depot. We will further see that considering multiple depots besides the other assumptions may provide better solutions.

A key aspect of this problem is that only a subset of nodes needs to be visited by vehicles through their routes to make the critical nodes reachable among each other. Note that in the literature, the routing problem with multiple vehicles and total latency objective has been tackled only under the requirement that all nodes should be visited (Angel-Bello et al. (2017)). Furthermore, no blocked edges exist in Angel-Bello et al. (2017)'s study because the problem has not been considered in the context of a disaster. Visiting all nodes, however, is not a practical way; especially, in

the aftermath of a disaster in which time (i.e. in the first 72 hours after the disaster) plays an essential role in the rate of casualties. Thus, it is necessary one should make efforts to resolve this challenge. In this dissertation, we introduce a novel multi-level network and then based on it we develop a mathematical model. Our findings show that the mathematical model is quite fast so that we can exert it for real data in the scale of either a large region or a city. Moreover, in order to investigate the strength of the mathematical model, we test it with random networks, where the depot node(s), critical nodes, number of blocked edges and their locations differ instance by instance. Thereby, we confirm its good performance under different settings. We will show later in Section 3.2.1 that visiting a subset of nodes (rather than visiting all of them) makes the problem more complicated in terms of the definition of the variables while increasing the elapsed time to solve the model considerably.

In line with the first problem, we call this problem the multi-vehicle ML-RCP (MML-RCP).

The remainder of this dissertation is organized as follows. A review of related studies is given in Chapter 2. The problem description with the methodologies are presented in chapter 3. We present the computational results associated with ML-RCP and MML-RCP in Chapters 4 and 5, respectively. Finally, Section 6 presents the concluding remarks.

Chapter 2

LITERATURE REVIEW

We first discuss studies on post-disaster road clearance and debris removal. Then, we review routing problems that minimize latency outside the disaster context.

2.1 Road clearance

Initial studies related to road clearance focused on determining the blocked roads to be repaired rather than constructing optimal routes for the work crew. Here we refer to the work troop (responsible for clearing roads) as a “vehicle”. Duque and Sörensen (2011) studied a disaster-affected network in which some of the roads are blocked. Weights were assigned to the towns according to their importance. Under a budget constraint, the objective was to minimize the weighted total travel time of each town to its nearest regional center. Their solution approach was based on GRASP and VNS metaheuristics and specified which blocked roads should be repaired. Özdamar et al. (2014) solved a multi-vehicle debris removal problem with two objectives. The first one minimizes the completion time of the operations while the second one maximizes the accessibility of the entire network throughout the clearing operations. They developed a mathematical model and a rule-based heuristic, which provide a balance between the two objectives while deciding the clearing order of the blocked edges as well as the assignment of the restoration tasks to the vehicles. Although the restoration tasks of the damaged roads are assigned to the vehicles, the routing of the vehicles is not considered. Aksu and Özdamar (2014) suggested a dynamic path-based MIP that decides which blocked edges should be cleared within a 3-day time horizon when resources are restricted. The performance of the algorithm for this multi-vehicle problem was tested on two data sets belonging to Turkey. Liberatore

et al. (2014) developed a multi-criteria model to optimize the recovery of the damaged roads and the commodity distributions to the people affected by hurricanes in Haiti. They considered reliability, time, cost and security as their criteria to decide which roads should be cleared under time and cost constraints. Yan and Shih (2007) proposed an MIP formulation to minimize the total restoration time in the network. A heuristic that divides the network into smaller ones was suggested so that a subproblem could be solved for each small network. However, no subproblem was able to be solved within the specified time limit. Therefore, in another study, Yan and Shih (2012) developed an ant colony system-based metaheuristic to solve instances of the problem with sizes seen in practice. Yan and Shih (2009) developed a multi-objective and multiple-commodity MIP model based on a time-space network. Their model considers both emergency roadway repair and relief distribution at the same time under time window constraints. The objective is to minimize the total time until all damaged roads get repaired. The authors also proposed an efficient heuristic to solve the same problem for a large network in their case study.

Further studies addressed the routing problem while deciding on the blocked edges to be cleared. Duque et al. (2016) developed an exact Dynamic Programming model (DP) that obtains the schedule and route of a repair crew. However, the DP falls short of solving large instances. Thus, the authors developed an iterated greedy-randomized constructive procedure which is efficient for solving large instances. Shin et al. (2019) developed an MIP and an Ant Colony Optimization (ACO) algorithm to restore the network after a disaster. Their models consider the schedule and route of both the relief vehicle and the repair crew. In another study, Moreno et al. (2019) suggested an exact model based on Benders decomposition for the crew scheduling and Routing Problem. The authors took advantage of feasibility cuts, multiple optimality cuts, and specialized valid inequalities to obtain high quality solutions. Lorca et al. (2017) developed a mathematical model for managing debris collection, disposal operations and transportation, where the goal of the problem is to reduce the conflict between objectives such as cost and duration. In addition, the authors provided a user-friendly

decision support tool which helps to decrease the gap between theory and practice in the post-disaster problems.

Şahin et al. (2016) proposed an exact model and a heuristic method to minimize the time until all critical nodes get visited. This objective is the same as minimizing the maximum latency. In their computational tests, they used data from a region of Istanbul with seven critical nodes. Berktaş et al. (2016) proposed mathematical models and heuristics to minimize (i) the maximum latency as in Şahin et al. (2016) and (ii) the total latency of the critical nodes as in our current study. Although they improved the methods in Şahin et al. (2016) for the maximum latency problem in terms of computational time, they did not test these methods with larger data. We note that their models work only when the triangular inequality holds. Moreover, in their heuristic approach, every time the vehicle traverses the same blocked edge, the clearing time is added to the objective function. Thus, it leads to an over-calculation of the objective value. This fact may cause a sub-optimal solution to be found, especially when the number of blocked edges is high. In the same vein, Ajam et al. (2019) proposed a MIP model based-heuristic and a metaheuristic that largely outperform the algorithms proposed for second problem in Berktaş et al. (2016), namely the MLP with road blockages. Furthermore, the authors verified the strength of their proposed algorithms for larger instances derived from real networks by introducing a novel lower bounding approach, which is based on solving an MIP model, iteratively. The problem sizes that could be solved by the heuristics included up to 30 critical nodes.

Several studies considered the case where the road damage causes the network to be disconnected, Kasaei and Salman (2016) developed two mathematical models as well as heuristic methods to find the route of a vehicle that clears the blocked roads to achieve connectivity. The first model minimizes the total time to restore all of the disconnected components and the second one maximizes the total collected prize by connecting components in a given time limit. Vodák et al. (2018) suggested a metaheuristic method based on Ant Colony Optimization (ACO) to connect all disconnected components in minimum total time. In their work, they reduced the size

of the network by using only boundary nodes of the components. As a result, their method is able to solve networks with up to 723 nodes. Akbari and Salman (2017b) developed mathematical models and heuristic algorithms for the multi-vehicle case of the first problem in Kasaei and Salman (2016) and Akbari and Salman (2017a) studied the second one, again with multiple vehicles. In the former work, the authors proposed an approach that solves a relaxation of the problem and implemented a feasibility procedure followed by a neighborhood search algorithm. In the latter work, they developed a mathematical model as well as a MIP model based-heuristic that decomposes the problem into single vehicle problems and solves them back-to-back with updated prizes. They derived upper bounds to this maximization problem by Lagrangian relaxation of a relaxed MIP and proved some optimality verification properties. They tested their methods on both randomly generated Euclidean and Istanbul road network instances, with a maximum of 350 nodes and around 700 edges. They derived optimal or near-optimal solutions with small gaps. We note that in the multi-vehicle routing problem, the coordination of the vehicle routes is required, since no road is allowed to be traversed before its unblocking procedure is finished. In another study, Morshedlou et al. (2018) developed two mathematical models dealing with synchronized routing problem for planning and scheduling restorative efforts in the wake of a disaster. In order to increase the flexibility of their models, the authors considered the number of restoration crews designated to each disrupted component as a decision variable. Moreover, each disrupted component is allowed to be partially active during its restoration process, in the second formulation. The authors further, employed a relaxed MIP with some valid inequalities to obtain coordinated routes.

Çelik et al. (2015) defined a stochastic debris clearance problem in which road clearing times are uncertain. In this multi-period problem, the objective is to reconnect the supply and demand nodes. In each period, the clearance time as well as other information is updated. The main decision is to find a sequence of roads which are cleared in each period. The authors developed a heuristic policy. Note that recently, a survey on network restoration and recovery is presented by Çelik (2016)

from a multi-disciplinary perspective and includes related studies regarding the road clearance problem as well.

2.2 Minimizing total latency

The Minimum Latency Problem (MLP) is a single-vehicle node-routing problem in which the edges are not blocked and every node should be visited. This problem arises in applications such as repair services, where the total waiting time of the customers is of concern (rather than the traveling cost of the vehicle) and is also called the Traveling Repairman Problem by some authors. However, in our first problem, nodes may be visited more than once if desired, and edges may be traversed more than once if desired. This variant of the TSP was proposed, apparently independently, by three sets of authors Cornuéjols et al. (1985), Fleischmann (1985), Orloff (1974), which are calling as Steiner TSP. Sahni and Gonzalez (1976) proved that MLP is NP-hard in general metrics. In spite of the fact that there are some similarities with the Traveling Salesman Problem (TSP), Goemans and Kleinberg (1998) showed that the MLP is more difficult to solve computationally than the TSP.

Angel-Bello et al. (2013) developed strong mathematical models for the MLP using a multi-level network representation. These models only work for complete input networks and were tested with up to 40 nodes with an average computational time of 100 seconds for the largest instances. Sarubbi et al. (2008) solved the time-dependent multi-commodity MLP for asymmetric instances with up to 80 nodes by a branch-and-cut algorithm based on Benders Decomposition. Méndez-Díaz et al. (2008) presented a three-index MIP formulation as well as several valid inequalities associated with their MIP. The proposed model is able to solve instances with up to 40 nodes. Bulhões et al. (2018) recently developed a novel branch-and-price algorithm (BP) for a set partitioning formulation. This new algorithm is able to solve the instances left unsolved in the literature (Roberti and Mingozzi (2014)). Luo et al. (2014) proposed an exact branch-and-price-and-cut algorithm for the multi-vehicle MLP with a distance constraint, to visit all nodes only once. An ad hoc label-setting algorithm

with bi-directional search strategy is developed to solve the pricing problem. They tested 180 instances with up to 50 nodes and obtained an average gap less than 0.5%. Arellano-Arriaga et al. (2019) developed a bi-objective problem which minimizes the latency and the travel time of a single-vehicle. The authors proposed two heuristic methods for approximating the Pareto fronts, namely, a strategic memetic search and evolutionary algorithm with intelligent local search to tackle this problem.

Several studies focused on developing heuristic and metaheuristic approaches for the MLP. Salehipour et al. (2011) developed a metaheuristic method which is based on a GRASP for the construction phase and Variable Neighborhood Descent (VND) and VNS for the improvement step. They derive a lower bound by sorting the edges of a minimum spanning tree and an upper bound from the nearest neighbor heuristic. They compared the metaheuristic solutions with upper and lower bounds on data with up to 1000 nodes. The execution times of the metaheuristic turns out to be very high for instances with more than 500 nodes (more than 3 hours for 500 nodes, with an average gap of 50% and 10% compared to the lower and upper bound, respectively). Dewilde et al. (2013) addressed the MLP with profits, where the vehicle visits a subset of nodes to maximize the total revenue in a time limit. A tabu search algorithm with various neighborhood structures is presented to solve large data sets. They solved the problem with up to 500 nodes and improved the Salehipour et al. (2011)'s solutions by an average of 16% within the given computation time limit. Silva et al. (2012) introduced a metaheuristic for the MLP in which GRASP and VND with random neighborhood orderings with double-bridge perturbation mechanism are used for the construction and improvement steps, respectively. Their test instances contain up to 1000 nodes, where the average execution time for 1000-node instances is as high as 27,500 seconds. They compared the best and average solutions found by their method with the upper bounds computed using the nearest neighbor heuristic, as in Salehipour et al. (2011). The average improvement obtained can be as high as 15%. Mladenović et al. (2013) proposed a General VNS (GVNS) metaheuristic to solve Traveling Deliveryman Problem (TDP) that outperforms the one suggested by

Salehipour et al. (2011) . They took advantage of classical neighborhoods used for the TSP, and efficiently adapted them for solving the TDP.

Several investigations have been devoted to the multi-vehicle MLP. Nucamendi et al. (2015) developed a mathematical model based on a single-commodity flow formulation which had been introduced by Gavish and Graves (1978). The aim of the model is to find m disjoint paths to visit all nodes, where m is the number of vehicles. Their formulation is able to solve instances with up to 40 nodes. They also developed a metaheuristic algorithm composed of two stages. For the initial solution, they used clustering strategies (CS) which are based on the distance of the nodes to the depot node. The authors also checked the quality of the solutions once they consider randomness in the construction of the initial solutions. For the improvement phase, they implemented local search in which both intra-route and inter-route composite moves have been considered. By doing so, they could compare their metaheuristic algorithm solutions with those found by the mathematical model on instances having up to 50 nodes. However, for larger instances they only compared their metaheuristic with their initial solutions. In another study, Nucamendi-Guillén et al. (2016) presented a new mathematical model which is based on a multi-level network, which finds the optimal solution quite faster than the previous mathematical model in Nucamendi et al. (2015). To solve larger instances, they developed a metaheuristic based on an iterated greedy algorithm (IG) (which is a special case of a multi-start (MS) algorithm) for the initial solution. For the improvement phase, they employed five local search strategies which are arranged in two intra and inter-composite local search. Their results indicate that the average gap between the mathematical model and the metaheuristic is less than 1%. In order to understand the performance of their metaheuristic for larger instances, they derived strong lower bounds from the Cumulative Capacitated Vehicle Routing Problem (CCVRP) in the literature, where they relaxed the constraints associated with demand and supply nodes' capacities. The average gap between their metaheuristic algorithm and the relaxed CCVRP is less than 3%. Bang (2018) developed a metaheuristic algorithm which is based on GRASP

as the initial solution. Furthermore, the author implemented a VND algorithm by considering a distance constraint for each vehicle, which improves the solution. Angel-Bello et al. (2017) proposed five mathematical models for the multi-vehicle minimum latency problem. The first three formulations were derived from classical models and from a flow-based formulation to the multiple TSP. The last two ones are obtained from time-dependent formulations based on a multi-level network, which have a much better performance compared to the first three ones in terms of computational time. They solved instances up to 80 nodes and 16 vehicles with an average computational time of 80 seconds. Note that in all studies addressed, the authors developed their models based on a disjoint path assigned to each vehicle.

2.3 Comparison to our work

Although various studies addressed the MLP as mentioned above, to the best of our knowledge, the number of studies that focus on road clearance and minimizing total latency is very limited. In our problem in MR-LCP, blocked edges can be cleared and traversed afterwards. In the second traversal of such blocked edges, the clearing time is set to zero and consequently, shortest path times are modified. For the resulting problem, we provide a lower bounding scheme for the first time here.

In addition, all of the studies associated with multi-vehicle MLP assume networks in which all nodes must be visited, following the standard definition of MLP in the literature. With this in mind, in MML-RCP, our concern is to visit only critical nodes in order to make them reachable since visiting all nodes is not efficient in the immediate response phase. In a broader perspective, in order to obtain a more realistic solution to MML-RCP, we assume that each critical node must be visited by a vehicle and non-critical nodes may or may not be visited through the routes. Moreover, we incorporate multiple depots in our model to diminish the latency of reaching the critical locations. We keep the disjoint paths assumption similar to those in the literature.

Chapter 3

PROBLEM DEFINITION AND METHODOLOGIES

In this chapter, we describe the problem definition and the methodologies used for both ML-RCP and MML-RCP, each in a separated section.

3.1 *Problem definition associated with ML-RCP*

Consider a disaster-affected region in which some road segments are blocked. We represent the road network by an undirected graph having a subset of edges blocked and a subset of nodes identified as the “critical” nodes. The remaining elements of the graph are the origin node, the non-critical nodes that may or may not be visited and intact edges which may be traversed in the route of the road clearing team. Each edge has a traversal time and each blocked edge has an additional clearing time. The road clearing team, referred to as the “vehicle”, starts its route from the origin node, clears the blocked edges on its route and completes the route at one of the critical nodes when all critical nodes have been visited. The problem is to find a route, which is an open walk, starting at the origin node and visiting all of the critical nodes such that the total latency of the critical nodes is minimized. The latency of a critical node is the travel time from the origin node, where the vehicle is initially positioned, to that critical node via a walk, including both the traversal time of all edges and the clearing time of the blocked edges in the walk. The origin node is termed the depot throughout the article. The solution specifies the route for the clearing team, the order in which the critical nodes are visited and which blocked edges are cleared and in which order. Here we note that nodes other than the critical ones may also be visited in a feasible route. These nodes are called intermediate nodes. We refer to this problem as the Minimum Total Latency in Road Clearance Problem (ML-RCP).

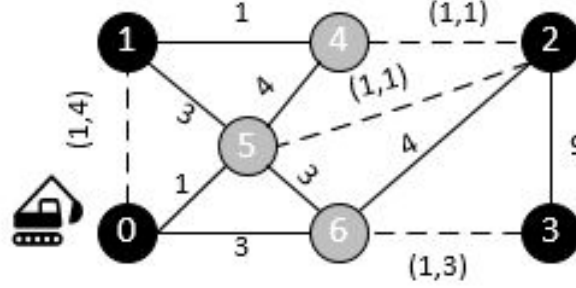


Figure 3.1: Example of ML-RCP

Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E represents the set of edges. We define bi-directional arcs $A = \{(i, j) \cup (j, i) : \{i, j\} \in E\}$ since we need directions for the traversal of the edges. All of the parameters associated with arcs (i, j) and (j, i) are assumed to be symmetric. The origin node, the critical nodes and the non-critical nodes form the node set. We denote the origin node as 0. Blocked edges are represented by $B \subseteq E$. Moreover, u_{kl} is the time required to clear edge $\{k, l\} \in B$ and t_{kl} is the time required to traverse through arc $(k, l) \in A$. Let $N \subseteq V$ be the set of critical nodes, where $N \cup \{0\} = DN$.

Fig. 3.1 shows a simple example of the ML-RCP in which nodes 1, 2 and 3 are critical, whereas nodes 4, 5 and 6 are non-critical. The blocked edges are depicted by dashed lines. The numbers on the edges are the traversing time and clearing time of the blocked edges. In a feasible solution, the vehicle departs from node 0 and finishes its walk by visiting all the critical nodes specified as 1, 2 and 3 in this example. For instance, in one feasible solution, the route includes nodes 0-1-5-2-3. Here 5 is an intermediate node in between critical nodes 1 and 2, and blocked edges $\{0, 1\}$ and $\{5, 2\}$ should be cleared in the given order. In this feasible solution, latency of node 1 is 5, node 2 is 10 and node 3 is 19, adding up to a total latency of 34. The order in which the nodes are visited in the optimal route for this example is as follows: 0-5-2-4-1-5-6-3. In this solution, the latency of critical nodes 2, 1 and 3 are 3, 6 and 16, respectively, with total latency 25.

3.1.1 A lower bound for ML-RCP

Berktaş et al. (2016) developed a mathematical model for the ML-RCP. However, according to our computational results presented in Chapter 4, the model falls short of solving instances in which the number of critical nodes increases beyond 7 (where the model has 38,016 variables and 27,118 constraints in the most complicated instances named k20) in our test bed. We tried to obtain tight lower bounds from Berktaş et al. (2016)'s formulation. Nevertheless, the best lower bound we obtained from the exact formulation, when it is solved within a computational time limit (3 hours), was too loose. Therefore, we suggest a new method to generate strong lower bounds for cases with a higher number of critical nodes. This lower bound is valid for any input graph, including incomplete graphs. The method works on a transformed graph which is complete. The method is based on solving an MIP model, formulated on a complete graph $G_C = (V_C, E_C)$, on the set of critical nodes and the depot, iteratively. In this way, the size of the network reduces significantly. In this transformed graph, the travel time between each pair of nodes is defined by finding the shortest path between them in the original graph, $G = (V, E)$. The travel time of edge $(i, j) \in E$ is taken as $t_{ij} + u_{ij}B_{ij}$, where t_{ij} and u_{ij} are the traversal and clearing times of edge $(i, j) \in E$, respectively. Here the parameter B_{ij} is equal to one, if edge $(i, j) \in E$ is blocked; and zero, otherwise. We denote this shortest path duration for edge $(i, j) \in E$ as sp_{ij} .

Each time the MIP model that we present below is solved, the transformed graph is the input graph. The objective of this MIP model is to minimize the total time until a given (critical) destination node is visited. Moreover, a fixed number of intermediate critical nodes must be visited through the route. Hence, this MIP model minimizes the makespan of visiting a set of critical nodes including the destination node. We call this MIP, which is solved consecutively with different input parameters, the Makespan Minimization MIP (MM-MIP). Contrary to the ML-RCP which minimizes the total latency, the makespan problem minimizes the latency of the last visited critical node, i.e. the destination node.

We present the MM-MIP below. This mathematical model is derived from the sin-

gle commodity flow formulation which was proposed by Gavish and Graves (1978) for Multiple Traveling Salesman Problem (mTSP) and adapted to our problem. In each implementation of the model, two input parameters are specified: a destination node D and ICN, the number of intermediate critical nodes that should be visited before D .

MM-MIP (ICN,D):

Sets:

V_C : set of nodes consisting of the depot, 0, and the critical nodes $1, 2, \dots, n$

Parameters:

sp_{ij} : travel time of edge $(i, j) \in E_C$

Decision variables:

$x_{ij} = 1$, if edge $(i, j) \in E_C$ is traversed from node i to node j ; 0, otherwise

$y_i = 1$, if node $i \in V_C \setminus \{0\}$ is visited; 0, otherwise

f_{ij} : amount of flow on edge $(i, j) \in E_C$ from node i to node j

$$\min \sum_{i \in V_C} \sum_{j \in V_C, j \neq i} sp_{ij} x_{ij} \quad (3.1)$$

s.t.

$$\sum_{i \in V_C \setminus \{0\}} x_{0,i} = 1 \quad (3.2)$$

$$\sum_{i \in V_C, i \neq D} x_{iD} = 1 \quad (3.3)$$

$$x_{Di} = 0 \quad \forall i \in V_C \setminus \{0\}, i \neq D \quad (3.4)$$

$$\sum_{i \in V_C, i \neq j} x_{ij} - \sum_{k \in V_C \setminus \{0\}, k \neq j} x_{jk} = 0 \quad \forall j \in V_C \setminus \{0\}, j \neq D \quad (3.5)$$

$$x_{ij} + x_{ji} \leq 1 \quad \forall (i, j) \in E_C \quad (3.6)$$

$$\sum_{j \in V_C, j \neq i} x_{ji} = y_i \quad \forall i \in V_C \setminus \{0\} \quad (3.7)$$

$$\sum_{i \in V_C \setminus \{0\}} f_{0,i} = ICN + 1 \quad (3.8)$$

$$\sum_{i \in V_C, i \neq D} f_{iD} = 1 \quad (3.9)$$

$$\sum_{i \in V_C, i \neq j} f_{ij} - \sum_{k \in V_C \setminus \{0\}, k \neq j} f_{jk} = y_j \quad \forall j \in V_C \setminus \{0\}, j \neq D \quad (3.10)$$

$$f_{ij} \leq (ICN + 1)x_{ij} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \quad (3.11)$$

$$f_{ij} \geq x_{ij} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \quad (3.12)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \quad (3.13)$$

$$y_i \in \{0, 1\} \quad \forall i \in V_C \setminus \{0\} \quad (3.14)$$

$$f_{ij} \geq 0 \quad \forall i \in V_C, \forall j \in V_C, i \neq j \quad (3.15)$$

Objective function (3.1) is to minimize the total time of the path from 0 to D. Constraint (3.2) guarantees that the vehicle leaves the depot only once by visiting one of the critical nodes. Constraint (3.3) ensures that the destination node must be visited once at the end of the route. By Constraints (3.4) once the destination node is visited, the route ends. Constraints (3.5) are balance equations for intermediate critical nodes. Since the triangular inequality holds in the transformed graph, the vehicle traverses an edge at most once in one direction, which is stipulated by Constraint (3.6). Constraints (3.7) define if node i is visited or not. Constraints (3.8-3.10) are flow balance equations, where a unit flow is left at each intermediate critical node. In Constraint (3.8), the net flow out of the depot node should be equal to the total number of visits to all other nodes. Constraint (3.9) forces the model to send a unit flow to the destination node. Constraints (3.10) ensure that for the intermediate critical nodes, the net flow equals to one if the corresponding node is visited. In fact, the vehicle loses one unit of flow each time it visits a node. Constraints (3.11) prevent any positive flow on edge $\{i, j\}$, if it is not traversed. In addition, the maximum flow cannot be more than the number of intermediate critical nodes and the destination node. Constraints (3.12) ensure that if edge $\{i, j\}$ is used, a positive flow should be assigned to that edge. Constraints (3.13-3.15) define the domains of the variables.

Next, we describe our lower bounding approach. We provide a pseudo-code in Algorithm 1. Lines 1-2 explain how the transformed graph is formed. As specified in line 5, MM-MIP has two inputs for each implementation, which are D and ICN . Given the depot node 0 and the destination node D , exactly ICN intermediate critical nodes can exist in the route provided by MM-MIP in each run. The number of intermediate critical nodes, ICN , runs from zero to the number of critical nodes minus one. Recall that MM-MIP minimizes the latency of the last visited node, i.e. the destination node D , with a specific number of intermediate critical nodes that must be visited through the route. As a result, MM-MIP should be solved $(|V_C| - 1)^2$ times, as can be seen from lines 3-4. We need to rectify the value of the objective function found by MM-MIP in each run. It is possible that a blocked edge is traversed more than once through the route and as a result, the clearing time would be added more than once in the objective function. To obtain the correct objective function value of the corresponding solution, we transform this solution to the original graph such that the visiting order of the critical nodes is kept. We find the shortest path between each pair of critical nodes in the specified order. If a blocked edge is cleared in the shortest path between two critical nodes, we update this edge for the next shortest path problem and assume it is unblocked for the next uses. By doing so, if the vehicle traverses a blocked edge more than once, the clearing time of this edge is accounted for only one time in the objective function of the original problem. We call these steps “objective correction procedure”.

After the objective correction procedure, we obtain the value $Z_M(ICN, D)$ which corresponds to the minimum time that node D can be visited while exactly ICN different intermediate critical nodes are visited beforehand. After all model runs are completed, in lines 9-11, we keep all of η_{ICN} ($ICN = 0, 1, \dots, |V_C| - 2$). For instance, η_m shows the *minimum* latency value obtained by MM-MIP from all destination nodes in which m intermediate critical nodes are visited. Therefore, $\sum_{ICN=0}^{|V_C|-2} \eta_{ICN}$ is the lower bound that we name the Consecutive Makespan Lower Bound (CMLB).

We next describe an example in order to explain the lower bounding. Fig. 3.2 (a)

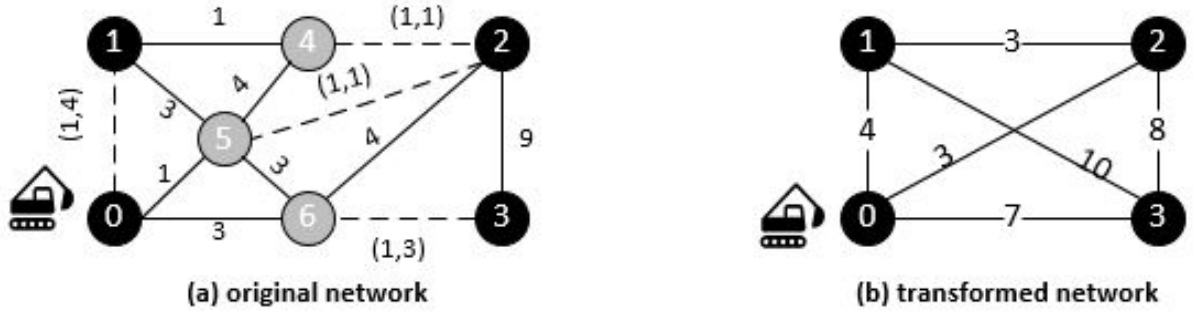


Figure 3.2: CMLB example

shows the same example that has been represented in Section 3.1. The transformed graph is represented in Fig. 3.2 (b) and the number on each edge $\{i, j\}$ is the shortest path time from critical node $i \in V_C$ to critical node $j \in V_C$ (sp_{ij}). On the newly formed graph, we start with one of the critical nodes as a destination node and we select node 1. MM-MIP finds the best route from the depot node to destination node 1, given the number of intermediate critical nodes (ICN) that must be visited, which are zero, one and two. Thus, MM-MIP should be solved three times, with zero, one and two intermediate critical nodes. Let L_i be the latency of critical node $i \in V_C$ of ML-RCP and $Z_M(ICN, D)$ be the optimal corrected objective function found by MM-MIP. In Fig. 3.2, the optimal total latency until all critical nodes are visited is equal to 25 ($3+6+16$) with the visiting order of depot, 2, 1 and 3. For destination node 1, the values of $Z_M(0, 1)$, $Z_M(1, 1)$ and $Z_M(2, 1)$ are 4, 6 and 18, respectively.

Afterwards, the next critical node should be chosen as the new destination node and we continue the same procedure for the rest of the critical nodes. So, for destinations 2 and 3, $Z_M(ICN, 2) = \{Z_M(0, 2) = 3, Z_M(1, 2) = 7, Z_M(2, 2) = 20\}$ and $Z_M(ICN, 3) = \{Z_M(0, 3) = 7, Z_M(1, 3) = 11, Z_M(2, 3) = 15\}$, respectively. Note that according to line 6 in Algorithm 1, the objective correction procedure has been performed after each run of MM-MIP in case a blocked edge is traversed more than once from the depot node to the destination node D. Suppose the visiting order of critical nodes of a route obtained from one run of MM-MIP is depot, 3, 2 and desti-

nation node 1. First, we begin by finding the shortest path between the depot and node 3. Suppose edge $(i, j) \in B$ is unblocked in this shortest path. We set $u_{ij} = 0$ meaning that it is not necessary to spend extra time for traversing $\{i, j\}$ for a second time. The next step is to find the shortest path between 3 and 2. The same procedure applies to the shortest path between 3 and 2. The procedure continues as described, until all of the critical nodes in the route are visited.

When all corrected objective values ($Z_M(ICN, D)$) are found from MM-MIP, we take the minimum latency value over all destination nodes, which is $\eta_{ICN} = \min_{D \in V_C \setminus \{0\}} Z_M(ICN, D)$. Note that by this definition, η_i shows the minimum possible time that one of the critical nodes is visited exactly after i different critical nodes are visited ($i = 0, 1, \dots, |V_C| - 2$). First, we find the solutions with no intermediate critical nodes. We choose the minimum value as the latency of the first visited critical node ($\eta_0 = \min\{4, 3, 7\} = 3$). Then, we select the solutions with one intermediate critical node in the route and take the minimum value as the latency of the second visited critical node ($\eta_1 = \min\{6, 7, 11\} = 6$). We perform the same procedure for the solutions with two intermediate critical nodes as well and take them into account as the latency of the third visited critical node ($\eta_2 = \min\{18, 20, 15\} = 15$). Finally, we sum these latency values to obtain the total latency provided by the proposed lower bound, which is $\sum_{ICN=0}^2 \eta_{ICN} = 3 + 6 + 15 = 24$ (In this example, the gap between the lower bound and the optimal value is $\frac{1}{25} = 4\%$). Note that while we calculate η_i , it is possible that a certain node might correspond to multiple η_i values (for $i = 0, 1, \dots, |V_C| - 2$), although we did not encounter such a case in this example.

Proposition 1. *CMLB is a lower bound for the total latency of the ML-RCP.*

Proof. Suppose the permutation of the critical nodes in an optimal solution of the ML-RCP is $P = 0[1][2] \dots [n]$ where 0 is the depot node and $[i]$ is the index of the critical node visited in position i . Let $L_{[i]}$ be the latency of the critical node in position i in this particular solution. Recall that in the proposed lower bound, we aim to bound the latency value of each position i separately (instead of minimizing total latency in ML-RCP). The key idea of the lower bounding scheme is as follows.

Since in line 10 of Algorithm 1, $\eta_{[i]} = \min_{D \in V_C \setminus \{0\}} Z_M(i, D)$, it is less than or equal to $L_{[i]}$. In fact, $\eta_{[i]}$ takes the minimum latency value in position i considering all critical nodes and $L_{[i]}$ is one of such values and hence it is not necessarily the minimum latency value. Since $\eta_{[i]} \leq L_{[i]}$ for all $i = 0, 1, 2, \dots, |V_C| - 2$, we can conclude that $\sum_{i=0}^{|V_C|-2} \eta_{[i]} \leq \sum_{i=0}^{|V_C|-2} L_{[i]}$, meaning that CMLB is a lower bound to the objective value of ML-RCP. $\square$

Algorithm 1 Steps of the lower bounding method for ML-RCP

Input:

- The original graph G and data associated with G
- 1: Create a complete graph, $G_C = (V_C, E_C)$ on the set of critical nodes and the depot node.
 - 2: Find the shortest path time sp_{ij} , for all $(i, j) \in E_C$.
 - 3: **for** D in $V_C \setminus \{0\}$ **do**
 - 4: **for** $ICN = 0, 1, \dots, |V_C| - 2$ **do**
 - 5: Solve MM-MIP(ICN, D)
 - 6: Implement the objective correction procedure.
 - 7: **end for**
 - 8: **end for**
 - 9: **for** $ICN = 0, 1, \dots, |V_C| - 2$ **do**
 - 10: $\eta_{ICN} = \min_{D \in V_C \setminus \{0\}} Z_M(ICN, D)$
 - 11: **end for**
 - 12: Calculate the total latency value ($\sum_{ICN=0}^{|V_C|-2} \eta_{ICN}$) and assign it to CMLB.
-

3.1.2 Solution methods

This section provides two solution methods for the ML-RCP in order to find high quality solutions within a short time. First, we present a MIP model based-heuristic that solves an MIP on a transformed graph. Second, we develop a metaheuristic which does not need a commercial solver.

3.1.2.1 MIP model based-heuristic approach

In this section, we suggest a MIP model based-heuristic approach to solve instances of ML-RCP with high number of critical nodes. Similar to Section 3.1.1, the input graph

can be either complete or incomplete. However, the method works on a transformed complete graph. The method is based on solving an MIP model formulated on a complete graph on the set of critical nodes and the depot, and hence reduces the size of the problem significantly. We work on the transformed graph, $G_C = (V_C, E_C)$, defined exactly as in Section 3.1.1.

On the newly formed graph, we find the visiting order of the critical nodes using the MIP model of MLP proposed in Angel-Bello et al. (2013). This model is formulated by a multi-level network whose graphical representation is similar to the Picard and Queyranne representation (Picard and Queyranne (1978)) for time-dependent TSP. We present the MIP model of Angel-Bello et al. (2013) for the sake of completeness below. Note that in the MLP, no blocked edges exist and every node should be visited exactly once.

Sets:

V_C : set of nodes consisting of the depot, 0, and the critical nodes $1, 2, \dots, n$.

K : set of *positions* = $\{1, 2, \dots, n\}$

Parameters:

sp_{ij} : travel time of edge $(i, j) \in E_C$

Decision variables:

$x_{ik} = 1$, if node i is in position k in a permutation; 0, otherwise; $\forall i \in V_C \setminus \{0\}, k \in K$

$y_{ijk} = 1$, if node i is in position k and node j is in position $k + 1$ in a permutation; 0, otherwise;

$$\forall i, j \in V_C \setminus \{0\}, j \neq i, k \in K \setminus \{n\}$$

$$\min z = n \sum_{i \in V_C \setminus \{0\}} sp_{0,i} x_{i,1} + \sum_{k \in K \setminus \{n\}} \sum_{i \in V_C \setminus \{0\}} \sum_{j \in V_C \setminus \{0\}, j \neq i} (n - k) sp_{ij} y_{ijk} \quad (3.16)$$

s.t.

$$\sum_{k \in K} x_{ik} = 1 \quad \forall i \in V_C \setminus \{0\} \quad (3.17)$$

$$\sum_{i \in V_C \setminus \{0\}} x_{ik} = 1 \quad \forall k \in K \quad (3.18)$$

$$\sum_{j \in V_C \setminus \{0\}, j \neq i} y_{ijk} = x_{ik} \quad \forall i \in V_C \setminus \{0\}, k \in K \setminus \{n\} \quad (3.19)$$

$$\sum_{j \in V_C \setminus \{0\}, j \neq i} y_{ijk} = x_{i,k+1} \quad \forall i \in V_C \setminus \{0\}, k \in K \setminus \{n\} \quad (3.20)$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in V_C \setminus \{0\}, k \in K \quad (3.21)$$

$$y_{ijk} \geq 0 \quad \forall i \in V_C \setminus \{0\}, j \in V_C \setminus \{0\}, j \neq i, k \in K \setminus \{n\} \quad (3.22)$$

The objective is to minimize the total latency. Constraints (3.17) guarantee that each critical node must occupy a single position. Constraints (3.18) force that each position is allowed to be captured by a single critical node. By Constraints (3.19), only one arc can leave from position k . Constraints (3.20) ensure that only one arc can arrive at position $k+1$. Finally, Constraints (3.21) and (3.22) define the variables with their domains.

By solving this MIP formulation, the best visiting order of the critical nodes is found for the new complete graph with shortest path times on the edges. However, it is possible that the solution may not be optimal to the original problem, ML-RCP, if a blocked edge is traversed more than once since its clearing time is added in each traversal. Therefore, we implement the objective correction procedure as we have already described in Section 3.1.1.

We use the same example shown in Fig. 3.2 to illustrate the MIP model based-heuristic. The best visiting order of the critical nodes according to the optimal solution of the formulation in Angel-Bello et al. (2013) is depot, 2, 1 and 3. First, we begin by finding the shortest path between the depot and node 2. Suppose edge $(i, j) \in B$ is unblocked in this shortest path. We correct the objective value by setting $u_{ij} = 0$, meaning that it is not necessary to spend time for unblocking $\{i, j\}$ if it is traversed for the second time. The next step is to find the shortest path between 2 and 1. The same procedure applies to the shortest path between 2 and 1. The procedure continues as described, until all of the critical nodes are visited. The objective value

found by the MIP model based-heuristic approach is 25, which is equal to the optimal solution of ML-RCP.

The steps of the MIP model based-heuristic algorithm are given in Algorithm 2.

Algorithm 2 Steps of the MIP model based-heuristic approach for ML-RCP

Input:

The original graph G and data associated with G .

- 1: Create a complete graph, $G_C = (V_C, E_C)$ on the set of critical nodes and the depot node.
 - 2: Find the shortest path time sp_{ij} , for all $(i, j) \in E_C$
 - 3: Solve Angel-Bello et al. (2013) MIP formulation to find the best visiting order of the nodes.
 - 4: Implement the objective correction procedure.
 - 5: **return** the route and total latency.
-

It is worth mentioning that if no commercial solver is available, any efficient heuristic algorithm for the MLP (e.g. Silva et al. (2012) and Mladenović et al. (2013)) can be used in line 3 of Algorithm 2 in place of the exact formulation by Angel-Bello et al. (2013). This would lead to an alternative metaheuristic to the one presented in the next section.

3.1.2.2 Metaheuristic algorithm

In this section, we propose an alternative approach, namely a metaheuristic to solve ML-RCP. This approach has the advantage that it does not need a commercial solver. Thus, it may be preferred by the planners. Our metaheuristic is based on the constructive phase of the GRASP, followed by a VNS for an improvement step. Both of these steps are repeated a number of times until a specified time limit is reached and the best solution found is kept. The outline of the metaheuristic approach is shown in Algorithm 3.

3.1.2.2.1 GRASP

Algorithm 3 Metaheuristic approach for ML-RCP

-
- 1: **while** the time limit is not over **do**
 - 2: Find an initial solution using GRASP construction step, say x_G .
 - 3: Improve x_G using VNS and obtain the new x_B .
 - 4: **end while**
 - 5: **return** Best solution found, x_B .
-

GRASP is a metaheuristic approach that was first introduced by Resende (2001). The main idea of this method is adding randomness to the greedy choice in order to change the behavior of the deterministic greedy heuristic and to explore more diverse solutions.

When we implement the GRASP construction step to our problem, we start with the input graph being the transformed graph as described in Section 3.1.1. Note again that the transformed graph is a complete graph, $G_C = (V_C, E_C)$, consisting of only the depot and the critical nodes. The travel time between each pair of critical nodes is equal to the shortest path distance (sp_{ij}) between them in the original graph.

We implement the construction step of GRASP procedure described in Algorithm 4. We define a partial sequence called S in which only the depot node exists at the beginning of the algorithm (line 1). We create a Candidate List (CL) such that all of the nodes except the depot node exist in this list (line 2). We pick node r in order to calculate the distances between node r and each node remaining in CL . Initially r is set to the depot node. Then, the algorithm sorts the nodes in CL in non-decreasing order according to their distances with respect to node r (line 6). In GRASP method, in order to add randomness to greediness a parameter such as $\alpha \in (0, 100)$ is defined. In fact, this parameter helps the algorithm to control the balance between greediness and randomness. Setting $\alpha = 0$ entails a strictly deterministic greedy search. On the other hand, if $\alpha = 100$, it will be a completely random search. As α increases, randomness also increases as well. Given the parameter α , we make a new Restricted Candidate List (RCL) in which first $\alpha\%$ of the candidates in the sorted CL exist (line 7). Next, the algorithm selects a random node with equal probabilities from RCL and adds it to the end of the partial sequence S (line 9). Then, the new selected

random node c is removed from the CL (line 11) and replaces node r (line 10). Once all the nodes in the transformed graph are added to S , the algorithm terminates (line 5). In this stage, the output is S in which the order of all critical nodes are known. However, to find the route for the original graph, we need to perform a subroutine procedure which is shown in Algorithm 5. This procedure is used in Algorithms 4 and 6.

Algorithm 4 GRASP construction step for ML-RCP

```

1:  $S \leftarrow depot$ 
2: Initialize  $CL$ , the critical node list
3:  $CL \leftarrow CL \setminus \{depot\}$ 
4:  $r \leftarrow depot$ 
5: while  $CL \neq \emptyset$  do
6:   Sort  $CL$  in non-decreasing order of distance from node  $r$ .
7:   Build  $RCL$  by choosing the first  $\alpha$  percent of the nodes in sorted  $CL$ .
8:   Select a random node  $c \in RCL$ .
9:   Add  $c$  to  $S$ .
10:   $r \leftarrow c$ 
11:   $CL \leftarrow CL \setminus \{r\}$ 
12: end while
13: Apply the subroutine procedure presented in Algorithm 5.
14: return  $x_G$ 

```

Algorithm 5 Subroutine procedure for GRASP and VNS

Input:

Sequence S (shows the order of all critical nodes)

- 1: Find the shortest path between each pair of critical nodes (with respect to the order of S) and form route x .
 - 2: Apply the objective correction procedure (presented in Section 3.1.2.1).
 - 3: Calculate the updated objective function value.
 - 4: **return** route x and its objective function value.
-

The output of this subroutine is route x which is obtained from the order of the critical nodes in the list S and the shortest paths from the original graph.

At the end, the GRASP constructs the initial solution and returns the initial feasible route (line 14).

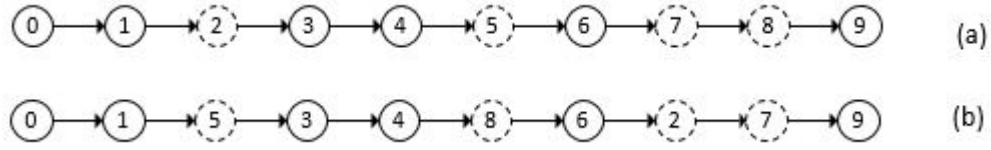


Figure 3.3: An example illustrating the shaking procedure

3.1.2.2.2 VNS

For the improvement procedure, a local search based on VNS is executed. This meta-heuristic method was first introduced by Mladenović and Hansen (1997). The main idea of VNS is to use various neighborhood structures which are consecutively repeated. Switching from one neighborhood to another helps the algorithm to escape from a local optimum. However, it is still possible to get stuck in a local optimum. To avoid it, a procedure called shaking is added. The goal of using shaking is to disturb some parts of the current solution without affecting the entire solution. In general, the shaking procedure is executed outside the neighborhood search loop. Nevertheless, some authors used it between each pair of neighborhoods. According to our observations from the experiments given in chapter 4, we prefer to execute the shaking procedure outside the neighborhood search loop. In the shaking procedure, we select 20% of the nodes in the current solution randomly with equal probabilities. Then, we remove them and insert them randomly with equal probabilities in the empty positions in different orders. Fig. 3.3 depicts an example of the shaking procedure in which nodes 2, 5, 7 and 8 are removed from the current solution (Fig. 3.3(a)) and then inserted in different positions (Fig. 3.3(b)). Note that in the example, we removed more than 20% of the nodes for illustration purposes. We also tested other percentages for the shaking procedure. Nevertheless, for most instances, 20% gives the best solution. The outline of the VNS for ML-RCP is given in Algorithm 6. Note that in the VNS, we again apply the neighborhood search on the transformed complete graph as in the previous procedures. Let $N_1, N_2, \dots, N_{k_{max}}$ be the neighborhood structures. When a local search is implemented on a solution x using a neighborhood structure

N_k , we denote the new solution by $N_k(x)$. Let x_G be the initial solution obtained from GRASP construction step (Algorithm 4) and x_V the best solution found from each iteration of VNS which is shaken. We set a time limit T_{max} for the metaheuristic (line 2). In order to capture a large variety of initial solutions, we change α dynamically while running the metaheuristic within T_{max} . Suppose $\alpha_1, \alpha_2, \dots, \alpha_p$ are to be used. Then we divide T_{max} to p intervals and use α_i during the i^{th} time interval. The body of the VNS method is in lines (6-15) of Algorithm 6. The neighborhood structures are applied one by one in the order of k . If a neighborhood structure gives a better solution than the previous one, the new solution is replaced as the new best one and the algorithm starts from the first structure to test and find another solution. On the other hand, if the algorithm is unable to find a better solution in the current structure, it goes to the next neighborhood structure. In addition, we define a set of moves to prevent repetitive orders which makes the procedure faster to reach the best solution. The VNS algorithm terminates when the given time limit T_{max} is reached. At the end of each iteration, the current best solution (x_B) is shaken (line 16) in order to prevent staying in a local optimum. This solution is used in the next iteration and will be compared with the new generated x_G (line 4) and the best solution of x_G and x_V is taken as the new initial solution for the VNS. Note that we apply the subroutine procedure introduced in Algorithm 5 in lines 10 and 17 to use the right objective value when comparing two solutions.

Next, we describe each of the neighborhood structures performed in VNS. In our search we use four neighborhood structures which are known to perform well in vehicle routing problems. We use the neighborhoods in the transformed graph in the order that we present them. Note that through the preliminary experiments, we observed that the performance of the metaheuristic is relatively insensitive to the order of the neighborhoods. Thus, we used the orders from small to large neighborhoods, as it is common in the VNS algorithm. When we evaluate the objective function value of the newly found solution, x' , we calculate latency since this takes only $O(|V_C|)$ time.

- **Random swap** (N_1): In this neighborhood structure, two nodes are randomly

Algorithm 6 The metaheuristic for ML-RCP including GRASP construction step and VNS

Let the latency of a solution x be denoted by $l(x)$

- 1: x_V is empty and $l(x_V) =$ a large number
- 2: **while** time limit is not over **do**
- 3: Obtain an initial solution x_G (from GRASP construction step).
- 4: $l(x_B) = \min\{l(x_G), l(x_V)\}$, Let x_B be the solution with better objective value between x_G and x_V
- 5: $k \leftarrow 1$
- 6: **while** $k \leq k_{max}$ **do**
- 7: local search: $x' \leftarrow N_k(x_B)$
- 8: **if** x' better than x_B **then**
- 9: $x_B \leftarrow x'$
- 10: Apply the subroutine procedure presented in Algorithm 5.
- 11: $k \leftarrow 1$
- 12: **else**
- 13: $k \leftarrow k + 1$
- 14: **end if**
- 15: **end while**
- 16: Shake: $x_V \leftarrow x_B$
- 17: Apply the subroutine procedure presented in Algorithm 5.
- 18: **end while**
- 19: **return** x_B

selected and the position of this pair of nodes are exchanged.

- **Random Swap-adjacent** (N_2): One node is randomly selected and its position is swapped with the right or left adjacent node with equal probability.
- **Random remove-insert to the end** (N_3): One node is randomly selected and removed and is inserted after the last node.
- **2-opt** (N_4): Each pair of non-adjacent edges are removed and other two edges are inserted in order to reconnect the nodes and form a feasible solution.

Note that we used the order of these neighborhood structures based on the studies engaged with minimum latency problem in literature (i.e., Salehipour et al. (2011)) to get the best possible solutions for our problem as well. The order presented here is not nested. However, we conducted a sensitivity analysis to see the difference in solutions when the order of neighborhood structures changes which is provided in Section 4.5. The results denote that the solutions are almost insensitive to the order of neighborhoods.

3.2 Problem definition associated with MML-RCP

We represent the disaster-affected road network in an undirected graph $G = (V, E)$, where V and E denote the set of nodes and set of edges in graph G , respectively. A traversal time t_{ij} associated with edge $(i, j) \in E$ is given, where all traversal times are symmetric. That is, t_{ij} is the same for both directions (i, j) and (j, i) . Blocked edges are known and represented by the set $B \subseteq E$. In addition, extra times, namely clearing times for edges associated with blocked edges are given as u_{ij} . Similar to t_{ij} , u_{ij} is also symmetric. We define a set $V_C \subseteq V$ consisting of all critical nodes and the depot node(s). The remaining network elements are the intact edges and the non-critical nodes that act as intermediate edges and nodes in between visiting the critical nodes. The clearing teams or vehicles are located in the depot node(s) at the beginning of the road clearance operation. Each vehicle departs from a specific depot

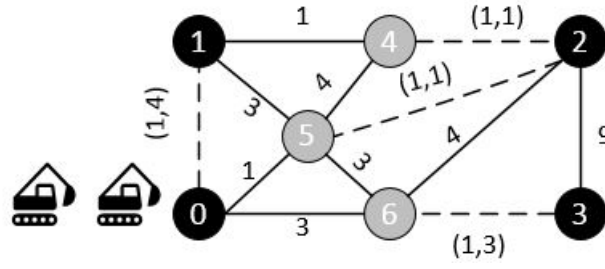


Figure 3.4: Example of MML-RCP

node to visit a subset of the critical nodes and finishes its route at one of the critical nodes. Here we note that each vehicle must visit at least one critical node through its route and all critical nodes must be visited by the vehicles at the end of the operation. Furthermore, each vehicle may clear some blocked edges in order to reach the next critical node. Some of the non-critical nodes may be visited by the vehicles through their routes that we call them as “intermediate” nodes. Recall that similar to other studies, we assume a disjoint path for each vehicle, where all of the critical nodes must be visited exactly once and non-critical nodes may be visited but at most one time. The problem is to find an optimal route for each vehicle such that total latency of all critical nodes is minimized. In addition, the solution specifies which blocked edges should be cleared in what order by each vehicle. This problem can be considered as a heuristic approach since we eliminate the synchronization between the vehicles by assuming disjoint paths between them. However, this heuristic approach gives feasible solutions and is much faster compared to assuming synchronization between vehicles.

Fig. 3.4 shows an example of MML-RCP with two vehicles positioned in the depot node 0 (we will further provide an example with multiple depots). Nodes 0, 1, 2 and 3 are critical and the other nodes (4, 5 and 6) are non-critical. The blocked edges are represented by dashed lines. The traversal and clearing times (for the blocked edges) are shown on top of each edge. Here we describe a feasible solution for each vehicle that departs from the depot node, namely node 0. The route of the first vehicle includes nodes 0-5-1, where the latency of critical node 1 is 4. Intermediate

node 5 has been visited in the route. Note that no blocked edges exist in the route of the first vehicle. The route of the second vehicle includes nodes 0-6-3-2, where the latency of critical nodes 3 and 2 are 7 and 16, respectively. In this route, node 6 is an intermediate node and the vehicle needs to clear the blocked edge (6, 3) in order to reach critical node 3. At the end, the vehicles have visited all critical nodes with a total latency of 27 ($4+7+16$), which includes both traversal and clearing times in the objective value.

3.2.1 Mathematical model

Here let us state that the problem is NP-hard since it generalizes the multi-vehicle MLP in two aspects. When there are no blocked edges and all nodes are critical we obtain the multi-vehicle MLP (recall that Sahni and Gonzalez (1976) proved that MLP is NP-hard, so is the multi-vehicle MLP). In order to find optimal solutions for the MML-RCP within a reasonable time limit, we present a novel multi-level network which is based on a multi-level network introduced by Angel-Bello et al. (2017). Having smaller execution times is the advantage of using a multi-level network in MLP. Then, We propose a mathematical model defined on a multi-level network which is different from those in Angel-Bello et al. (2017). First, we briefly explain the standard multi-level network proposed by Angel-Bello et al. (2017) (recall that all nodes must be visited by the vehicles in their models) and then we describe ours with an example to make the concept clearer. Furthermore, we list the differences of our mathematical model with the ones in Angel-Bello et al. (2017).

Fig. 3.5 shows the multi-level network provided by Angel-Bello et al. (2017), in which the number of levels is equal to $N = |V| - m$, where V and m are the set of nodes and the number of vehicles, respectively. The reason for considering this number of levels is because of the assumption that forces all vehicles to visit at least one node (in our problem each vehicle must visit at least one critical node). Level 1 consists of a copy of nodes of the set V except the depot node. Levels 2, 3, $\dots$, N consist of nodes that are copies of the nodes in V . Finally, level $N+1$ consists of a

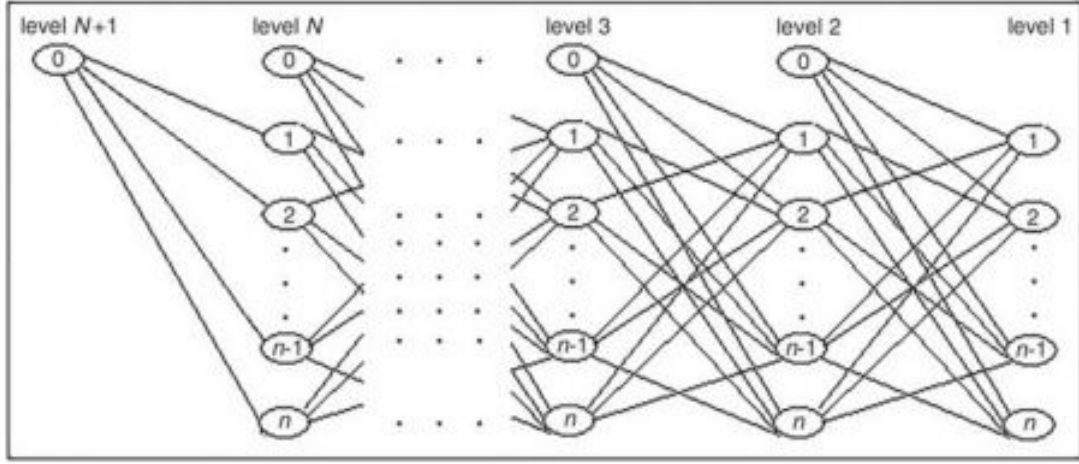


Figure 3.5: Graphical representation of the multi-level network in (Angel-Bello et al. 2017)

copy of only the depot node. In any feasible solution, each path starts from a copy of the depot node (located in one of the levels 2, 3, $\dots$, $N+1$) and visits the lower levels and reaches level 1 for the final node. Each vehicle must visit a single node in each level lower than the starting node. For instance, if a vehicle starts its path from the depot node in level 4, a single node in each level 3, 2 and 1 must be visited. Note that all vehicles depart from the depot node, which is positioned in levels 2, 3, $\dots$ and $N+1$.

Fig. 3.6 demonstrates a feasible solution associated with the (Angel-Bello et al. 2017)'s multi-level network. There are 8 nodes (including the depot node 0) in this example, where all of them must be visited. Two vehicles are assigned for this model. Vehicle 1, whose route shown by narrow black lines, starts its route from the depot node in level 5 by visiting nodes 1, 4, 7 and 6. On the other hand, vehicle 2, whose route shown by thick black lines, departs from the depot node in level 4 while visiting nodes 3, 2 and 5. Thus, all of the nodes are visited by the vehicles.

We derive a novel multi-level network from Angel-Bello et al. (2017)'s network whose differences are listed below:

1. Unlike Angel-Bello et al. (2017)'s network, we define levels only for the critical

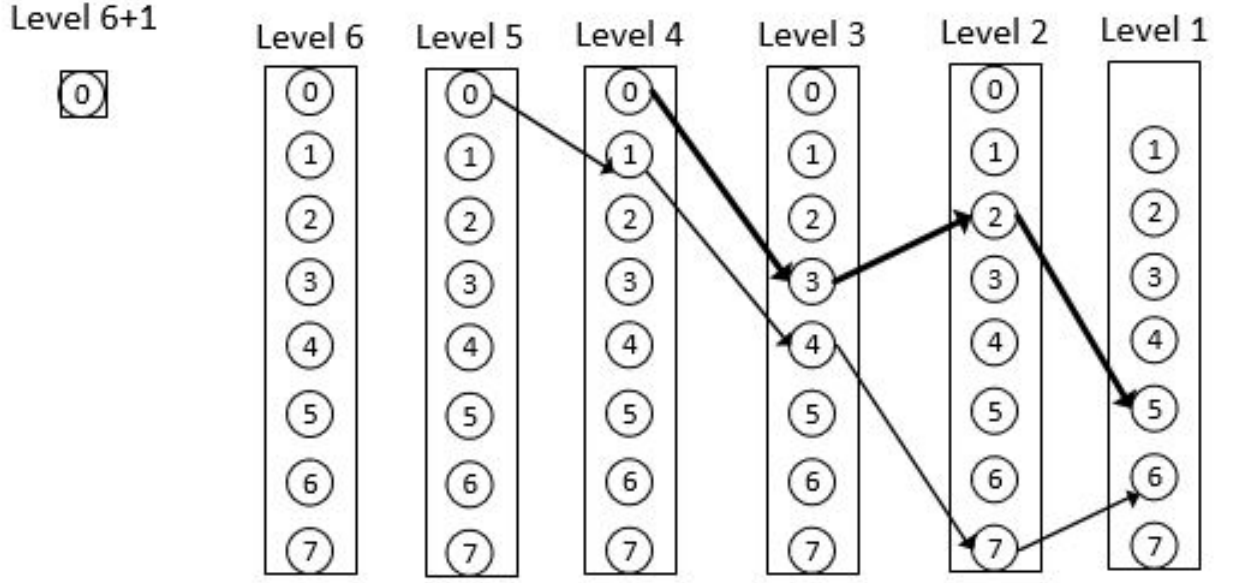


Figure 3.6: A feasible example of the multi-level network in (Angel-Bello et al. 2017)

nodes (we call them critical levels), which varies from 0 to $N = |V_C| - m$, where V_C is the set of critical nodes including the depot node.

2. We consider non-critical nodes between two levels n and $n+1$. In other words, we put a copy of all non-critical nodes between every two levels.
3. Since a subset of nodes (including critical nodes and some non-critical nodes) is visited through the routes, the definition of decision variables and constraints changes significantly.
4. Our multi-level network is able to solve the problems engaged with multiple depots (see Section 3.2.1.1).

Let us explain our multi-level network with a small example. Fig. 3.7 (a) demonstrates our multi-level network constructed from a complete network with nodes 0, 1, 2, ... and 7, where node 0 is the depot node and nodes 0, 2, 4, 5 and 7 are critical nodes. In addition, nodes 1, 3 and 6 are non-critical and hence, may or may not be

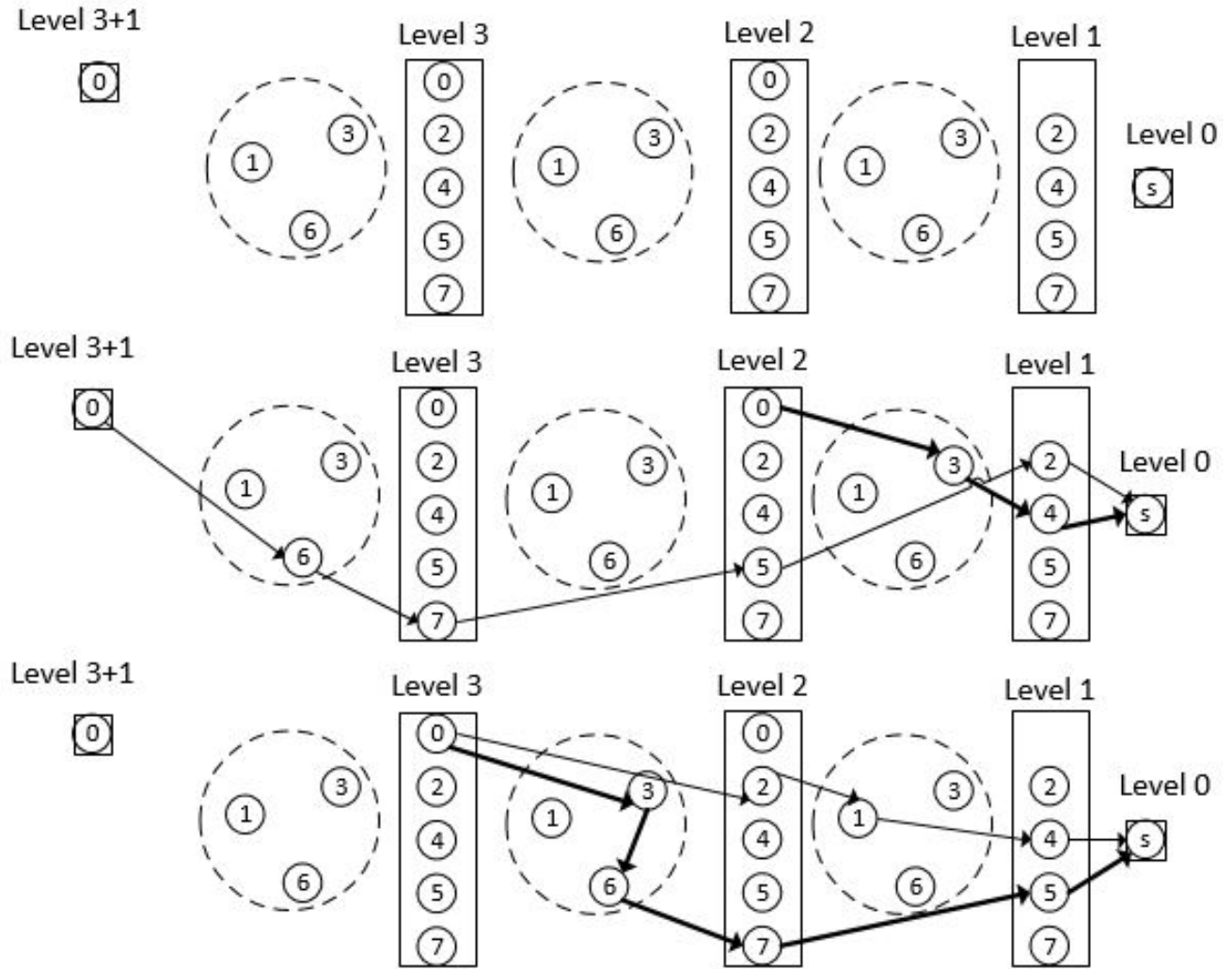


Figure 3.7: An example of the multi-level network

visited through the routes. Two vehicles are assigned to visit all the critical nodes, whose routes are depicted with narrow and thick black lines (we call them narrow and thick black vehicles). The number of levels for critical levels varies from 0 to $N = 5 - 2 = 3$. Note that we do not consider any levels for the non-critical nodes and instead we put a copy of all non-critical nodes between two levels n and $n+1$.

Fig. 3.7 (b) and 3.7 (c) show some of the feasible routes provided by MML-RCP. As we observe, each vehicle visits at least one critical node. A vehicle may or may not visit the non-critical nodes through its route. For instance, in Fig. 3.7 (b), the

narrow black vehicle starts its route from level 3+1, visits node 6 and then the critical node 7. On the other hand, the vehicle directly continues to reach the critical node 5 without visiting any non-critical nodes. In another example depicted in Fig. 3.7 (c), the thick black vehicle leaves the depot in level 3 because in the solution, it needs to visit 2 critical nodes. Then it visits non-critical nodes 3 and 6 in its route. Afterwards, the vehicle continues on the path to reach a critical node. Recall that all paths must be disjoint and the non-critical nodes can be visited at most once. With this in mind, we define travel time (c_{ij}) of edge $(i, j) \in E$ as $t_{ij} + u_{ij}B_{ij}$, where t_{ij} and u_{ij} are the traversal and clearing times of edge $(i, j) \in E$, respectively. The parameter B_{ij} is equal to one, if $(i, j) \in E$ is blocked; and zero, otherwise. Thus, if a vehicle traverses blocked edge $(i, j) \in B$, the travel time of this edge is $c_{ij} = t_{ij} + u_{ij}$. Note that assuming disjoint paths happens where multiple vehicles exist in the depot(s). As a result, if the problem is solved for only one vehicle, we do not need to consider disjoint paths through the model. In addition, we note that since we eliminate the synchronization between the vehicles, the following formulation is considered as a heuristic approach.

Recall that the input network for MML-RCP should be complete (where all nodes should have access to each other with a travel time which is not equal to a big number).

Given the provided description, we introduce our mathematical model for MML-RCP in the following such that only a single depot exists. We will further explain in Section 3.2.1.1 how to tackle MML-RCP with multiple depots.

s = dummy sink node

m = Number of vehicles

$N = |V_C| - m$

Sets:

V_C : set of critical nodes consisting of the depot, 0, and the critical nodes

V : Set of all nodes including the dummy sink node

Parameters:

c_{ij} : travel time of edge $(i, j) \in E$

Decision variables:

$y_{ijkl}^r = 1$, if the edge $(i, j) \in E$ is used while going from critical node $k \in V_C$ in level $r + 1$ to critical node $l \in V_C$ in level r ; 0, otherwise (Note that we define y as a continuous variable in the model not as a binary one)

$x_{ikl}^r = 1$, if node $i \in V$ is visited while going from critical node $k \in V_C$ in level $r + 1$ to critical node $l \in V_C$ in level r ; 0, otherwise

$x_{kkl}^r = 1$, if critical node $k \in V_C$ is visited while going from critical node $k \in V_C$ in level $r + 1$ to critical node $l \in V_C$ in level r ; 0, otherwise

$x_{lkl}^r = 1$, if critical node $l \in V_C$ is visited while going from critical node $k \in V_C$ in level $r + 1$ to critical node $l \in V_C$ in level r ; 0, otherwise

$$\min \sum_{(i,j) \in E, i \neq j} \sum_{l \in V_C \setminus \{0\}} \sum_{r=1}^N c_{ij}^r y_{ij0l}^r + \sum_{(i,j) \in E, i \neq j} \sum_{k \in V_C \setminus \{0\}, k \neq l} \sum_{l \in V_C \setminus \{0\}} \sum_{r=1}^{N-1} c_{ij}^r y_{ijkl}^r \quad (3.23)$$

s.t.

$$\sum_{l \in V_C \setminus \{0, k\}} \sum_{r=1}^{N-1} x_{kkl}^r + x_{kks}^0 = 1 \quad \forall k \in V_C \setminus \{0\} \quad (3.24)$$

$$\sum_{k \in V_C \setminus \{0\}} x_{kks}^0 = m \quad (3.25)$$

$$\sum_{l \in V_C \setminus \{0\}} \sum_{k \in V_C, k \neq l} \sum_{r=1}^N x_{ikl}^r \leq 1 \quad i \in V \setminus V_C, i \neq s \quad (3.26)$$

$$\sum_{l \in V_C} \sum_{k \in V_C \setminus \{0\}, k \neq l} \sum_{i \in (V \setminus V_C) \cup \{l\}, i \neq s} y_{iklk}^1 = m \quad (3.27)$$

$$\sum_{r=1}^N \sum_{j \in (V \setminus V_C) \cup \{l\}, j \neq s} \sum_{l \in V_C \setminus \{0\}} y_{0j0l}^r = m \quad (3.28)$$

$$\sum_{j \in (V \setminus V_C) \cup \{l\}, j \neq i, s} y_{ijkl}^r = x_{ikl}^r \quad r = 1, 2, \dots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l, \\ i \in (V \setminus V_C) \cup \{k\}, i \neq s \quad (3.29)$$

$$y_{0j0l}^r + \sum_{i \in (V \setminus V_C) \cup \{k\}, i \neq j, s} y_{ijkl}^r = x_{jkl}^r \quad r = 1, 2, \dots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l, \\ j \in (V \setminus V_C) \cup \{l\}, j \neq s \quad (3.30)$$

$$\sum_{i \in (V \setminus V_C) \cup \{l\}} \sum_{l \in V_C, l \neq k} y_{iklk}^r = \sum_{w \in V_C \setminus \{0\}, w \neq k} x_{kkw}^{r-1} \quad k \in V_C \setminus \{0\}, r = 2, 3, \dots, N-1 \quad (3.31)$$

$$\sum_{i \in (V \setminus V_C) \cup \{l\}, i \neq s} y_{iklk}^1 = x_{kks}^0 \quad k \in V_C \setminus \{0\}, l \in V_C, k \neq l \quad (3.32)$$

$$\sum_{l \in V_C, l \neq k} x_{klk}^1 = x_{kks}^0 \quad k \in V_C \setminus \{0\} \quad (3.33)$$

$$x_{kkl}^r = x_{lkl}^r \quad r = 1, 2, \dots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l \quad (3.34)$$

$$x_{00l}^N = x_{l0l}^N \quad l \in V_C \setminus \{0\} \quad (3.35)$$

$$\sum_{i \in (V \setminus V_C) \cup \{0\}} y_{ij0l}^N = x_{j0l}^N \quad l \in V_C \setminus \{0\}, j \in (V \setminus V_C) \cup \{l\}, j \neq s \quad (3.36)$$

$$\sum_{j \in (V \setminus V_C) \cup \{l\}} y_{ij0l}^N = x_{i0l}^N \quad l \in V_C \setminus \{0\}, i \in (V \setminus V_C) \cup \{0\}, i \neq s \quad (3.37)$$

$$\sum_{k \in V_C \setminus \{0, l\}} x_{llk}^{N-1} = x_{l0l}^N \quad l \in V_C \setminus \{0\} \quad (3.38)$$

$$\sum_{i \in (V \setminus V_C) \cup \{0\}} y_{ik0k}^N = \sum_{w \in V_C \setminus \{0\}, w \neq k} x_{kkw}^{N-1} \quad k \in V_C \setminus \{0\} \quad (3.39)$$

$$y_{ijkl}^r \geq 0 \quad i, j \in V, k, l \in V_C, k \neq l, r = 1, 2, \dots, N \quad (3.40)$$

$$x_{ikl}^r \in \{0, 1\} \quad i \in V, k, l \in V_C, k \neq l, r = 0, 1, \dots, N \quad (3.41)$$

The objective function (3.23) minimizes the total latency of all critical nodes. By Constraints (3.24) each critical node is visited exactly once. Constraint (3.25) guarantees that all vehicles must go to the dummy sink node at the end of their routes. The non-critical nodes may or may not be visited (at most once) due to Constraints (3.26). Constraint (3.27) forces all vehicles to visit level 1 at one of the critical nodes

as a destination node. Constraint (3.28) ensures that exactly m disjoint paths are assigned to the vehicles. Constraints (3.29-3.39) are the connectivity constraints that guarantee the continuity of the paths between each consecutive critical level. We divide these connectivity constraints into two groups of constraints. The first group (3.29-3.35) are the connectivity constraints belonging to levels $0, 1, 2, \dots$ and N . Since in level $N+1$ only the depot node exists, we have to consider separate constraints for this level (Constraints (3.36-3.39)) to ensure the connectivity of each path. Moreover, these constraints guarantee the continuity of the paths in case the non-critical nodes are visited between critical levels. Eventually, Constraints (3.40) and (3.41) define the domains of the variables. Note that given the properties of the variables and their coefficients and the constraints, we defined decision variable y as a continuous variable (rather than a binary variable), decreasing the computational times significantly.

It is worth mentioning that considering the synchronization between the vehicles yields more realistic solutions where we do not assume disjoint paths. In other words, when a vehicle clears a blocked edge, the other vehicles are not allowed to traverse that edge until it is unblocked completely. In latter traverses, only the traversing time of that edge should be taken into account for further calculations (clearing time of a blocked edge incurs only one time). In the following, we present the mathematical model of MML-RCP where the synchronization between the vehicles are incorporated to the model (adapted from Akbari and Salman (2017b)). However, note that the following mathematical model is very weak in terms of run times and is able to solve the instances up to 13 nodes, 4 critical nodes and 2 vehicles. Once we add more nodes, the model is almost unable to solve the problem even within 5-hour time limit. Thus, assuming disjoint paths enable the model to be solved for larger instances.

The decision variables are presented in Table 3.1. In addition, there are extra sets and parameters used in the following formulation:

The walks start from the depots and end in a dummy sink node, indexed as $n + 1$. D denotes the set of depots and P_d is the set of vehicles which are initially positioned in depot $d \in D$.

Table 3.1: Decision variables used in the model associated with synchronization

Notation	Type	Definition
$x_{ijl}^k, \forall (i, j), (j, l) \in A, \forall k \in K$	binary	If vehicle k crosses (j, l) right after (i, j) in its walk
$z_{ijl}^k, \forall (i, j), (j, l) \in A$	binary	If (i, j) is unblocked by vehicle k
$y_{ijl}^k, \forall j \in V_C, \forall i : (i, j) \in A, \forall l : (j, l) \in A, \forall k \in K$	binary	If vehicle k reaches critical node j for the first time while crossing (j, l) right after (i, j) in its walk
$f_{ij}^k, \forall (i, j) \in A, \forall k \in K$	continuous	The flow corresponding to vehicle k 's route on (i, j) from node i to j
$v_i^k, \forall i \in V \cup \{n+1\}, \forall k \in K$	continuous	Number of times vehicle k visits node i
$t_{ijl}^k, \forall (i, j), (j, l) \in A, \forall k \in K$	continuous	The time that vehicle k arrives to node j from i before going to l
$s_{ijl}^k, \forall (i, j), (j, l) \in A$	continuous	If arc (i, j) is in the walk of vehicle k ; it is the earliest time that vehicle k can enter $\{i, j\}$. Otherwise, it is 0
$\tau_{ijl}^k, \forall j \in V_C, \forall i : (i, j) \in A, \forall l : (j, l) \in A, \forall k \in K$	continuous	The time that vehicle k arrives to node j from i before going to l . Otherwise becomes M
$LL_{ijl}^k, \forall j \in V_C, \forall i : (i, j) \in A, \forall l : (j, l) \in A, \forall k \in K$	continuous	The first time that vehicle k arrives to node j from i before going to l
$T_j, \forall j \in V_C$	continuous	latency of node j

$$\min \sum_{j \in V_C} T_j \quad (3.42)$$

$$T_j \leq \tau_{ijl}^k \quad \forall j \in V_C, i : (i, j) \in A, l : (j, l) \in A, \forall k \in K \quad (3.43)$$

$$\sum_{l: (j, l) \in A} \sum_{i: (i, j) \in A} \sum_{k \in K} y_{ijl}^k = 1 \quad \forall j \in V_C \quad (3.44)$$

$$T_j = \sum_{l: (j, l) \in A} \sum_{i: (i, j) \in A} \sum_{k \in K} LL_{ijl}^k, \quad j \in V_C \quad (3.45)$$

$$LL_{ijl}^k \geq \tau_{ijl}^k - M(1 - y_{ijl}^k) \quad \forall j \in V_C, i : (i, j) \in A, l : (j, l) \in A, \forall k \in K \quad (3.46)$$

$$LL_{ijl}^k \leq \tau_{ijl}^k, \quad \forall j \in V_C, i : (i, j) \in A, l : (j, l) \in A, \forall k \in K \quad (3.47)$$

$$\tau_{ijl}^k = M(1 - x_{ijl}^k) + t_{ijl}^k \quad \forall j \in V_C, i : (i, j) \in A, l : (j, l) \in A, \forall k \in K \quad (3.48)$$

$$y_{ijl}^k \leq x_{ijl}^k \quad \forall j \in V_C, i : (i, j) \in A, l : (j, l) \in A, \forall k \in K \quad (3.49)$$

$$x_{ijl}^k \leq \sum_{h: (l, h) \in A} x_{jlh}^k \quad \forall (i, j) \in A, k \in K, l : (j, l) \in A, \forall k \notin P_i \quad (3.50)$$

$$x_{ijl}^k \leq \sum_{h: (h, i) \in A} x_{hij}^k \quad \forall (i, j) \in A, \forall k \in K, l : (j, l) \in A, \forall k \notin P_i \quad (3.51)$$

$$\sum_{h: (h, i) \in A} x_{hij}^k \leq 1 \quad \forall (i, j) \in A, \forall k \in K \quad (3.52)$$

$$\sum_{l: (j, l) \in A} x_{ijl}^k \leq 1 \quad \forall (i, j) \in A, \forall k \in K \quad (3.53)$$

$$\sum_{j: (d, j) \in A} \sum_{l: (j, l) \in A} x_{djl}^k = 1, \quad \forall d \in D, \forall k \in P_d \quad (3.54)$$

$$\sum_{(i,j) \in A} x_{ij(n+1)}^k = 1 \quad \forall k \in K \quad (3.55)$$

$$\sum_{l:(j,l) \in A} x_{ijl}^k \geq z_{ij}^k \quad \forall (i,j) \in B, k \in K \quad (3.56)$$

$$\sum_{l:(j,l) \in A} x_{ijl}^k + \sum_{h:(i,h) \in A} x_{jih}^k \leq 2 \left(\sum_{\kappa=1}^K z_{ij}^{\kappa} + \sum_{\kappa=1}^K z_{ji}^{\kappa} \right) \quad \forall (i,j) \in B, \forall k \in K \quad (3.57)$$

$$\sum_{\kappa=1}^K (z_{ij}^{\kappa} + z_{ji}^{\kappa}) \leq 1 \quad \forall (i,j) \in B \quad (3.58)$$

$$\sum_{j \in V \cup \{(n+1)\}: (i,j) \in A} (f_{ij}^k - f_{ji}^k) = -v_i^k \quad \forall k \in K, \forall i \in V \cup \{(n+1)\} \setminus D \quad (3.59)$$

$$\sum_{j \in V \cup \{(n+1)\}} (f_{dj}^k - f_{jd}^k) = \sum_{i \in V \cup \{(n+1)\} \setminus \{d\}} v_i^k \quad \forall k \in P_d, \forall d \in D \quad (3.60)$$

$$\sum_{j \in V} f_{j(n+1)}^k = 1 \quad \forall k \in K \quad (3.61)$$

$$f_{ij}^k \leq (n-1) \sum_{l:(j,l) \in A} x_{ijl}^k \quad \forall k \in K, \forall (i,j) \in A, j \in V \cup \{(n+1)\} \quad (3.62)$$

$$f_{ij}^k \geq \sum_{l:(j,l) \in A} x_{ijl}^k \quad \forall k \in K, \forall (i,j) \in A, j \in V \cup \{(n+1)\} \quad (3.63)$$

$$\sum_{j:(j,i) \in A: (i,l) \in A} x_{jil}^k = v_i^k \quad \forall k \in K, \forall i \in V \cup \{(n+1)\} \quad (3.64)$$

$$t_{ijl}^k \leq M x_{ijl}^k \quad \forall k \in K, i : (i,j) \in A, \forall l : (j,l) \in A \quad (3.65)$$

$$s_{ij}^k \leq \sum_{h:(h,i) \in A} t_{hij}^k + M (1 - z_{ij}^k) \quad \forall k \in K, \forall (i,j) \in B \quad (3.66)$$

$$s_{ij}^k \geq \sum_{h:(h,i) \in A} t_{hij}^k \quad \forall k \in K, \forall (i,j) \in B \quad (3.67)$$

$$s_{ij}^k \geq \sum_{l:(j,l) \in A} t_{ijl}^{\kappa} - M (1 - z_{ij}^{\kappa}) - 2M \left(1 - \sum_{h:(h,i) \in A} x_{hij}^k \right) \quad \forall k, \kappa \in K, \forall (i,j) \in B, \kappa \neq k \quad (3.68)$$

$$\sum_{l:(j,l) \in A} t_{ijl}^k \geq s_{ij}^k + b_{ij} z_{ij}^k + \sum_{l:(j,l) \in A} x_{ijl}^k c_{ij}, \forall k \in K \quad \forall (i,j) \in B \quad (3.69)$$

$$\sum_{l:(j,l) \in A} t_{ijl}^k \geq \sum_{h:(h,i) \in A} t_{hij}^k + \sum_{l:(j,l) \in A} x_{ijl}^k c_{ij} \quad \forall k \in K, \forall (i,j) \in A \setminus B \quad (3.70)$$

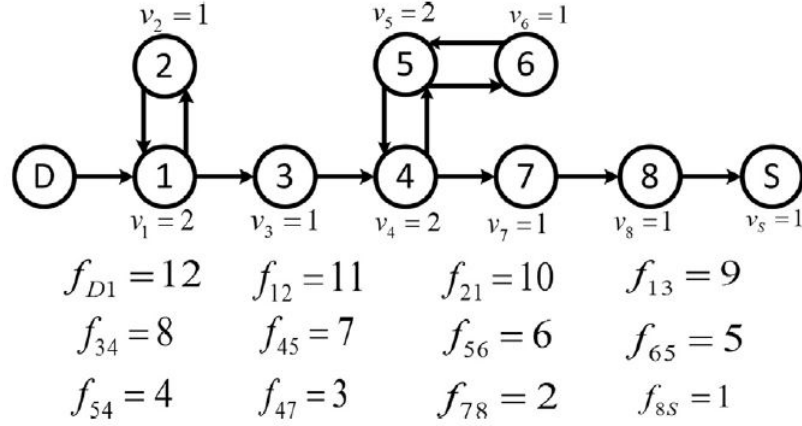


Figure 3.8: Demonstration of the flow formulation

$$x_{ijl}^k \in \{0, 1\}, \forall k \in K, \quad i : (i, j) \in A, \quad \forall l : (j, l) \in A \quad (3.71)$$

$$z_{ij}^k \in \{0, 1\}, \quad \forall (i, j) \in B, \quad \forall k \in K \quad (3.72)$$

$$f_{ij}^k \geq 0, \quad \forall (i, j) \in A, \quad \forall k \in K \quad (3.73)$$

$$v_i^k \geq 0, \quad \forall i \in V, \quad \forall k \in K \quad (3.74)$$

$$t_{ijl}^k \geq 0, \forall k \in K, \quad i : (i, j) \in A, \quad \forall l : (j, l) \in A \quad (3.75)$$

$$s_{ij}^k \geq 0, \quad \forall (i, j) \in B, \quad \forall k \in K \quad (3.76)$$

Objective function calculates the total latency until all critical nodes are visited by vehicles. Note that all walks end in the sink node indexed as $n + 1$. Constraints (3.43) guarantee that the latency of critical node j should be less than equal to all visiting times of node j by all vehicles. Constraints (3.44)- (3.48) force the model to pick the minimum visiting time of critical node j as the latency of node j . Constraints (3.44) ensure that only one of the visiting times of node j should be considered as the latency of critical node j . Constraints (3.48) provide the visiting time of critical node j for each vehicle visiting critical node j (On the other hand, if critical node j is not visited by a vehicle its visiting time becomes equal to big M). By constraints (3.46) and (3.47), LL_{ijl}^k finds the minimum time to reach critical node j over all vehicles and routes. As a result, Constraints (3.45) calculate the latency of critical node j . If no

vehicle traverses node j , thus, variable y should be zero as well which is shown by Constraints (3.49). Constraints from (3.50) to (3.53) are flow balance equations. A vehicle should enter an arc first, to leave it in the next step of its walk. It also cannot leave an arc unless it is traversed in the previous step of the walk. Constraints (3.50) and (3.51) set these properties. Each arc is traversed at most once by a particular vehicle, Constraints (3.52) and (3.53) enforce this property. Each vehicle starts its walk in its depot and ends it in the sink node. This property is ensured by Constraints (3.54) and (3.55). Constraints from (3.56) to (3.58) set the relation between x_{ijl}^k and z_{ij}^k decision variables. Constraint (3.56) forces that a blocked edge gets open only if one of the vehicles crosses it. Constraints (3.57) ensure that traversing a blocked edge is only possible after unblocking it by one of the vehicles. By (3.58), a blocked edge should be opened by at most one vehicle. In order to ensure connectivity of the walks, we define flow variables f_{ij}^k for every vehicle and for each arc $(i, j) \in A$. The net flow out of a depot node is the total number of visits to all nodes except the depot. For other nodes, net flow equals to the number of visits to the corresponding node. Fig. 3.8 gives an example of the usage of the flow variables and how they enable the route of a vehicle to be connected. In the given walk in Fig. 3.8, total number of visits is 12. A flow of 12 leaves the depot node and at each visit to any node, one unit of flow is left. Finally one unit of flow enters the sink node. Constraints (3.59) and (3.60) ensure that a vehicle leaves one unit of flow each time it visits a node. Constraints (3.61) ensures that walks end in the sink node. Constraint (3.62) does not allow flow on an edge unless it is traversed. Constraint (3.63) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Constraint (3.64) counts the number of times vehicle k visits node $i \in V$. Constraint sets (3.65) to (3.70) are for setting time limits. Constraint (3.65) ensures that, if $x_{ijl}^k = 0$, then $t_{ijl}^k = 0$, where M can be set to $\sum_{(i,j) \in A} c_{ij} + \sum_{(i,j) \in B} b_{ij}$. Constraints (3.66), (3.67) and (3.68) are used to calculate s_{ij}^k values. Variable s_{ij}^k shows the earliest time that edge $(i, j) \in B$ is traversable by vehicle k , if vehicle k is planned to cross it. On the other hand, s_{ij}^k should be forced to zero, if vehicle k does not traverse $(i, j) \in B$. Traversing a blocked

edge can only start by a vehicle when either the particular vehicle is the opener of the edge, or the edge is already unblocked by another vehicle. Due to constraints (3.66) and (3.67), if vehicle k is the opener of edge $(i, j) \in B$, the earliest time that it is traversable by k is the time when it arrives to (i, j) . On the other hand, if $z_{ij}^k = 0$, then (3.66) is redundant, and by (3.67) vehicle k can start traversing (i, j) no sooner than its arrival time to node i . If a vehicle traverses a blocked edge without opening it, it is only possible if the edge is unblocked earlier by another vehicle. In addition, $s_{ij}^k = 0$ should be satisfied, if vehicle k is not planned to cross $(i, j) \in B$. Constraint sets (3.69) and (3.70) calculate the arrival times to an edge (i, j) for $(i, j) \in B$ and $(i, j) \in A \setminus B$, respectively. Constraints (3.71)-(3.76) are the domains of the decision variables.

3.2.1.1 MML-RCP with multiple depots

In this section, we analyze a generalization of MML-RCP with multiple depots, which has not been addressed in the literature before. Here we assume that each of the vehicles might start their route from a different depot. We also note that multiple vehicles might be pre-positioned in the same depot. In addition, the number of vehicles in each depot should be known. In order to consider this generalization of MML-RCP, some properties associated with the mathematical model presented in Section 3.2.1 should be modified. Ahead of the modified mathematical model, we should modify our multi-level network. We define a depot set, namely D including all depots in the network. Since there are multiple depots, a copy of all depots should be put in each level except level 1. Recall that similar to the multi-level network in Fig. 3.5, a copy of only depot nodes exists in level $N + 1$. Moreover, the number of vehicles pre-positioned in each depot, should be given as a parameter prior to solving the mathematical model. Fig. 3.9 provides two feasible solutions of MML-RCP with multiple depots, where depot set $D = \{\alpha, \beta, \gamma\}$ consists of three depot nodes. Nodes 2, 4, 5 and 7 are critical, and nodes 1, 3 and 6 are non-critical that may or may not be visited by the vehicles. Two vehicles are assigned to this example, where the route of

each vehicle is represented by narrow and thick black lines. In the upper graph, the vehicles are pre-positioned in different depots (α and β), whereas in the lower graph, all vehicles are pre-positioned in the same depot (α).

In the following, we present the necessary modifications to the proposed mathematical model in Section 3.2.1 to address the multiple depots problem. First, we need to update the critical nodes set. That is, all of the depots should be added to set V_C (i.e. $V_C = \{\alpha, \beta, \gamma, 2, 4, 5, 7\}$ is the critical node set in the example associated with Fig. 3.9). Second, we replace index 0 with index $d \in D$ in all constraints given in Section 3.2.1. We define parameter N_d , $d \in D$, which denotes the number of vehicles pre-positioned in depot node $d \in D$, where $\sum_{d \in D} N_d = |k|$ ($|k|$ is the total number of vehicles). Then, we eliminate Constraint (3.28) and instead add the following constraint to the mathematical model:

$$\sum_{r=1}^N \sum_{j \in (V \setminus V_C) \cup \{l\}, j \neq s} \sum_{l \in V_C \setminus D} y_{djl}^r = N_d \quad d \in D \quad (3.77)$$

Constraints (3.77) set the number of vehicles designated to each depot. Note that the number of levels should be modified according to: $N = |V_C| - |k| - |D| - 1$, where $|k|$ is the number of vehicles. By means of this new modified mathematical model, we can solve the instances in MML-RCP engaged with multiple depots.

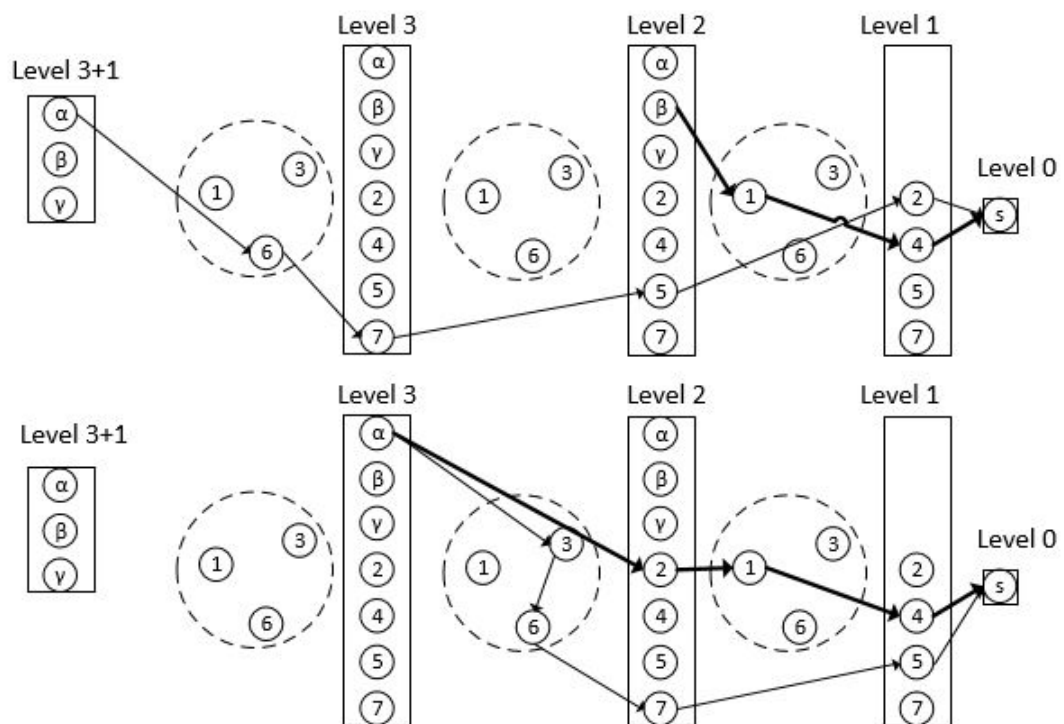


Figure 3.9: An example of the multi-level network with multiple depots

Chapter 4

COMPUTATIONAL RESULTS ASSOCIATED WITH ML-RCP

In this chapter, we compare the performance of the mathematical model proposed by Berktaş et al. (2016), the MIP model based-heuristic and the metaheuristic developed for ML-RCP with respect to the lower bounding approach (CMLB). We also have a comparison with Berktaş et al. (2016)'s method. The computational experiments were conducted with Python 2.7.12 using Gurobi 7.0 on Intel Xeon E5-2643 CPU @ 3.30GHz 3.30GHz (two processors) computer with 32 GB RAM, running under the Windows 7 operating system.

Before we begin to present the computational results associated with ML-RCP, we describe the data generation in the following section.

4.1 *Data sets*

In order to test both the exact and the heuristic solution methods, we used three data sets from Istanbul. The first one is based on the Kartal district in Istanbul. The second one is based on a simplified version of the Istanbul road network, and the third one is a detailed road network of the southwestern region of Istanbul.

4.1.1 *Kartal data*

The Kartal data is taken from Kilci et al. (2015) and Şahin et al. (2016) and is based on a complete network with 45 nodes (Fig. 4.1). Table 4.1 shows the node numbers of the critical nodes. We select subsets of size 3, 4, 7, 11 and 15 of the critical nodes in the runs involving this network. Detailed information about the generation of this data can be found in the two studies cited above. The data set includes 20 instances

Table 4.1: Subsets of critical nodes in Kartal network

3 selected critical nodes	14, 21, 22
4 selected critical nodes	26, 33, 41, 43
7 selected critical nodes	14, 21, 22, 26, 33, 41, 43
11 selected critical nodes	5, 14, 16, 21, 22, 26, 30, 33, 36, 41, 43
15 selected critical nodes	4, 5, 10, 14, 16, 21, 22, 26, 30, 33, 36, 38, 41, 43, 44

in which the set of critical nodes, travel times, and the number and locations of the blocked edges differ. The travel times are derived from actual travel distances between the nodes and the vehicle speed is assumed to be 20 km/h. The matrix is symmetric and satisfies the triangular inequality. In order to generate different scenarios, with the same network but different set of blocked edges, Şahin et al. (2016) assumed 4 levels of earthquake severity, which varies from 1 to 4. Setting the Severity of Earthquake (SOE) equal to 1 entails a less severe earthquake, while setting SOE=4 yields the highest number of blocked edges in the network. We refer to Şahin et al. (2016) for an explanation of how the blocked edges are selected.

Clearing times (u_{kl}) are calculated according to $u_{kl} = SOE * t_{kl} + U[0, \max t_{ij} \forall (i, j) \in A]$ depending on the SOE and the travel times (t_{ij}). In addition, a random number having a uniform distribution is added to the clearing times.

Five different instances are provided for each SOE in which the number of blocked edges are equal but the set of blocked edges differ in each SOE group. Since there are 4 levels of SOE, the total number of instances is 20. Table 4.2 presents the SOE and the corresponding number of blocked edges in these instances. For example, instances k1,...,k5 have the same number of blocked edges (124) with SOE=1, while the set of blocked edges are different.

4.1.2 Istanbul data

The simplified and southwestern Istanbul data sets were generated from the road network of the Istanbul city with 74 nodes and 179 edges for the simplified one, and 250 nodes and 539 edges for the southwestern region by Akbari and Salman (2017b).

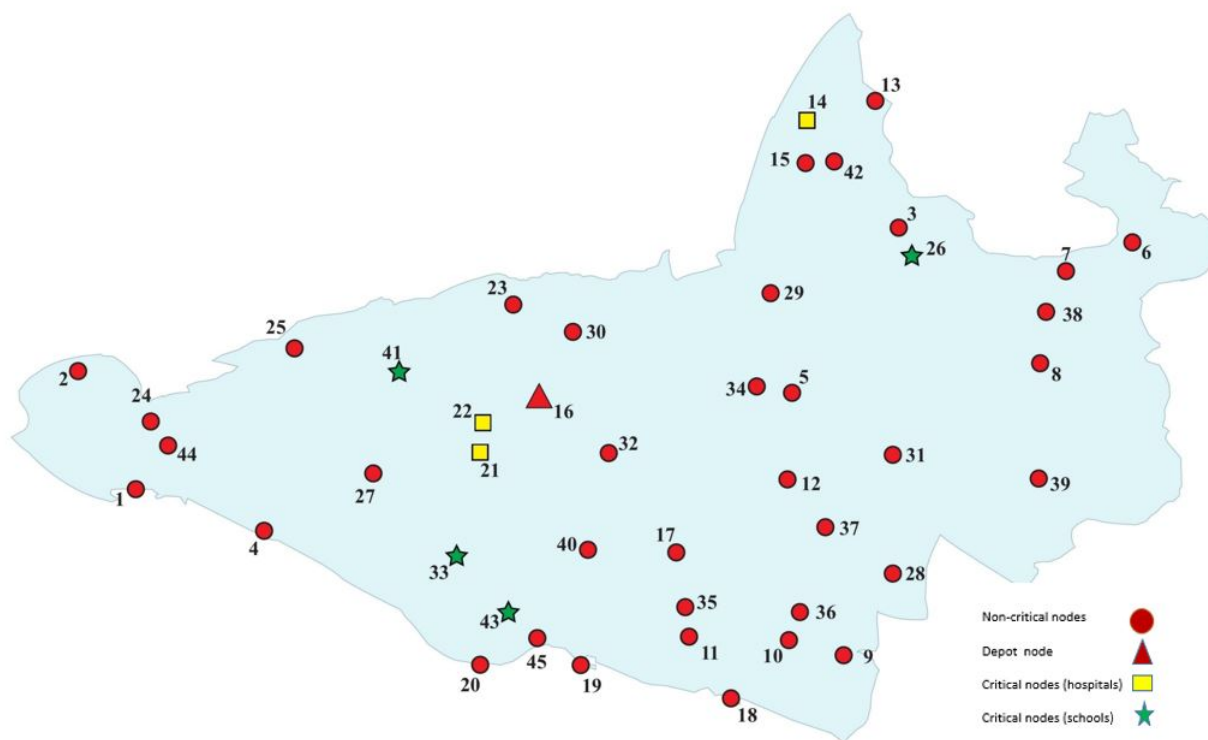


Figure 4.1: Kartal Network

Table 4.2: Kartal instances and the corresponding SOE and number of blocked-edge settings

Kartal instances	SOE	No. of blocked edges
k1,...,k5	1	124
k6,...,k10	2	441
k11,...,k15	3	574
k16,...,k20	4	806

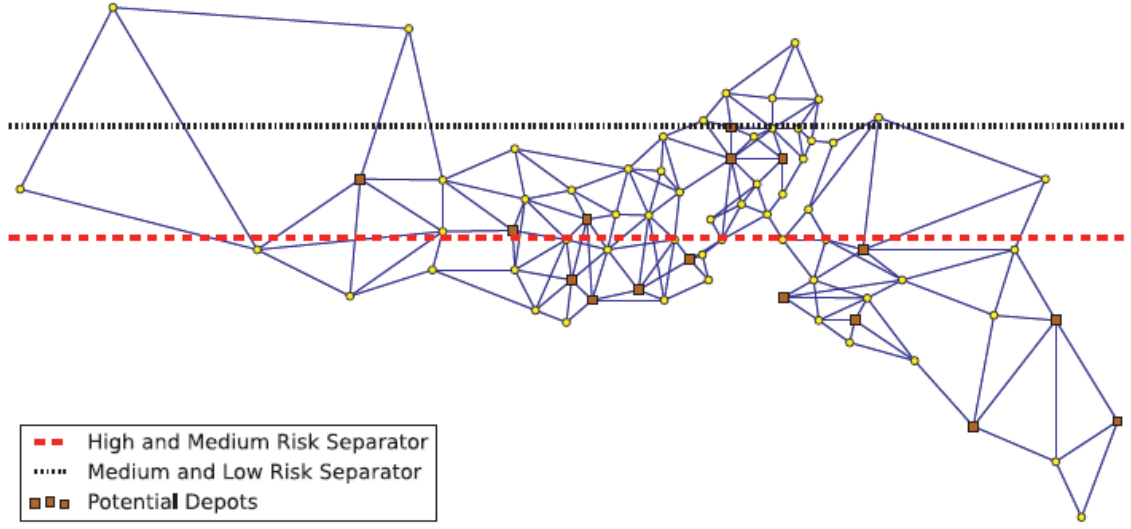


Figure 4.2: Istanbul Network

In these instances, to ensure that the solution definitely clears some blocked edges, the number of connected components was intentionally set between three and thirteen. More details as well as the graphical representation of these networks can be found in Akbari and Salman (2017b).

Based on the proximity of edges to the epicenter of the earthquake scenarios predicted in IMM (2002), the edges are categorized into three different groups as high, medium and low-risk edges. The probability that an edge is blocked after an earthquake for the low, medium and high-risk edges is 0.1, 0.2 and 0.3, respectively. The figure of the simplified Istanbul road network is given in Fig. 4.2. Edges located at the top of the higher horizontal line in Fig. 4.2 lie in the group of low-risk edges. Edges located between the two horizontal lines are in the group of medium-risk edges and the edges under the lower horizontal line are high-risk edges. If nodes of an edge are located in two different groups, we placed the particular edge in the more risky group.

We selected 35 potential nodes for the depot node and the critical nodes. In different instances (including 80 instances in total), the depot and the critical nodes are chosen randomly among the potential nodes with equal probabilities. Furthermore,

the blocked edges are selected randomly among all edges with equal probabilities. For simplified and southwestern Istanbul data sets, the traversal cost t_{ij} on edge $\{i, j\}$ is equal to the time it takes for a vehicle to go from node i to node j , assuming that the speed of the vehicle is 50 km/h, using the real road distances. The unblocking time, u_{ij} on edge $\{i, j\}$ is a random number generated according to: $u_{ij} = t_{ij} \cdot X$, where X has a uniform distribution between 100 and 300.

Overall, we test 80 instances for Istanbul data and 100 instances for Kartal data. We provide the Istanbul data in website <https://doi.org/10.6084/m9.figshare.7285019.v1>, which includes the set of critical nodes, the depot, blocked edges and their corresponding traversal and clearing times.

The computational results are summarized in Tables 4.3 - 4.9. The results are grouped according to the number of critical nodes, which is shown with No.CN, and the three networks. The name of the instances are in the first column. Let us name the objective function value found by the exact mathematical model Z_E , and by MIP model based-heuristic as Z_H . We let Z_{CMLB} denote the value of the lower bound. In addition, we provide the best objective values found by the metaheuristic within the time limit (T_{max}) of fifteen and thirty minutes for Kartal and Istanbul data sets as Z_{meta} . We do not show the objective values in the following tables and are rather concerned with the gaps. Computational times for the exact formulation, CMLB and the MIP model based-heuristic method are stated as T_E , T_{CMLB} and T_H in seconds, respectively. Moreover, we report the time when the metaheuristic finds the solution with the last improvement, T_{best} in seconds. As we mentioned in Section 3.1.2.2, we set different values for the parameter α to obtain various initial solutions taken from the GRASP construction step. Namely, we use zero, 10, 30 and 50 as the α value in the metaheuristic. We also tested values 15, 25, 35, 40 and 45 for α . However, we have not noticed any significant changes in the initial solutions. We also tested higher values for α through the preliminary experiments which show that selecting α greater than 50 leads to worse solutions. In addition, since 4 different values for parameter α are considered, we divided the time limit (T_{max}) to 4 intervals in each of which we

used the aforementioned α values for generating various initial solutions.

4.2 Kartal data results

Table 4.3 shows the computational times of Berktaş et al. (2016)'s MIP model (T_E), the MIP model based-heuristic (T_H) and the lower bounding (T_{CMLB}) solutions for Kartal data with 3, 4 and 7 critical nodes. Note that in order to be fair in the computational time comparison, we coded Berktaş et al. (2016)'s model and ran it in our workstation. In solving the exact model, as the number of critical nodes increases, CPU times increase exponentially. For instance, for k1, the CPU time is 2.6s for three critical nodes, 42.8s for four critical nodes and 589.4s for seven critical nodes. In addition to the number of critical nodes, also the number of blocked edges influences the computational times significantly. In Table 4.2, as SOE increases, the number of blocked edges increases as well. Consequently, the number of variables and constraints associated with the blocked edges increases. For example, instances k1 (generated with SOE=1) and k6 (generated with SOE=2) are solved in 589.4s and 3101.9s, respectively. Note that in both of the instances the number of critical nodes is seven.

We noticed that the solution of the MIP model based-heuristic is the same as the optimal one found by the exact MIP model (Z_E) in all of the 3, 4 and 7 critical node cases for all instances. Furthermore, the computational times of the MIP model based-heuristic is at most around 0.2s for these instances, while for the exact formulation, the computational time gets as high as over 10,000s. Also, the increase in the number of critical nodes has a tremendous effect in the computational time of the MIP.

To test the performance of the lower bounding procedure, we provide the computational time of CMLB (T_{CMLB}) and show the optimality gap (%) between the solutions found by the MIP model based-heuristic (as an upper bound) and CMLB (as a lower bound), which is equal to $\frac{Z_H - Z_{CMLB}}{Z_H} \times 100$. CMLB found the optimal solution in all instances with 3 critical nodes. For the cases with 4 and 7 critical nodes, CMLB found almost the optimal solution with the average gaps of 1.45% and

3.4%, respectively.

We report the results of Kartal data sets for 11 and 15 critical nodes in Table 4.4. Once the number of critical nodes increases to 11, the exact model could not solve any of the instances in the 3-hour time limit optimally. On the other hand, the MIP model based-heuristic finds a feasible solution which is either the same or better than the best solution found in the 3-hour time limit by the exact model in less than 1 second. The column shown with I (%) for 11 critical nodes presents how much better the solution found by the MIP model based-heuristic is, in comparison to the solution of the exact model (I (%) = $\frac{Z_E(3h) - Z_H}{Z_E(3h)} \times 100$). The values in this column vary from 0 to around 20 percent.

We compare the MIP model based-heuristic solutions Z_H with the best solutions of the metaheuristic, Z_{meta} found within 15-min time limit (T_{max}) as well. We calculate the difference percentage of Z_H with Z_{meta} shown by D (%), which is equal to $\frac{Z_{meta} - Z_H}{Z_{meta}} \times 100$. Except $k16$, Z_{meta} reached Z_H for 11 critical nodes. Although we set a 15-min time limit for the metaheuristic, the best solutions are found within the first minutes in an average of 130 seconds and their corresponding D (%) are zero in all but one case. Similar to the previous table, we report the gap (%) between the objective values of the MIP model based-heuristic and CMLB in the last column.

The mathematical model performs poorly with a gap of around 90% on the average when we increase the number of critical nodes to 15, which makes us unable to compare its solutions with the solutions of the MIP model based-heuristic. Hence, we compare Z_H , Z_{meta} and Z_{CMLB} . We calculate the difference D (%) and gap (%) exactly in the way we did when 11 critical nodes exist. For 15 critical nodes, as we see the computational time for the MIP model based-heuristic (T_H) is still very small, namely around 1s. Additionally, the average D (%) between the two methods (MIP model based-heuristic and metaheuristic) is reasonably small and the average gap between the lower bound and the upper bound provided by the MIP model based-heuristic is less than 5%. The results for the metaheuristic show that this method almost reached Z_H in an average of 180 seconds which testifies that the proposed

Table 4.3: Results of Kartal data sets for the mathematical model, CMLB and the MIP model based-heuristic

No.CN	3			4				7			
LN	$T_E(s)$	$T_H(s)$	$T_{CMLB}(s)$	$T_E(s)$	$T_H(s)$	$T_{CMLB}(s)$	gap (%)	$T_E(s)$	$T_H(s)$	$T_{CMLB}(s)$	gap (%)
k1	2.6	0.04	0.21	42.8	0.06	0.34	2.74	589.4	0.21	1.7	6.4
k2	2.3	0.04	0.19	25.8	0.06	0.42	2.74	511.4	0.20	1.8	4.4
k3	2	0.04	0.14	10.6	0.06	0.31	2.7	757	0.21	1.8	6.3
k4	2	0.04	0.22	6.4	0.06	0.4	2.74	598.6	0.21	1.9	4.4
k5	2.1	0.04	0.27	6.7	0.06	0.31	2.74	473.4	0.20	1.7	4.4
k6	3.3	0.05	0.27	13.7	0.06	0.36	0	3101.9	0.17	1.7	0.7
k7	3.2	0.04	0.18	12.9	0.06	0.33	0	1792	0.17	1.9	3.2
k8	19.1	0.05	0.21	58	0.06	0.37	0	1834	0.17	1.9	2.5
k9	2.4	0.05	0.24	126.3	0.06	0.33	0	1732.5	0.17	1.9	0
k10	2.9	0.05	0.23	13.2	0.06	0.4	0	2985.1	0.17	1.7	0
k11	3.6	0.05	0.28	34.4	0.06	0.37	2.74	2990.6	0.17	1.9	4.2
k12	3.8	0.08	0.17	118.7	0.07	0.38	0	4032.4	0.21	1.8	4.2
k13	2.3	0.05	0.25	64.4	0.07	0.47	0	9236.8	0.22	1.8	4.1
k14	5.3	0.04	0.28	17.6	0.07	0.38	4.88	2261.6	0.17	1.8	0
k15	4.7	0.04	0.31	62.7	0.10	0.36	0	3251.8	0.2	1.7	5.3
k16	5.4	0.05	0.19	123.8	0.09	0.38	0	10800	0.22	1.9	1.4
k17	5.4	0.05	0.29	137.8	0.07	0.33	5.33	8211.2	0.21	1.7	6.2
k18	5.7	0.05	0.19	160.7	0.07	0.37	0	10800	0.2	1.8	0.3
k19	5	0.05	0.22	176.4	0.08	0.36	1.25	9657.8	0.24	1.8	4.8
k20	3.5	0.04	0.25	121	0.08	0.4	1.07	10587.3	0.20	2.3	5.7
Average	4.3	0.05	0.23	66.7	0.07	0.37	1.45	4310.2	0.20	1.8	3.4

metaheuristic also can find near-optimal solutions within a reasonable time limit for this number of critical nodes and this network size. In fact, if we examine the convergence of the metaheuristic according to T_{best} , we see that the solutions converge in their first minutes in most of the instances.

4.3 Istanbul network results

With both simplified and southwestern Istanbul networks, we tested 10 instances, each with 15, 20, 25 and 30 critical nodes, making a total of 80 instances. Compared to the Kartal data sets, Istanbul data sets have a larger number of critical and intermediate nodes, and more blocked edges. As a result, the mathematical model proposed by Berktaş et al. (2016) is generally unable to find a feasible solution within the 3-hour time limit. Hence, we only compare the lower bound coming from CMLB, the MIP model based-heuristic and the metaheuristic for this data set. The results for both simplified and southwestern Istanbul networks are summarized in Tables 4.5 - 4.8. In

Table 4.4: Results of Kartal data sets for CMLB, MIP model based-heuristic and metaheuristic

No.CN	11						15				
I.N	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$I(\%)$	$D(\%)$	gap (%)	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)
k1	0.47	477.5	5.9	1.4	0	3.5	1.1	263.7	28.6	0.3	7.8
k2	0.45	0.0	5.6	2.8	0	3.9	1.1	172.7	26.7	0.5	7.4
k3	0.41	482.2	5.6	0	0	3.9	1.1	242.5	27.3	3.1	6
k4	0.52	0.0	5.8	6.7	0	4.5	1.1	238.6	27.8	0	4.6
k5	0.49	0.0	5.5	6.7	0	4.5	1.1	264.9	27.3	0	4.6
k6	0.50	489.2	6	10.6	0	1.9	1.2	284.1	21.6	1.8	3.2
k7	0.43	0.0	5.9	1.9	0	3.1	1.2	454.9	27.8	1.6	6
k8	0.40	0.0	5.6	0	0	2.1	1.2	0	23.4	0	4.7
k9	0.43	11.8	5.7	0.6	0	0.9	1.2	5.7	26.6	0	5.1
k10	0.41	0.0	5.9	3.4	0	0.3	1.1	0	25.2	0	2.6
k11	0.57	32.1	6	3	0	3.1	1.2	268.1	24.6	1.8	5
k12	0.40	23.9	6.5	3.2	0	2.6	1.1	26.7	25.1	4.9	3.4
k13	0.42	478.5	7.3	5.3	0	1.9	1.2	270.3	27.9	3.9	3.9
k14	0.35	0.0	6.1	20.6	0	0.3	1.2	0	23.1	0.6	5.5
k15	0.49	3.1	6.2	7.5	0	3.7	1.3	0	26.1	5.5	4.2
k16	0.49	516.4	5.9	5.6	0.82	4.8	1.3	8.6	23.6	2.6	4.7
k17	0.47	0.0	6.2	21.8	0	7.9	1.3	588.2	22.9	6.7	5.4
k18	0.59	78.3	6.3	7.3	0	5.3	1.2	334.6	22	1.1	5.9
k19	0.46	0.0	6.4	6.4	0	4.1	1.1	0	22.9	0	2.1
k20	0.53	17.4	5.7	13.1	0	7.1	1.2	235.7	20.3	0	6.7
Average	0.46	130.54	6	6.4	0.04	3.5	1.2	182.9	25	1.7	4.9

each table the instance name is stated in the first column. Similar to Section 4.2, we report T_H , T_{best} , T_{CMLB} , $D(\%)$ and gap (%) in these tables as well.

In column $D(\%)$ we see both negative and positive values. If $D(\%)$ is positive, it means that the MIP model based-heuristic finds a better solution. On the other hand, if $D(\%)$ is negative, the metaheuristic obtains a better solution in comparison to the MIP model based-heuristic. In the last row of the tables, the corresponding average values have been reported. For the average of $D(\%)$, the positive average shows the average of all positive $D(\%)$ values, whereas the negative one shows the average of all negative $D(\%)$ values.

As we notice in Tables 4.5 - 4.6, the average $D(\%)$ increases once the number of critical and intermediate nodes increases indicating that the solutions of the MIP model based-heuristic perform better than those found by the metaheuristic in much less computational time. However, the differences are not significantly high; the metaheuristic works effectively within its computational time. Given the average T_{best}

which shows the time of the last improvement, we notice that a 15-min time limit may as well be set for the metaheuristic since for most of the instances, the average difference D (%) would not change much when we continue to run the metaheuristic up to 30 minutes. It is also worth mentioning that the average gap (%) between the MIP model based-heuristic and CMLB is at most 11.8%.

Tables 4.7 and 4.8 present the computational time of the most complicated network, namely, southwestern Istanbul. Although the number of nodes as well as the number of blocked edges in southwestern Istanbul instances is higher than those in Istanbul ones, the average gaps (%) have not been changed significantly, implying the applicability of the MIP model based-heuristic approach in the post-disaster response stage. Note that since the D (%) values in Tables 4.5 - 4.8 are not substantial, we can use the metaheuristic algorithm in case no commercial solver is accessible.

In order to verify the effectiveness of the neighborhood structures (N_1 to N_k) in the metaheuristic, we depict the improvement percentage of the solutions implemented by VNS when we take the initial solutions from the GRASP construction step in Fig. 4.3 for simplified and southwestern Istanbul data sets. This figure shows that as the number of critical nodes increases the improvement percentage of the initial solutions increases for all time limits (5, 10 and 30 minutes) except for 25 critical nodes in the simplified Istanbul instance. It also indicates that the increase in the time limit does not completely affect the improvement percentage.

4.4 Comparison with the related study in the literature

In this section, we compare the results of our heuristic methods with the ones in Berktaş et al. (2016). We coded Berktaş et al. (2016)'s mathematical model and presented the computational times in Table 4.3. We also coded the MIP model based-heuristic method of Berktaş et al. (2016) and executed the runs on the same workstation. Since their model only works for complete networks, we only present the results of the Kartal data sets. We tested their model with up to 15 critical nodes for the MIP model based-heuristic method. We present their results in Table 4.9. For

Table 4.5: Results for simplified Istanbul data for 15 and 20 critical nodes

No.CN	15					20				
I.N	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)
IS1	0.5	31.6	20.3	0.0	9.1	11.4	49.3	195.7	-2.5	13.9
IS2	0.6	253.3	22.0	-0.3	4.7	3.4	234.2	156.7	0.0	9.9
IS3	0.9	15.7	20.8	-1.4	8.6	2.8	207.8	182.9	-1.3	10.3
IS4	0.9	1686.5	30.8	2.4	8.2	4.1	7.0	164.3	0.0	10.8
IS5	0.7	0.0	22.0	0.0	7.1	2.5	0.0	181.9	-3.1	12.6
IS6	0.7	700.4	25.6	0.8	10.8	2.0	0.0	100.9	0.0	3.2
IS7	0.5	122.3	27.2	0.5	10.7	2.4	81.3	200.2	-0.2	8.2
IS8	0.7	0.0	25.3	3.5	6.5	3.7	553.6	147.3	-1.6	14.9
IS9	0.8	1469.5	24.5	-1.6	11.9	2.4	255.7	128.0	-0.6	11.7
IS10	0.4	13.7	21.5	0.0	1.9	7.9	1641.0	155.2	-2.4	11.1
Average	0.7	429.3	24.0	P: 1.0 N: -1.1	8.0	4.3	303.0	161.3	P: 0.0 N: -1.7	10.7

Table 4.6: Results for simplified Istanbul data for 25 and 30 critical nodes

No.CN	25					30				
I.N	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)
IS1	44.4	674.3	530.3	-5.8	14.4	114.6	326.1	1693.3	4.5	9.4
IS2	23.0	155.7	292.8	3.4	9.2	255.0	437.6	2217.6	-0.2	12.5
IS3	9.2	310.7	336.2	0.5	7.9	122.2	0.0	1883.7	3.8	12.5
IS4	44.8	627.1	530.5	0.4	6.1	303.8	990.1	2498.3	-1.5	3.8
IS5	7.0	1799.7	333.8	0.1	6.9	113.6	1649.9	1580.1	-0.3	18.2
IS6	7.3	766.6	823.0	1.2	8.8	112.0	1637.0	1030.6	9.0	13.3
IS7	37.9	1587.4	626.9	-1.6	15.2	99.9	421.3	2403.0	9.7	9.7
IS8	10.7	1156.1	509.6	0.1	13.1	429.8	18.4	2349.8	3.4	14.7
IS9	47.4	741.2	1204.3	1.0	13.4	98.3	1144.0	2994.8	11.9	14.8
IS10	55.8	570.4	587.3	2.6	4.3	348.0	302.3	2875.4	6.9	9.0
Average	28.8	838.9	577.5	P: 1.2 N: -3.7	9.9	199.7	692.7	2152.7	P: 7.0 N: -0.6	11.8

Table 4.7: Results for southwestern Istanbul data for 15 and 20 critical nodes

No.CN	15					20				
I.N	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)
sw1	1.3	0.0	21.5	-0.8	8.7	3.5	1275.7	123.9	-0.1	10.9
sw2	1.2	858.8	26.2	-0.8	9.7	3.1	19.7	117.9	0.2	7.1
sw3	1.0	122.8	29.1	-0.8	12.5	3.2	1574.9	99.5	-3.8	12.5
sw4	1.2	779.0	37.2	0.5	11.7	2.9	630.2	115.3	0.1	12.7
sw5	1.1	1007.2	27.8	2.9	11.0	3.4	1479.4	118.4	-3.2	11.2
sw6	1.2	0.0	29.5	-0.7	7.0	3.7	1381.0	145.9	0.0	9.9
sw7	1.3	718.2	20.3	-3.5	10.9	4.4	17.0	98.8	-9.0	17.8
sw8	1.1	15.6	25.9	-0.1	8.5	5.5	697.4	174.7	2.0	11.4
sw9	1.2	940.3	26.4	1.0	9.5	3.9	1368.4	79.3	2.0	11.9
sw10	1.1	0.0	14.0	0.0	3.4	3.4	760.1	156.7	0.9	8.6
Average	1.2	444.2	26.3	P: 1.5 N: -1.1	9.3	3.7	920.4	123.0	P: 0.9 N: -4.0	11.4

Table 4.8: Results for southwestern Istanbul data for 25 and 30 critical nodes

No.CN	25					30				
I.N	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)	$T_H(s)$	$T_{best}(s)$	$T_{CMLB}(s)$	$D(\%)$	gap (%)
sw1	35.7	901.2	442.3	3.0	7.8	67.4	508.9	1109.6	0.6	7.5
sw2	43.3	275.5	287.6	7.6	7.9	132.1	844.9	1752.9	0.2	12.0
sw3	10.0	0.0	339.7	4.1	8.3	102.1	916.3	1398.6	2.6	11.7
sw4	44.8	0.0	250.8	3.9	7.2	58.2	1714.3	1048.9	8.4	10.2
sw5	54.1	591.1	543.0	-0.9	12.5	117.3	693.6	722.2	2.7	10.7
sw6	40.1	1377.8	378.3	1.6	9.9	182.6	173.0	1503.7	13.0	11.7
sw7	33.5	6.1	497.1	2.7	10.0	175.3	0.0	1871.7	2.7	11.0
sw8	7.3	16.3	394.4	0.7	6.6	212.3	1745.3	969.5	7.4	9.6
sw9	7.0	136.3	444.8	1.4	11.3	87.7	913.2	771.5	6.7	16.6
sw10	34.0	542.0	415.0	0.0	8.2	196.7	400.6	1205.6	8.6	4.4
Average	31.0	384.6	399.3	P: 3.1 N: -0.5	9.0	133.2	791.0	1235.4	5.3	10.5

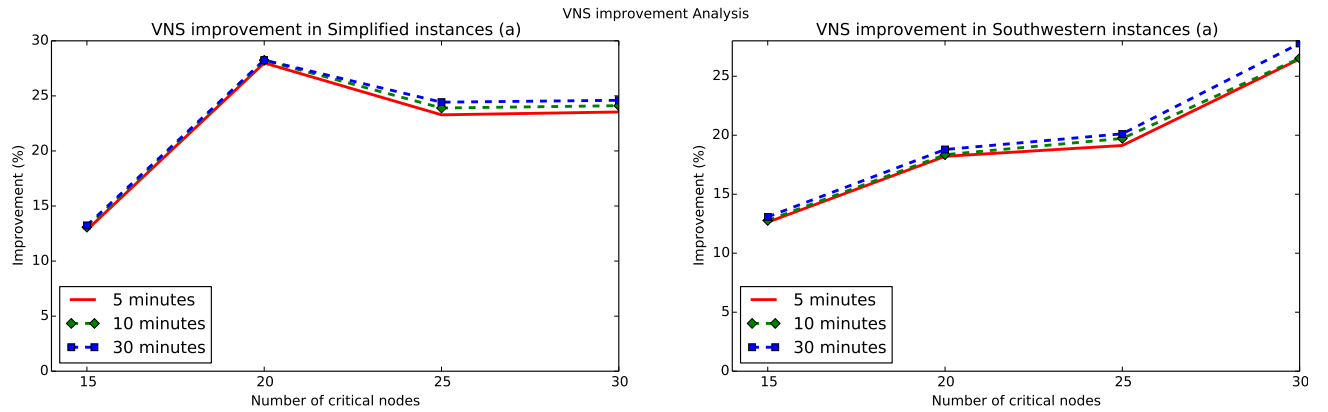


Figure 4.3: VNS analysis for simplified and southwestern Istanbul

Table 4.9: Heuristic results of Berktaş et al. (2016) for Kartal data sets

No.CN	11		15 (within 1-hour time limit)		
I.N	T_H (s)	$T_{\text{Berktaş et al}}$ (s)	T_{total} (s)	gap (%)	Berktaş et al gap (%)
K1	0.5	345.3	29.7	7.8	64.2
k2	0.4	419.7	27.9	7.4	52.0
k3	0.4	254.0	28.3	6.0	67.8
k4	0.5	321.0	28.9	4.6	63.5
k5	0.5	319.3	28.4	4.6	63.4
k6	0.5	380.8	22.8	3.2	64.5
k7	0.4	1213.7	29.1	6.0	67.9
k8	0.4	683.9	24.6	4.7	49.5
k9	0.4	605.1	27.8	5.1	64.9
k10	0.4	477.3	26.4	2.6	64.0
k11	0.6	604.4	25.8	5.0	68.6
k12	0.4	677.9	26.2	3.4	67.3
k13	0.4	651.5	29.1	3.9	65.8
k14	0.3	413.5	24.3	5.5	54.1
k15	0.5	831.4	27.4	4.2	63.7
k16	0.5	1236.4	24.9	4.7	72.0
k17	0.5	573.9	24.2	5.4	67.6
k18	0.6	660.9	23.2	5.9	60.1
k19	0.5	508.2	24.0	2.1	67.3
k20	0.5	536.0	21.5	6.7	58.8
Average	0.5	585.7	26.2	4.9	63.4

11 critical nodes, both methods reached the optimal solution in all instances, with their corresponding computational times shown as $T_{\text{Berktaş et al}}$ and T_H . (T_H shows the computational time of our MIP model based-heuristic algorithm as described before.) For 11 critical nodes, the average computational time of Berktaş et al. (2016)'s method is 585.71 seconds, while we found the same solutions in 0.5 seconds on the average with our MIP model based-heuristic. When we increased the number of critical nodes to 15, we noticed that the computational time of Berktaş et al. (2016)'s method increases tremendously. Therefore, we set a 1-hour time limit to run their model for 15 critical nodes and found out that none of the instances were solved within this time limit. We note that the optimality gap of the MIP model in their heuristic (within the given time limit) provided by the Gurobi solver is 63.4% on the average, as seen in the last column of Table 4.9. In contrast, the MIP in our method is solved exactly in an average time ($T_{\text{total}} = T_H + T_{\text{CMLB}}$) of 26.2 seconds. Note that Berktaş et al. (2016) have not considered an objective correction procedure.

Table 4.10: Results of Simplified Istanbul data for 25 critical nodes

	Perturbation(%)					
I.N	5	10	15	20	40	50
IS1	261.8	259.2	263.7	259.2	276.4	282.5
IS2	155.5	155.5	155.5	155.5	178.2	155.5
IS3	132.7	145.4	145.4	132.7	158.5	141.5
IS4	212.4	192.9	192.9	192.9	252.6	252.6
IS5	209.9	211.3	212.8	209.9	209.9	209.9
IS6	277	277	278.9	277.0	277	280.6
IS7	302.5	300.7	300.7	300.7	312.5	345
IS8	235.7	233.9	234.2	234.2	242.3	245.6
IS9	180.9	165.9	165.2	167.0	165.2	167
IS10	181.1	176.3	176.3	176.3	176.3	181.1

4.5 Sensitivity analysis

In this section, we test our metaheuristic of ML-RCP by changing some important parameters. One such parameter in the metaheuristic is perturbation percentage as explained in Section 3.1.2.2. We tested a set of values for this parameter and present how the best objective value found by the algorithm is affected in Table 4.10. According to the results, once we put the perturbation rate equal to 20 percent, the metaheuristic finds better solutions in most of the instances. We also noticed that solutions get worse when the perturbation rate is more than 20 percent. Therefore, in our computational tests we set this parameter to 20 %.

Second, we changed the parameter α in metaheuristic which shows the greediness of the GRASP construction step. Table 4.11 depicts the range of parameter α . We see that the solutions get worse when α is getting more than 30.

Third, we changed the order of the neighborhood structures to see the behavior of the metaheuristic. There are 4 neighborhood structures and as a consequence there are 24 combinations of the orders. Among which, we report 4 of the orders shown in Table 4.12. We see that changing the orders did not cause significant changes. In our experiments we use 1-2-3-4.

Table 4.11: Best objective value found by metaheuristic in Simplified Istanbul data for 25 critical nodes

I.N	alpha (%)					
	0	10	20	30	50	100
IS1	259.2	259.2	259.2	262.7	275.3	281.5
IS2	155.5	161.2	161.2	165.4	178.4	188.4
IS3	132.7	134.6	134.6	132.7	141.6	149.2
IS4	194.5	192.9	192.9	192	200.6	232.1
IS5	211.2	221	209.9	209.9	231.5	255.7
IS6	277	275.2	281.3	275.2	289.3	300.7
IS7	300.7	300.7	300.7	300.7	312.5	354.5
IS8	235.6	234.2	235.9	233.9	244.6	244.6
IS9	177.8	166.5	166.5	166.5	166.5	166.5
IS10	199.3	178.6	176.3	184.5	176.3	199.5

Table 4.12: Best objective value found in Simplified Istanbul data for 25 critical nodes

I.N	Neighborhood structures orders			
	1-2-3-4	1-3-2-4	4-3-2-1	1-4-2-3
IS1	259.2	259.2	259.2	259.2
IS2	155.5	157.8	155.5	155.5
IS3	132.7	145.6	135.8	131.4
IS4	192.9	192.9	192.9	192.9
IS5	209.9	209.9	212.7	209.9
IS6	277.0	277	277.0	288.9
IS7	300.7	300.7	300.7	300.7
IS8	234.2	237.5	234.2	235.6
IS9	167.0	166.9	167.0	172.5
IS10	176.3	176.3	177.8	176.3

4.6 Total and Maximum Latency Tradeoffs

In this section, we investigate how the optimal total latency value changes when a constraint is imposed on maximum latency. We vary the right hand side (RHS) of the constraint systematically to obtain a spectrum of solutions that reveal the tradeoffs in the two criteria. We first find the optimal maximum latency value by adapting the mathematical model introduced in Kasaei and Salman (2016) that gives the total time required to connect all disconnected components. Thus, we modified their formulation to connect all critical nodes and found the maximum latency values for our data sets. We use these values to set the RHS of the below constraint which is added to the model solved by the MIP model based-heuristic.

$$\sum_{i \in V_C \setminus \{0\}} sp_{0,i}x_{i,1} + \sum_{k \in K \setminus \{n\}} \sum_{i \in V_C \setminus \{0\}} \sum_{j \in V_C \setminus \{0\}, j \neq i} sp_{ij}y_{ijk} \leq T_{updated} \quad (4.1)$$

Constraint (4.1) imposes that the route length cannot exceed $T_{updated}$. The outline of the procedure is given in Algorithm 7.

Algorithm 7 Latency analysis for ML-RCP

- 1: Solve the modified Kasaei and Salman (2016)'s formulation and find the objective value, say T_{ml} .
 - 2: Given the new route, add the clearing time every time the vehicle traverses a blocked edge and update T_{ml} , say $T_{updated}$.
 - 3: Add constraint (4.1) to Angel-Bello et al. (2013) formulation.
 - 4: Implement Algorithm 2 with RHS equal to $T_{updated} \pm \delta\%$ to generate different total latency values.
-

When we solved the maximum latency problem and put the objective value T_{ml} as the RHS of the new constraint, the problem got infeasible most of the time. Because in the formulation that minimizes the maximum latency, the unblocking time is calculated directly in the objective function and as a result, if the vehicle traverses a blocked edge more than once, then the clearing time is added only once to the objective value. However, in the MIP model based-heuristic, the objective correction procedure is executed at the end. Thus, prior to running the new formulation, we

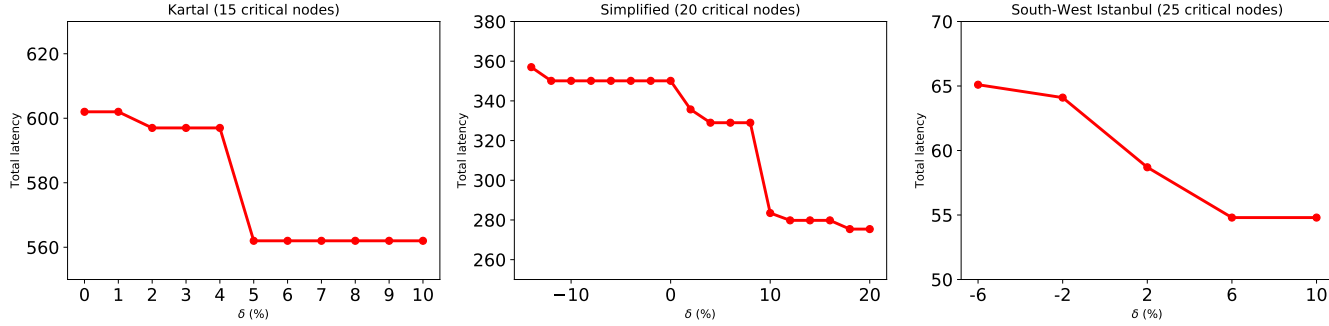


Figure 4.4: Results of ML-RCP under maximum latency constraint

add the clearing time every time the vehicle traverses the blocked edge in Constraint (4.1) and instead of T_{ml} , we consider $T_{updated}$ in the RHS to prevent infeasibility. Finally, we add this constraint to Angel-Bello et al. (2013) formulation and continue the procedure explained in Algorithm 2 to find the new value of total latency. In order to detect the changes in total latency, we add the parameter $\delta\%$ to $T_{updated}$. We pick one instance from each network to show the behavior of total latency when $T_{updated}$ changes. We choose 15, 20 and 25 critical nodes for Kartal, simplified and southwestern data sets, respectively. Fig. 4.4 shows the total latency versus the δ changes in the RHS of Constraint (4.1). We set an interval between zero and 10 for δ , in Kartal results. As we see in this figure, the total latency is getting better until δ increases to 5%, when the same solution found in the ML-RCP in instance k3 in Table 4.4 (for 15 critical nodes) is obtained. Our insight is, since only a few blocked edges are cleared and the graph is complete in Kartal data sets, the routes of the total latency and maximum latency solutions do not vary much after a small δ value.

For simplified Istanbul results, we set an interval between -14 and 20 for δ since for δ less than -14 the problem gets infeasible. As we expect, in Fig. 4.4, the total latency decreases as δ increases up to 18. The sensitivity of this instance is higher than the Kartal instance due to the higher number of clearing procedures required. Moreover, in Istanbul data sets, the graph is incomplete and more critical and intermediate nodes exist. Hence, various solutions are found as δ changes.

Finally, for the southwestern Istanbul data set, the parameter δ should vary be-

tween -6 and 10 to obtain all of the different solutions. This interval is small, considering the number of critical nodes. We can conclude that the close placement of the critical nodes makes the solution follow, usually, the same route for both total and maximum latency problems.

As a result, we suggest that following an approach similar to the analysis here may give more insight to the decision makers, since it is seen whether limiting maximum latency creates a big difference in total latency or not.

In this chapter, we tested various real and random data to guarantee the efficiency and effectiveness of all solution methods. Our main concern is the computational times and the quality of the solutions associated with ML-RCP. Hopefully, based on the results, both of the computational times as well as the quality of the solutions are reasonable, even for larger instances. We believe that these methods can be used by any organization engaged with disaster management, reducing the waiting time of the people and the number of casualties in the critical locations.

Chapter 5

COMPUTATIONAL RESULTS ASSOCIATED WITH MML-RCP

This chapter provides the solutions taken from the mathematical model proposed in Section 3.2.1.

We conduct the computational experiments with Python 2.7.12 using Gurobi 8.0 on Intel Xeon E5-2643 CPU @ 3.30 gigahertz 3.30 gigahertz (two processors) computer with 32 GB RAM, running under the Windows 7 operating system.

We test the mathematical model with two categories of data sets: 1) Kartal data which is based on a real data set, located in Istanbul, Turkey. We have already described its data setting in Section 4.1.1. 2) random data sets in which all parameters are considered randomly to verify the strength of the proposed mathematical model. In Section 5.1.1, we describe the data setting associated with random networks. First, we show the solutions and their run times associated with the Kartal data. Then, we present the solutions of the random networks. In both data sets, we run the model with up to 5 vehicles. We also provide some runs in the cases arising with multiple depots.

The results of the experiments are shown in Tables 5.1-5.7. Tables 5.1 and 5.2 display the results of Kartal data while Tables 5.3-5.7 represent the result of random data sets. First column of each table indicates the instance name. Objective function value and computational time of each instance are denoted by TL(m) and Time(m), respectively, where m is the number of vehicles pre-positioned in the depot. All run times are reported in seconds throughout the dissertation. The last row of each table reports the average computational time of the corresponding instances. We set a 1-hour time limit for the mathematical model for all instances including Kartal and

random data sets. Since all instances are solved optimally within this time limit, the gap is zero and thus, we do not provide a column associated with the gaps.

5.1 Data sets

5.1.1 Random data

In order to ensure the strength of the proposed mathematical model, we run the model with random networks where in each instance, the locations of nodes, the traversal and the clearing times, and the number of blocked edges differ. Similar to the Kartal data sets, we test the model with 7, 11 and 15 critical nodes. In addition, we include different total number of nodes in these networks (20, 30, 40 and 50). The location of all nodes are selected randomly with equal probability. Then, the traversal time between each pair of nodes are calculated, when all nodes are located. The clearing times are calculated according to: $u_{ij} = X.t_{ij}$, $\forall (i, j) \in B$, where X has a uniform distribution between 1 and 20. Moreover, the percentage of the blocked edges vary from 5 to 80. We uploaded these random data sets to the website <https://figshare.com/s/19531120f12b5a0f835f>, which consist of the set of nodes and critical nodes, the set of blocked edges with their clearing times, and the traversal times for each instance.

5.2 Kartal data results

In this section, we report the results of the Kartal data in Table 5.1, where we implement the mathematical model with 7, 11 and 15 critical nodes, each with 20 instances. Moreover, we conduct the experiments with a different number of vehicles. For 7, 11 and 15 critical nodes (No.CN) we employ up to 3, 4 and 5 vehicles to observe the strength of the mathematical model. Recall that in the proposed mathematical model, we have not defined an index associated with the vehicles. Therefore, increasing the number of vehicles decreases the total number of critical nodes designated to each vehicle. As a result, as the number of vehicles increases the average computational time

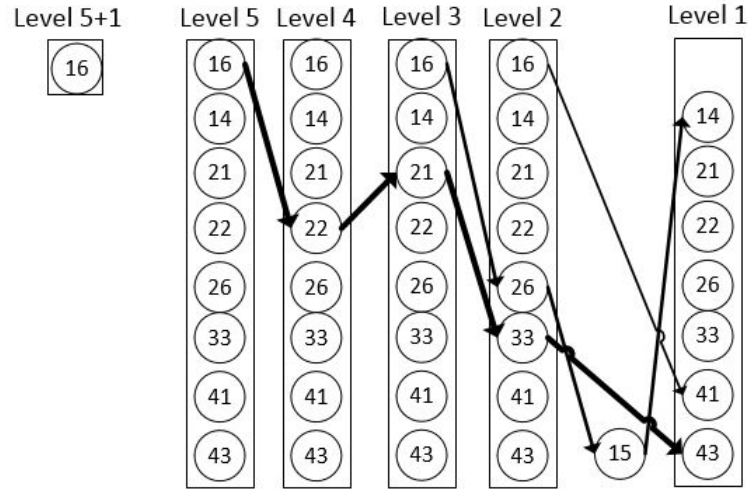


Figure 5.1: Graphical representation of instance k1 with three vehicles

of the mathematical model decreases in most of the instances. We can take advantage of this property if there are more vehicles available. In other words, having more vehicles not only decreases the total latency, but also decreases the computational times as well in most of the instances.

As we see, our proposed mathematical model is able to solve the problem optimally within a reasonably short computational time. For example, the average computational time for 15 critical nodes (with 2 vehicles) is 491.1 seconds (less than 10 minutes), justifying the applicability of our mathematical model in this real data located in Istanbul.

Fig. 5.1 depicts the graphical representation of instance k1 in which 3 vehicles are pre-positioned in the depot node 16. The first vehicle, departs from the depot node 16 located in level 5 and then visits critical nodes 22, 21, 33 and 43. The second vehicle, starts its route from the depot node 16 located in level 3 and then visits critical nodes 26 and 14. Note that at that point vehicle 2 has already visited intermediate node 15 (which is a non-critical node). Finally, the third vehicle, departs from the depot node 16 located in level 2 and directly visits critical node 41. Thus, all critical nodes are visited by these three vehicles.

5.3 Kartal data results with multiple depots

Recall that all of the experiments conducted were based on a single depot. We have already described in Section 3.2.1.1 how to tackle the problems in which the vehicles are pre-positioned in more than one depot node. In this section, we conduct the experiments with assuming multiple depots. Table 5.2 represents the results of the Kartal data with 15 critical nodes, where other than depot node 16, node 45 and 20 are added to the depot set (D), indicating that the vehicles can depart their routes from different (three) depot nodes. We assume that 2, 1 and 1 vehicles are pre-positioned in the depot nodes 16, 45 and 20, respectively where 4 vehicles are available in total. In addition, when 5 vehicles are available, we assign 2, 2 and 1 vehicles to the depot nodes 16, 45 and 20, respectively. Table 5.2 is divided in two parts, namely single depot versus multiple depots. The computational times in the multi-depot instances are slightly higher than those in the single-depot instances. When the number of depot nodes increases, the number of critical nodes increases as well (since we add all the depot nodes to the set V_C .) Note that the locations of the depot nodes may affect the the objective value and the computational times. For instance, the vicinity of the depot nodes to critical nodes may decrease the total latency value and its computational time. However, when the depot nodes are far from the critical nodes, the solution may yield in higher time. As we see in Table 5.2, although the average computational times of instances with multiple depots are higher than the computational times of instances with single depot, we notice a decrease in the objective value of all instances, indicating that using multiple depots besides multiple vehicles may yield better solutions. Fig. 5.2 demonstrates the comparisons of both single and multiple depots associated with Table 5.2, where in all instances the total latency values associated with multiple depots are less than the ones with the single depot. However, in terms of computational times, the trends are not the same as the trends of the total latency values. That is, in some instances with multiple depots, the computational time are higher than those with single depot and vice versa. The locations of the depots, their vicinity to the critical locations, the number of vehicles

pre-positioned in each depot node are the most important reasons that increase or decrease the computational times of the model with multiple depots.



5.4 A graphical analysis for a Kartal instance

In this section, we present the graphical solutions (on the Kartal map) of instance k1 (with 7 critical nodes) with 2, 3 and 4 vehicles in Figures 5.3, 5.4 and 5.5, respectively. The route of each vehicle is shown with a different type of line: namely normal line, a dashed line, a dashed with dot line and a dashed with two dots line. Critical nodes 14, 21 and 22 are displayed by yellow squares, which are the locations of the hospitals in this region. On the other hand, critical nodes 26, 33, 41 and 43 are denoted by green stars, which are the locations of the schools. As we observe from the optimal solution illustrated in Fig. 5.3, 2 vehicles depart from the depot node to visit all critical nodes, where node 15 is visited as an intermediate node. The routes of the first and the second vehicle are 16-22-21-33-43-41 and 16-26-15-14, respectively. The total latency associated with this solution is 100. When we employ 3 vehicles departing from the depot node in Fig. 5.4, we observe that the previous solution has not changed significantly. In fact, the edge (43,41) has been removed (from the previous route visited by the first vehicle in Fig. 5.3) and instead, the third vehicle directly visits critical node 43. By doing so, the total latency decreases and amounts to 85. In Fig. 5.5, where 4 vehicles are positioned in the depot node, the solution associated with this figure has not changed significantly (similar to the solutions of Fig. 5.4). The edge (33,43) has been removed from the previous solution and the critical node 43 is directly visited by the fourth vehicle. The corresponding total latency of the solution in this figure is 82, indicating that adding one more vehicle does not decrease the total latency significantly. Therefore, for this size of number of critical nodes, using at most 3 vehicles is more advantageous. Thus, for larger number of vehicles, we expect less improvement in total latency. Clearly, in the case of seven vehicles or more, the total latency will not change.

Table 5.2: Results of Kartal data sets for 15 critical nodes and 3 depots

instance	single depot				multiple depots			
	TL(4)	Time(4)	TL(5)	Time(5)	TL(4)	Time(4)	TL(5)	Time(5)
k1	223	149.8	204	95.7	191	219.6	174	236.5
k2	222	249.4	206	85.9	190	204.1	172	128.7
k3	227	105.6	210	127.5	190	159.7	173	190.8
k4	220	76.2	204	59.8	194	91	178	86.6
k5	220	76.2	204	60.3	194	89.4	178	86.9
k6	237	56.8	218	49.9	207	90.5	189	92.4
k7	250	116.2	232	164	223	163.5	197	91.9
k8	243	111.2	225	77.8	202	139.7	188	123.6
k9	249	176.1	224	82.4	217	147.7	199	112.5
k10	244	191.4	224	72.5	203	104.5	183	184.7
k11	269	133.3	246	83.7	251	240.3	223	234.1
k12	285	219.2	249	49.3	255	151.8	222	68.5
k13	265	138.6	239	94.8	235	371.6	211	187.5
k14	261	75.2	248	169.9	225	255.4	202	222.8
k15	244	207.5	224	92.5	203	208.3	185	165.8
k16	516	175.3	494	67.7	359	389.7	287	78
k17	443	222.5	413	152.3	350	226.2	310	85.2
k18	478	144	438	182.8	377	374.7	316	273.8
k19	371	130.3	331	49.5	347	158.5	301	139.3
k20	407	84.5	349	46.5	378	86.8	322	78.8
Average		142		93.2		193.7		143.4

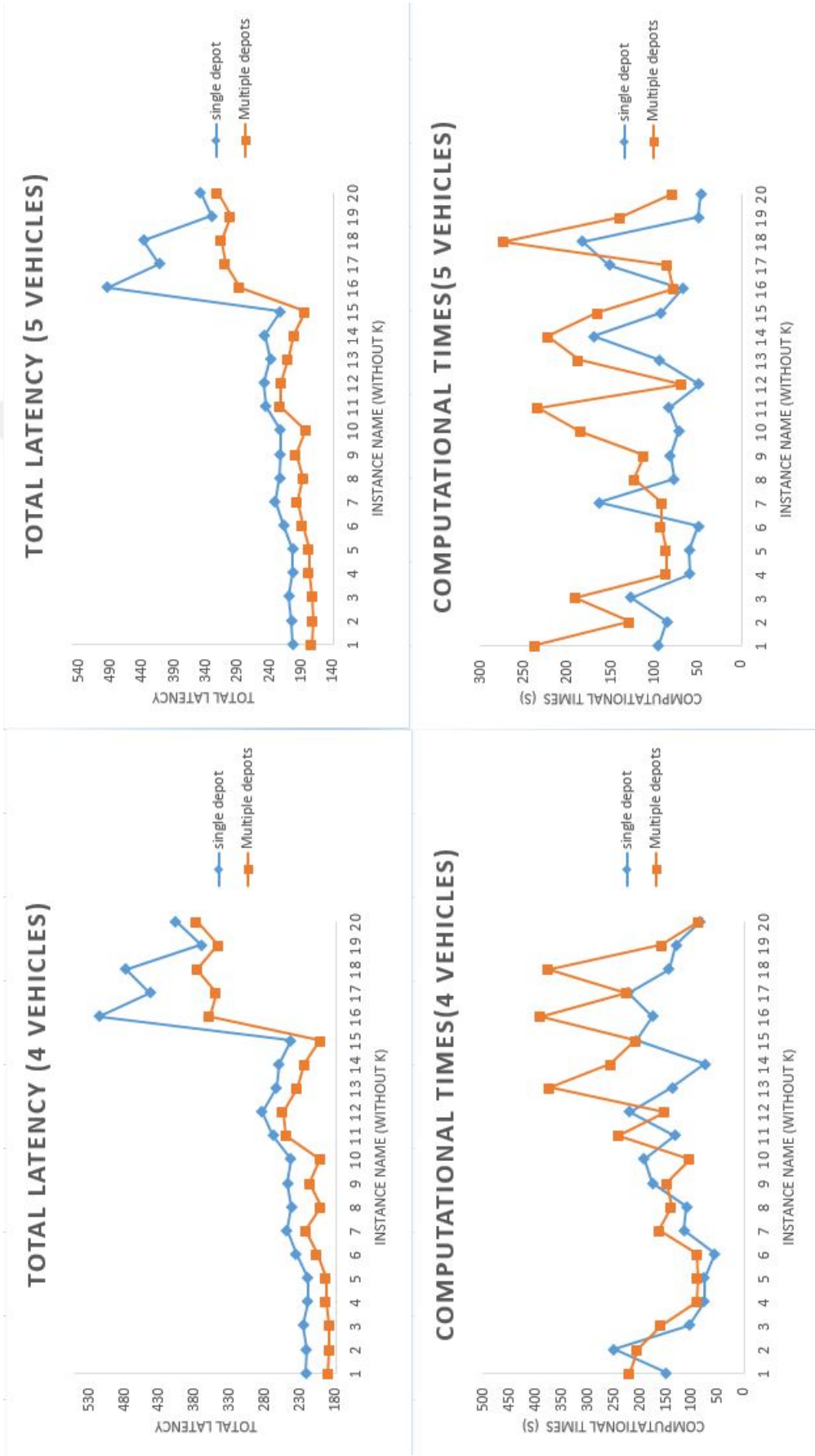


Figure 5.2: Graphical analysis associated with Table 5.2

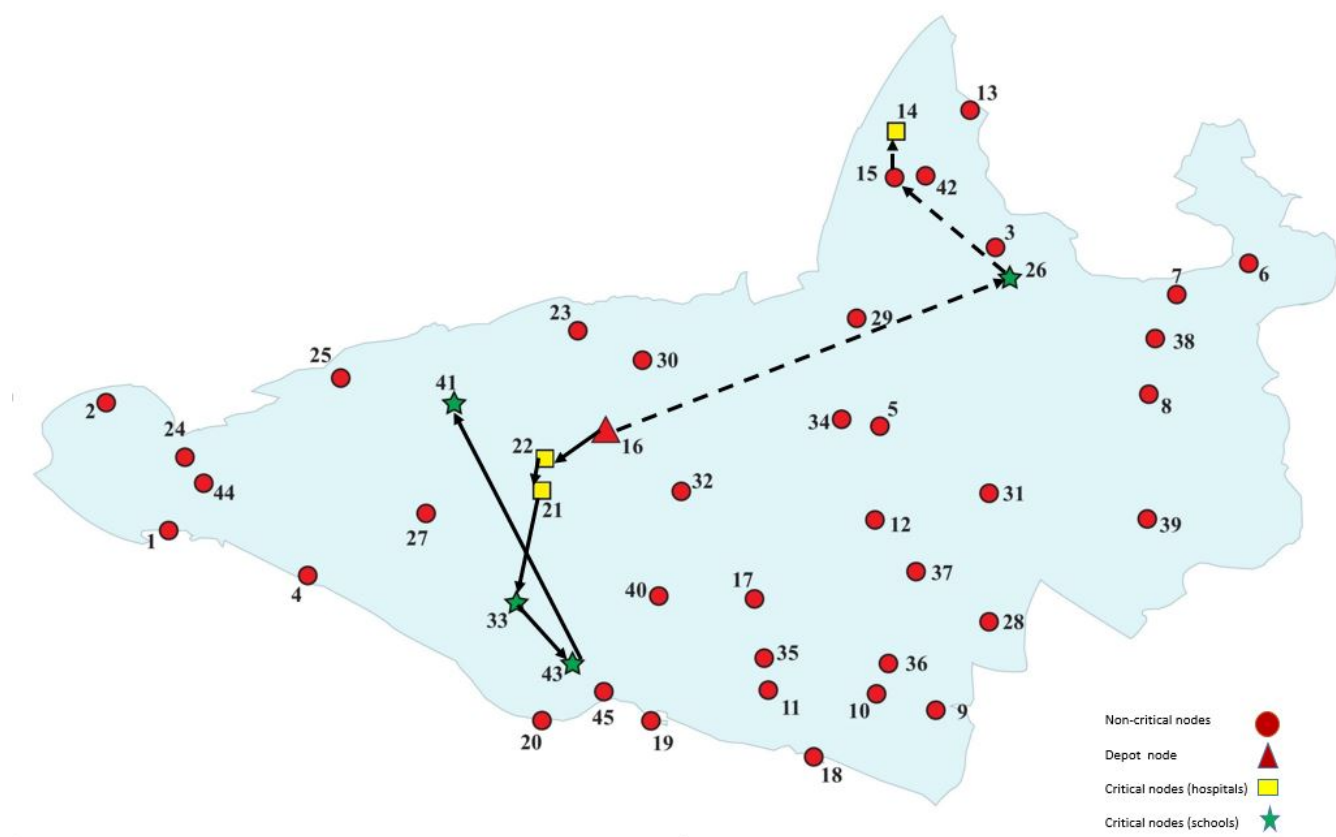


Figure 5.3: Route of instance k1 with 2 vehicles



Figure 5.4: Route of instance k1 with 3 vehicles

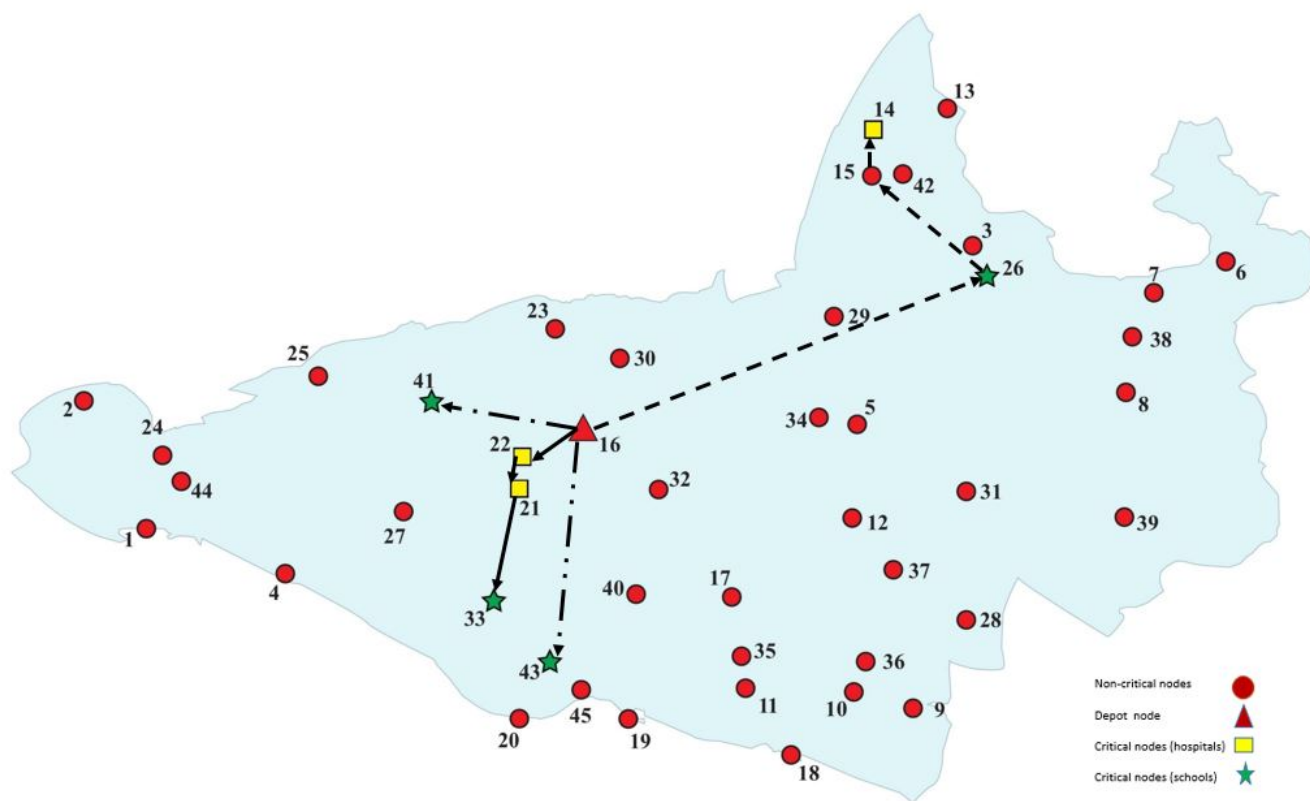


Figure 5.5: Route of instance k1 with 4 vehicles

5.5 Random data results

To ensure the performance of the mathematical model, we carry out experiments with random data sets where the locations and number of the depot(s), critical locations, blocked edges differ in each instance. In addition, traversal and clearing times change instance by instance. We set the number of critical nodes similar to the Kartal data. Nevertheless, we vary the number of total nodes, namely 20, 30, 40 and 50. Recall that in the Kartal data, total number of nodes was fixed (45 nodes) in all instances. With this mind, the numbers in the top row of the tables associated with random data (5.3-5.7) denote the total number of nodes which is shown by No.N. In these data sets, although we use various values for the parameters, the average computational times still maintain reasonable and the mathematical model find the optimal solutions for all random instances within the 1-hour time limit. For example, in Table 5.7, the average computational time (with 2 vehicles) is around 1000 seconds while we set the maximum amount on their parameters, implying the capability of the mathematical model for the data sets generating randomly. Note that in the real life the number of critical nodes will not exceed beyond 15, indicating that this formulation is able to be employed in the problems associated with real life data sets.

In conclusion, by taking advantage of the proposed novel multi-layer network, we developed a mathematical model which is able to solve MML-RCP optimally within a sufficiently short running time limit.

Table 5.3: Results of random data sets for 7 critical nodes with a total number of nodes of 20, 30, 40 and 50

No.N Instance	20				30				40				50			
	TL(2)	Time(2)	TL(3)	Time(3)	TL(2)	Time(2)	TL(3)	Time(3)	TL(2)	Time(2)	TL(3)	Time(3)	TL(2)	Time(2)	TL(3)	Time(3)
r1	407.5	0.3	384.1	0.2	399.7	0.7	352.2	0.5	350.8	1.2	303.0	0.8	240.9	2.2	208.5	1.4
r2	459.4	0.3	420.8	0.2	310.1	0.7	245.4	0.5	441.5	2.1	379.5	0.8	423.2	2.2	389.1	1.4
r3	571.4	0.3	500.4	0.2	388.4	0.7	306.1	0.5	476.8	1.2	351.7	0.8	433.2	2.1	364.6	1.3
r4	265.7	0.3	243.5	0.2	267.7	0.7	254.9	0.5	301.5	3.1	253.9	0.8	360.0	2.5	314.4	1.3
r5	415.4	0.2	353.7	0.2	415.1	0.7	381.1	0.5	309.0	1.1	269.2	0.8	194.4	2.0	176.2	1.3
r6	337.7	0.3	261.7	0.2	426.8	0.7	371.3	0.5	378.4	1.2	299.8	0.8	389.3	2.2	348.6	1.4
r7	320.0	0.3	253.5	0.3	379.9	0.7	296.4	0.5	340.0	1.3	298.2	0.8	328.6	2.2	268.9	1.4
r8	318.7	0.3	251.4	0.2	530.3	0.7	448.9	0.5	218.9	1.2	181.6	0.8	294.8	2.3	259.2	1.3
r9	613.3	0.3	535.9	0.2	281.0	0.7	278.5	0.5	397.3	1.2	356.2	0.8	518.4	2.3	389.7	1.4
r10	214.0	0.3	170.4	0.2	207.9	1.1	158.1	0.5	298.8	1.3	273.0	0.8	294.3	2.2	271.4	1.4
r11	216.7	0.3	198.0	0.2	498.8	0.7	463.5	0.6	319.0	1.2	267.9	0.8	350.4	2.4	325.3	1.5
r12	374.7	0.3	343.9	0.2	459.3	0.7	396.9	0.5	427.4	1.7	372.8	1.4	357.0	4.0	309.2	1.5
r13	311.0	0.3	230.6	0.2	323.3	0.7	276.1	0.5	283.9	1.3	239.5	0.8	303.3	2.2	264.7	1.4
r14	204.0	0.3	185.3	0.2	279.1	1.0	203.0	0.5	276.2	1.2	261.0	0.8	307.7	2.1	270.0	1.4
r15	395.4	0.3	374.5	0.2	330.6	0.7	296.8	0.5	425.0	2.4	308.6	0.8	396.3	2.2	366.0	1.4
r16	312.7	0.2	280.8	0.2	243.6	0.7	229.2	0.5	323.8	1.2	302.9	0.8	233.8	2.2	208.9	1.4
r17	292.7	0.2	292.6	0.2	382.9	0.7	336.7	0.5	231.8	1.2	231.8	0.8	285.9	2.2	271.4	1.4
r18	287.9	0.2	259.0	0.2	438.6	1.3	346.7	0.5	371.4	1.3	327.6	0.8	324.3	2.2	266.5	1.4
r19	368.4	0.2	350.1	0.2	343.4	0.8	299.3	0.5	239.3	1.3	218.6	0.8	316.8	2.5	300.0	1.5
r20	286.6	0.2	230.3	0.2	193.1	0.8	173.4	0.5	257.6	1.2	210.8	0.8	290.4	2.2	246.1	1.3
Average		0.3		0.2		0.8		0.5		1.4		0.8		2.3		1.4

Table 5.4: Results of random data sets for 11 critical nodes with total number of 20 and 30 nodes

No.N	20						30					
Instance	TL(2)	Time(2)	TL(3)	Time(3)	TL(4)	Time(4)	TL(2)	Time(2)	TL(3)	Time(3)	TL(4)	Time(4)
r1	840.0	6.8	597.1	1.7	568.2	1.1	898.9	15.9	652.4	8.2	539.9	5.9
r2	970.9	3.5	814.3	1.8	757.4	2.0	901.7	12.1	735.4	7.4	686.7	6.2
r3	1353.4	3.5	1028.6	1.9	924.4	1.4	33.6	33.6	1152.7	23.8	958.5	8.3
r4	659.2	2.4	540.4	2.8	484.9	1.3	725.8	25.2	541.3	8.8	479.3	6.4
r5	912.5	4.9	732.2	2.2	679.6	1.5	1102.7	8.6	840.5	12.9	703.1	8.7
r6	957.8	3.6	873.6	3.0	796.2	3.1	1169.7	34.0	898.6	10.8	828.4	10.4
r7	751.9	5.5	583.5	3.6	463.3	1.2	811.6	9.3	660.3	8.0	600.8	8.5
r8	863.0	4.8	646.7	1.8	572.3	1.2	1088.8	17.7	897.5	12.0	824.6	9.0
r9	1218.9	4.7	955.8	2.0	843.2	1.4	987.2	16.3	832.9	14.0	745.9	8.2
r10	654.9	3.1	573.0	1.6	548.7	1.2	939.9	31.1	752.8	21.5	695.5	17.3
r11	750.9	6.6	548.7	2.2	463.9	1.7	809.4	15.2	622.2	10.7	537.7	7.7
r12	1050.5	6.4	814.0	2.6	696.4	1.1	845.4	11.4	733.0	7.1	713.0	5.8
r13	836.8	29.2	602.4	4.8	545.2	1.7	919.1	16.8	752.0	13.7	640.3	6.1
r14	833.0	2.1	636.6	1.7	561.4	1.1	868.2	8.6	702.6	7.6	655.1	6.1
r15	925.8	4.4	761.5	1.7	713.2	1.1	1013.4	19.2	820.2	9.8	726.1	6.4
r16	658.2	2.2	520.7	3.5	453.4	1.2	1103.6	25.1	905.2	8.3	850.1	8.9
r17	1096.0	6.8	872.1	2.7	797.4	1.9	761.7	25.4	657.4	25.6	620.7	10.1
r18	928.3	4.3	820.3	2.4	788.9	1.3	762.6	39.7	574.0	14.1	531.7	12.3
r19	1115.5	7.8	931.3	3.8	853.4	1.3	748.5	24.8	560.6	7.2	515.9	5.7
r20	858.0	3.9	679.8	2.5	605.8	1.1	821.6	29.0	577.9	7.2	515.4	5.9
Average		5.8		2.5		1.4		20.9		11.9		8.2

Table 5.5: Results of random data sets for 11 critical nodes with total number of 40 and 50 nodes

No.N	40						50					
Instance	TL(2)	Time(2)	TL(3)	Time(3)	TL(4)	Time(4)	TL(2)	Time(2)	TL(3)	Time(3)	TL(4)	Time(4)
r1	765.8	88.0	618.0	57.7	562.5	11.7	782.8	99.2	587.7	30.7	521.8	24.2
r2	880.8	25.8	692.4	12.8	616.6	10.7	1014.9	131.4	867.8	26.7	829.9	21.2
r3	1248.1	31.5	1017.4	13.3	937.8	16.6	1208.5	136.4	888.0	25.1	793.9	20.5
r4	821.0	33.3	698.0	18.3	648.6	23.6	1017.5	193.4	756.4	28.4	723.8	24.4
r5	791.1	44.1	584.4	13.2	496.2	10.9	1010.3	144.0	746.7	23.5	661.3	24.8
r6	1106.3	14.1	1000.2	13.0	957.0	9.9	1380.4	81.6	1097.1	57.8	953.9	20.1
r7	664.9	63.0	515.2	12.5	464.1	12.5	787.5	134.1	650.3	36.7	615.9	35.6
r8	866.3	21.9	678.2	12.9	607.3	10.4	791.2	148.2	657.2	60.6	581.2	43.0
r9	1027.3	45.7	843.1	21.1	750.7	11.1	819.8	35.2	641.3	23.5	570.5	19.6
r10	689.8	68.8	550.2	44.0	478.8	11.3	931.1	197.3	790.3	42.5	763.3	28.2
r11	819.5	22.9	653.9	13.1	598.4	10.5	1061.4	68.0	890.0	27.8	822.0	28.9
r12	1260.7	69.3	1021.9	37.1	904.7	18.2	1088.3	119.1	949.1	123.6	848.5	20.2
r13	769.4	145.6	562.0	68.2	470.0	10.0	781.5	29.4	669.6	24.0	637.5	19.9
r14	782.6	69.8	636.3	59.4	565.7	20.6	768.1	49.2	710.5	37.0	690.4	26.8
r15	844.5	36.4	701.2	31.0	622.3	11.2	1066.9	100.5	852.5	26.2	807.5	21.0
r16	597.5	49.9	453.9	12.4	411.9	10.3	749.9	33.7	666.2	30.0	623.6	22.2
r17	985.2	54.5	821.9	19.1	761.3	10.9	790.9	62.8	587.7	29.9	537.1	24.5
r18	998.2	94.2	734.2	13.7	650.3	24.8	957.2	194.3	725.5	64.9	617.3	21.0
r19	663.0	38.2	567.2	56.9	494.2	13.0	1157.4	163.9	978.4	35.9	900.7	23.5
r20	513.3	45.3	385.5	21.1	324.7	10.8	1024.3	141.9	861.9	49.8	794.5	27.1
Average		53.1		27.5		13.4		113.2		40.2		24.8

Table 5.7: Results of random data sets for 15 critical nodes with total number of 50 nodes

Instance	TL(2)	Time(2)	TL(3)	Time(3)	TL(4)	Time(4)	TL(5)	Time(5)
r1	1638.1	1500.4	1271.6	456.6	1127.6	144.1	1077.7	123.9
r2	1646.0	1578.4	1197.0	120.7	1060.1	111.4	976.8	87.4
r3	2256.1	859.1	1749.6	236.4	1568.5	301.6	1441.8	141.2
r4	1278.3	654.9	1016.9	354.8	914.0	358.0	847.3	90.7
r5	1384.1	640.0	1055.3	255.8	904.7	194.5	836.0	279.5
r6	2505.3	502.6	2018.8	380.8	1733.8	138.8	1608.6	234.1
r7	1185.5	602.5	1001.6	584.7	906.7	235.0	855.5	131.5
r8	1581.4	813.0	1227.0	577.0	1058.0	271.6	959.2	107.7
r9	1819.3	1012.3	1436.5	662.0	1241.7	440.9	1120.2	104.6
r10	1671.1	571.2	1274.9	370.6	1077.7	116.1	1008.1	97.7
r11	1518.2	1075.6	1157.7	145.5	995.2	104.8	921.5	80.8
r12	1543.5	2873.2	1135.1	555.5	992.0	377.4	895.9	384.7
r13	1321.4	1524.5	1012.6	1158.1	880.3	268.5	824.4	321.5
r14	1521.7	1053.8	1143.4	971.2	968.6	602.5	907.8	191.7
r15	1310.3	889.9	937.0	153.4	820.0	93.1	771.2	80.4
r16	1382.3	540.6	1031.9	136.1	912.8	105.7	869.6	120.0
r17	1542.8	409.5	1206.2	494.8	1077.2	132.7	1005.4	93.9
r18	1398.7	472.6	1199.7	388.2	1107.9	208.4	1048.6	159.3
r19	1559.1	2208.7	1225.4	345.2	1114.2	133.7	1076.7	277.7
r20	1328.3	278.3	1097.9	500.0	986.6	171.4	924.5	223.2
Average		1003.1		442.4		225.5		166.6

5.6 Tradeoff between total and maximum latency in MML-RCP

In this section, we investigate how the optimal total latency value changes when a constraint is imposed on maximum latency. We vary the right hand side (RHS) of the constraint systematically (based on ϵ -constraint method proposed by Bérubé et al. (2009)) to obtain a spectrum of solutions that reveal the tradeoffs in the two criteria. We first find the optimal maximum latency value by adapting the mathematical model introduced in Section 3.2.1. Then, we obtain the maximum latency value which is derived from the optimal solution of total latency value to find the range of RHS given the upper and lower bound values. Note that any value less than the lower bound values makes the MML-RCP infeasible. On the other hand, any value greater than the upper bound value makes the maximum latency constraint redundant.

We pick 2 different instances for this section. The first one is instance k1 from Kartal data sets with 4 vehicles, 15 critical nodes. The second one is randomly

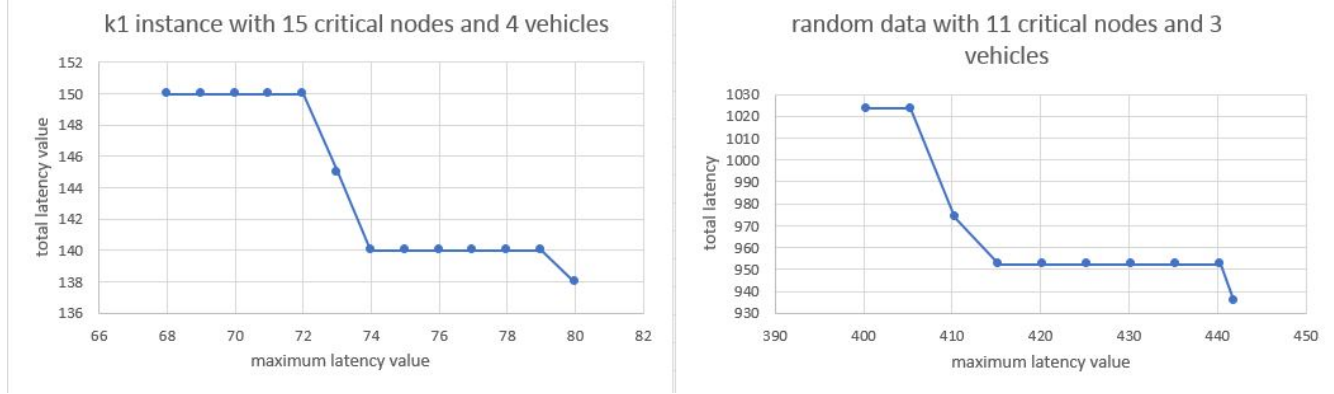


Figure 5.6: Results of MML-RCP under maximum latency constraint

generated similar to Section 5.1.1, with 3 vehicles and 11 critical nodes. Fig. 5.6 shows the tradeoffs between the total latency and maximum latency in k1 and random instance. As we see in k1 instance, the range of maximum latency is not very high, indicating that the solutions of the maximum and total latency are similar to each other. We expect these results for Kartal data since some of the critical nodes (i.e., hospitals) are close to each other. Thus, the solutions of both maximum and total latency are close to each other. On the other hand, in random data, the range of the maximum latency is higher than those in k1 instance. In this random data, the critical nodes are chosen in various locations. Therefore, considering maximum or total latency is important for decision makers.

As a result, we suggest that following an approach similar to the analysis here may give more insight to the decision makers, since it is seen whether limiting maximum latency creates a big difference in total latency or not.

In this chapter, we conducted the experiments for the MML-RCP problem, where multiple vehicles, multiple depots have been employed. Most importantly a subset of nodes (rather than all nodes) have been visited, which generalize and make the problem more realistic. Our results show that the mathematical model based on the multi-level network is able to solve and find the all solutions optimally. Note that we tested our instances up to 15 critical nodes with both real and random data to ensure

the strength of the model. Moreover, in the real world, having beyond 15 critical nodes in a region does not exist. Thus, we can employ this model to accelerate the process of helping the people trapped in the critical locations. It is worth mentioning that since this problem addresses the latency time or the waiting time of the people, using a fast and efficient method is a must in the problems arising in the the context of the disaster management.



Chapter 6

CONCLUSION AND REMARKS

In this thesis, we proposed solution methods for the road clearance problems arising in the disaster response phase to minimize the total latency of reaching the critical locations. There are two studies addressed in this dissertation.

The **first** one is the post-disaster road clearance problem with the aim of providing fast and effective methods to determine the route of a single vehicle (positioned in a depot node) which is assigned for clearing the most beneficial roads. An exact formulation has been already developed for this problem which is able to solve a very limited instances. We developed a MIP model based-heuristic approach based on a solving MIP on a transformed network. In the computational results, we tested the mathematical model developed by Berktaş et al. (2016) and observed that the number and the locations of the critical nodes have a significant effect on the solution computational time. Furthermore, the best lower bound found within a 3-hour time limit is quite weak. Therefore, we proposed a lower bounding method (CMLB), which solves a number of MIP formulations. Then, we suggested a MIP model based-heuristic approach to find high quality feasible solutions within a short time. For both Kartal and Istanbul data sets, the MIP model based-heuristic finds a feasible solution in a few seconds whose latency is better than or equal to that of the solution of the exact formulation found in the 3-hour time limit.

We also developed a metaheuristic based on running a GRASP construction step followed by VNS for improvement step repeatedly. Both heuristic approaches obtain more or less close solutions while the MIP model based-heuristic runs in much less time. We also compared our MIP model based-heuristic with the MIP model based-heuristic in Berktaş et al. (2016) that solves our problem using complete graphs. We

saw that the computational time of their heuristic is much higher than our MIP model based-heuristic's, especially when the number of critical nodes increases. Since these algorithms should be executed in the post-disaster stage, having short computational times is critical and enables the problem to be resolved in real-time as new information is obtained over the response stage. Finally, we provided an analysis to see the effect of limiting maximum latency on the total latency under different network structures. Our findings in this problem has been published in European Journal of Operational Research (Ajam et al. 2019).

In the **second** problem, we introduced a novel multi-level network for the road clearance problem in the context of disaster management with the objective of minimizing total latency of critical nodes until all of them are visited. We were able to incorporate multiple vehicles, multiple depots and most importantly visiting a subset of nodes (instead of visiting all nodes) to our problem. To the best of our knowledge, MML-RCP has not been addressed in the literature. Based on the novel multi-level network, we proposed a mathematical model that is able to solve MML-RCP optimally with reasonable computational times. We tested two categories of data sets: the first one is based on the real data set, located in Istanbul and the second one is based on the random data sets. The reasonable computational times of the mathematical model testify the performance of the proposed model. It is worth mentioning that we have not intentionally defined an index associated with vehicles. By means of this, average computational times decreases when the number of vehicles increases. Recall that having small computational time is extremely important for the problems arising in the aftermath of a disaster. We believe this new mathematical model can be applied to any data sets associated with a region or a city.

To sum up, the contributions of this thesis have been itemized below:

1. We developed a MIP model based-heuristic method which is able to solve larger instances in short times with high quality solutions.
2. We introduced an Objective Correction Procedure to ensure the quality of the

solutions provided by the MIP model based-heuristic.

3. We introduced a novel lower bounding approach testifying the quality of solutions provided by the MIP model based-heuristic.
4. Alternatively, we provided a metaheuristic approach based on the GRASP construction step for the initial solution, following by VNS for the improvement phase. The practitioners that do not have access to commercial solvers can employ this metaheuristic.
5. We conducted an analysis between makespan and total latency problem.
6. We could consider multiple vehicles, multiple depots and most importantly visiting a subset of nodes (instead of visiting all nodes) to our MML-RCP problem.
7. We developed a novel multi-level network for the MML-RCP. Based on that, we developed a fast mathematical model.
8. Computational times in all methods given in this thesis, are reasonably short, which crucially important for the problems engaged with disaster management.

It is worth mentioning that we can exert these methodologies for other applications in addition to disaster problems. For instance, we can use the methodologies for health care problems where the patients are waiting for treatment or transplantation. Or, we can use them for the transportation in metropolises in order to minimize the waiting time of the people to get in the buses or metros.

For future work, considering synchronization between the vehicles (relaxing the disjoint path assumption) would be a valuable extension that may better cope with adverse conditions after a disaster. Considering this assumption is computationally much more challenging since the coordinate of the vehicles is needed, which arises because an edge opened by a vehicle can later be traversed by other vehicles. We may use the solutions provided by MML-RCP as the initial solutions and then by

developing heuristic methods, we try to obtain near-optimal solutions. In addition, we will conduct a bi-objective augmented ϵ -constraint analysis to show the trade-offs between total latency and makespan in MML-RCP. Another significant extension is to consider resilience of the network. Resilience of a communication network is its ability to maintain the same level of functionality in the face of internal changes and external disturbances as a result of large-scale natural disasters and corresponding failures, weather-based disruptions, technology-related disasters, and malicious human activities. There are some criteria to evaluate the resilience of a network in the aftermath of a disaster. The ones which are close to our problem can be considered as the percentage of the functional edges or the number of people having access to critical location over the time. In line with the network resilience, we can develop models associated with weighted latency problems (i.e., we can assign more weights to hospitals). Note that the advantage of the formulation proposed in the second problem (MML-RCP) is its speed in solving the problem. However, it is unable to provide latency of each critical node separately in its model. As a result, assigning weights is not possible to be incorporated in our problem.

BIBLIOGRAPHY

- Ajam, M., Akbari, V., and Salman, F. S. (2019). Minimizing latency in post-Disaster road clearance operations. *European Journal of Operational Research*, 277(3):1098–1112.
- Akbari, V. and Salman, F. S. (2017a). Multi-vehicle prize collecting arc routing for connectivity problem. *Computers and Operations Research*, 82:52–68.
- Akbari, V. and Salman, F. S. (2017b). Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. *European Journal of Operational Research*, 257(2):625–640.
- Aksu, D. T. and Özdamar, L. (2014). A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61:56–67.
- Angel-Bello, F., Alvarez, A., and García, I. (2013). Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, 37(4):2257–2266.
- Angel-Bello, F., Cardona-Valdés, Y., and Álvarez, A. (2017). Mixed integer formulations for the multiple minimum latency problem. *Operational Research*, pages 1–30.
- Arellano-Arriaga, N. A., Molina, J., Schaeffer, S. E., Álvarez-Socarrás, A., and Martínez-Salazar, I. A. (2019). A bi-objective study of the minimum latency problem. *Journal of Heuristics*, 25(3):431–454.
- Bang, B. H. (2018). A GRASP+VND Algorithm for the Multiple Traveling Repairman Problem with Distance Constraints. *Journal of Computer Science and Cybernetics*, 33(3):272–288.
- Berktaş, N., Kara, B. Y., and Karaşan, O. E. (2016). Solution methodologies for debris removal in disaster response. *EURO Journal on Computational Optimization*, 4(3-4):403–445.
- Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). An exact -constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European journal of operational research*, 194(1):39–50.

- Bulhões, T., Sadykov, R., and Uchoa, E. (2018). A branch-and-price algorithm for the Minimum Latency Problem. *Computers and Operations Research*, 93:66–78.
- Çelik, M. (2016). Network restoration and recovery in humanitarian operations: Framework, literature review, and research directions.
- Çelik, M., Ergun, Ö., and Keskinocak, P. (2015). The Post-Disaster Debris Clearance Problem Under Incomplete Information. *Operations Research*, 63(1):65–85.
- Cornuéjols, G., Fonlupt, J., and Naddef, D. (1985). The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical programming*, 33(1):1–27.
- Şahin, H., Kara, B. Y., and Karasan, O. E. (2016). Debris removal during disaster response: A case for Turkey. *Socio-Economic Planning Sciences*, 53:49–59.
- Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C., and Vansteenwegen, P. (2013). Heuristics for the traveling repairman problem with profits. *Computers and Operations Research*, 40(7):1700–1707.
- Duque, P. M., Dolinskaya, I. S., and Sörensen, K. (2016). Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, 248(1):272–285.
- Duque, P. M. and Sörensen, K. (2011). A GRASP metaheuristic to improve accessibility after a disaster. *OR Spectrum*, 33(3):525–542.
- Fleischmann, B. (1985). A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research*, 21(3):307–317.
- Gavish, B. and Graves, S. (1978). The travelling salesman problem and related problems.
- Goemans, M. and Kleinberg, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming, Series B*, 82(1-2):111–124.
- IMM, I. M. M. (2002). The study on a disaster prevention/mitigation basic plan in istanbul including seismic microzonation in the republic of turkey.
- Kasaei, M. and Salman, F. S. (2016). Arc routing problems to restore connectivity of a road network. *Transportation Research Part E: Logistics and Transportation Review*, 95:177–206.
- Kilci, F., Kara, B. Y., and Bozkaya, B. (2015). Locating Temporary Shelter Areas after an Earthquake. *European Journal of Operational Research*, 243(1):1–21.

- Liberatore, F., Ortuño, M. T., Tirado, G., Vitoriano, B., and Scaparra, M. P. (2014). A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. *Computers & Operations Research*, 42:3–13.
- Lorca, Á., Çelik, M., Ergun, Ö., and Keskinocak, P. (2017). An optimization-based decision-support tool for post-disaster debris operations. *Production and Operations Management*, 26(6):1076–1091.
- Luo, Z., Qin, H., and Lim, A. (2014). Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. *European Journal of Operational Research*, 234(1):49–60.
- Méndez-Díaz, I., Zabala, P., and Lucena, A. (2008). A new formulation for the Traveling Deliveryman Problem. *Discrete Applied Mathematics*, 156(17):3223–3237.
- Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.
- Mladenović, N., Urošević, D., and Hanafi, S. (2013). Variable neighborhood search for the travelling deliveryman problem. *4OR*, 11(1):57–73.
- Moreno, A., Munari, P., and Alem, D. (2019). A branch-and-Benders-cut algorithm for the Crew Scheduling and Routing Problem in road restoration. *European Journal of Operational Research*, 275(1):16–34.
- Morshedlou, N., González, A. D., and Barker, K. (2018). Work crew routing problem for infrastructure network restoration. *Transportation Research Part B: Methodological*, 118:66–89.
- Nucamendi, S., Cardona-Valdes, Y., and Angel-Bello Acosta, F. (2015). Minimizing customers’ waiting time in a vehicle routing problem with unit demands. *Journal of Computer and Systems Sciences International*, 54(6):866–881.
- Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., and Moreno-Vega, J. M. (2016). A mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem. *Journal of the Operational Research Society*, 67(8):1121–1134.
- Orloff, C. S. (1974). A fundamental problem in vehicle routing. *Networks*, 4(1):35–64.

- Özdamar, L., Tüzün Aksu, D., and Ergüne, B. (2014). Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, 48(4):249–262.
- Picard, J.-C. and Queyranne, M. (1978). The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling. *Operations Research*, 26(1):86–110.
- Ranghieri, F. and Ishiwatari, M. (2014). *Learning from Megadisasters: Lessons from the Great East Japan Earthquake*. The World Bank.
- Resende, M. G. C. (2001). Greedy randomized adaptive search procedures (GRASP). *Encyclopedia of optimization*, 2(2):373–382.
- Roberti, R. and Mingozzi, A. (2014). Dynamic ng-Path Relaxation for the Delivery Man Problem. *Transportation Science*, 48(3):413–424.
- Sahni, S. and Gonzalez, T. (1976). P-Complete Approximation Problems. *Journal of the ACM*, 23(3):555–565.
- Salehipour, A., Sörensen, K., Goos, P., and Bräysy, O. (2011). Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem. *4OR*, 9(2):189–209.
- Sarubbi, J., Luna, H., and Miranda, G. (2008). Minimum latency problem as a shortest path problem with side constraints. In *XIV latin Ibero-American congress on operations research (CLAIO)*.
- Shin, Y., Kim, S., and Moon, I. (2019). Integrated optimal scheduling of repair crew and relief vehicle after disaster. *Computers & Operations Research*, 105:237–247.
- Silva, M. M., Subramanian, A., Vidal, T., and Satoru, L. (2012). A simple and effective metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research*, 221(3):513–520.
- Vodák, R., Bíl, M., and Kivánková, Z. (2018). A modified ant colony optimization algorithm to increase the speed of the road network recovery process after disasters. *International Journal of Disaster Risk Reduction*, 31:1092–1106.
- Yan, S. and Shih, Y. L. (2007). A time-space network model for work team scheduling after a major disaster. *Journal of the Chinese Institute of Engineers, Transactions of the Chinese Institute of Engineers, Series A/Chung-kuo Kung Ch’eng Hsueh K’an*, 30(1):63–75.

- Yan, S. and Shih, Y. L. (2009). Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers and Operations Research*, 36(6):2049–2065.
- Yan, S. and Shih, Y.-L. (2012). An ant colony system-based hybrid algorithm for an emergency roadway repair time-space network flow problem. *Transportmetrica*, 8(5):361–386.





Appendices

Appendix 1

EXISTING FORMULATION ASSOCIATED WITH MLP

In this chapter, we present the mathematical model provided by Berktaş et al. (2016).

Sets and parameters:

N : set of all nodes

A : set of all edges

C : set of critical nodes

NC : set of non-critical nodes

I_{ij} : 0 if the arc (i, j) is blocked, 0 ,otherwise

t_{ij} : traversing time of arc (i, j)

c_{ij} : clearing time of arc (i, j)

C' : set of duplicated critical nodes

NC' : $NC \cup C'$

Decision variables:

y_{kl} : 1 if $l \in C$ is the first critical node visited after $k \in C$ (possibly with noncritical nodes in between) 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)

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

v_{ij}^k =1 if $(i, j) \in A'$ is cleaned to reach node $k \in C$, and 0 otherwise

r_k = time that node $k \in C$ is reached

$$\min \sum_{k \in C \setminus \{s\}} w_k r_k \quad (\text{A.1})$$

s.t.

$$\sum_{j \in NC' \cup \{l\}} x_{kj}^l = y_{kl} \quad \forall k, l \in C, k \neq l \quad (\text{A.2})$$

$$\sum_{i \in N'} x_{ij}^k - \sum_{h \in NC' \cup \{k\}} x_{jh}^k = 0 \quad \forall k \in C, \forall j \in NC' \quad (\text{A.3})$$

$$\sum_{i \in N'} x_{il}^l = 1 \quad \forall l \in C \setminus \{s\} \quad (\text{A.4})$$

$$C_l = \sum_{i, j \in N'} x_{ij}^l t_{ij} \quad \forall l \in C \setminus \{s\} \quad (\text{A.5})$$

$$y_{kl} + y_{lk} \leq 1 \quad \forall k, l \in C, k \neq l \quad (\text{A.6})$$

$$\sum_{k \in C: k \neq l} y_{kl} = 1 \quad \forall l \in C \setminus \{s\} \quad (\text{A.7})$$

$$\sum_{k \in C, l \in C \setminus \{s\}: k \neq l} y_{kl} = |C \setminus \{s\}| \quad (\text{A.8})$$

$$\sum_{l \in C: l \neq k} y_{kl} \leq 1 \quad \forall k \in C \quad (\text{A.9})$$

$$a_{kl} \geq y_{kl} \quad \forall k, l \in C \quad (\text{A.10})$$

$$a_{kl} + a_{lk} = 1 \quad \forall k, l \in C, k \neq l \quad (\text{A.11})$$

$$a_{ml} \geq a_{mk} + y_{kl} - 1 \quad \forall k, l, m \in C, k \neq l \quad (\text{A.12})$$

$$a_{sl} = 1 \quad \forall l \in C \setminus \{s\} \quad (\text{A.13})$$

$$r_s = 0 \quad (\text{A.14})$$

$$r_l \geq r_k + C_l + \sum_{i, j \in N: i < j} v_{ij}^k c_{ij} - (1 - y_{kl}) M \quad \forall k \in C, \forall l \in C \setminus \{s\} \quad (\text{A.15})$$

$$\sum_{l \in C \setminus \{s\}} v_{ij}^l \leq 1 - I_{ij} \quad \forall i, j \in N: i < j \quad (\text{A.16})$$

$$2 - v_{ij}^l \geq x_{ij}^k + x_{ji}^k + x_{i'j}^k + x_{ji'}^k +$$

$$x_{ji'}^k + x_{ij'}^k + x_{i'j'}^k + x_{ji'}^k + a_{kl} \quad \forall i, j, k, l \in C : i < j, I_{ij} = 0, k \neq l$$

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

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

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

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

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

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

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

$$x_{ij}^k \in (0, 1)$$

$$v_{ij}^k \in (0, 1)$$

$$a_{kl}, y_{kl} \in (0, 1)$$

$$r_k, C_k \geq 0$$

The objective function calculates the weighted total latency. (A.2), (A.3) and (A.4) jointly construct the paths among the critical nodes and they are the reason behind the duplication. If critical node l is visited right after critical node k , (A.2)

ensures that the vehicle leaves k to go to l . The vehicle may go to the node l directly or it can first go to a noncritical node $j \in NC'$. For the latter case, if vehicle visits an intermediate node j to reach a critical node k , (A.3) ensures that the vehicle leaves this node; in other words, it makes the total flow entering j and leaving j equal. In this constraint, the first summation is over all nodes since the vehicle can depart from any node and the second sum is over the all noncritical nodes and the targeted critical node k . Constraint (A.4) guarantees that there is a flow entering each critical node. (A.5) calculates the travel time to go to critical node l from the previous critical node. (A.6) is a simple valid inequality that says either k precedes l , l precedes k or they are not consecutively visited critical nodes. Constraint (A.7) ensures that each critical node except the supply node has a predecessor critical node. (A.8) limits the total number of assignments to the number of critical nodes that we need to reach. (A.9) 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 (A.11) which assures either $k \in C$ is visited before $l \in C$ or vice versa. By (A.10) if $k \in C$ is visited just before $l \in C$ then k is visited before l and by (A.12) 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 (A.13) and (A.14). Constraint (A.15) 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 . With (A.16) it is guaranteed that an arc is cleaned only once and only if it is blocked. Constraints (A.17)-(A.20) prevent a blocked arc from being cleaned in latter usage. For example if a blocked arc (i, j) or one of its artificial versions have been 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 (A.21)-(A.24) ensure that a blocked arc is cleaned while going to a critical node 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 (A.17)-(A.24) together guarantee that when a blocked arc is used, it is cleaned once and debris is removed on its first usage.

