Minimizing Latency in Post-Disaster Road Clearance Operations

Meraj Ajam[a], Vahid Akbari[b], F. Sibel Salman[*a]

[a]*College of Engineering, Koç University, 34450 Sariyer, Istanbul, Turkey*
[b]*Nottingham University Business School, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, United Kingdom*

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. 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 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 matheuristic 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.

Keywords: Humanitarian logistics; road clearance; arc routing; minimum latency; heuristics

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

[*]Corresponding author
Email addresses: majam14@ku.edu.tr (Meraj Ajam), vahid.akbari@nottingham.ac.uk (Vahid Akbari), ssalman@ku.edu.tr (F. Sibel Salman[*])

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.

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 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. 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. The sooner each critical location is reached, the better chances are to prevent loss of lives and suffering.

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 paper, we use road clearance and unblocking interchangeably for the activities that enable access on a blocked edge and may as well include repairs. 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. 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 this arc routing problem is 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, 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

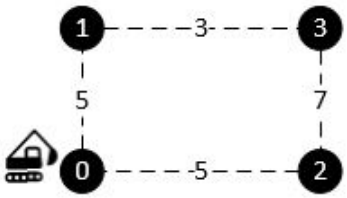

Figure 1: Example with different total latency for the same maximum latency value

provide an analysis where an upper bound can be set on maximum latency while minimizing total latency.

1.1. Our contributions

The problem under consideration 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 matheuristic 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 matheuristic 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 matheuristic and the metaheuristic have significantly smaller computational times in comparison to the matheuristic 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.

The remainder of this article is organized as follows. A review of related studies is given in Section 2; the problem description is presented in Section 3, and Section 4 provides the lower bounding approach. Section 5 addresses the two solution methods. In Sections 6 and 7, we describe the data sets and present the computational results, respectively. We demonstrate the trade-off between total and maximum latency in Section 8. Finally, Section 9 presents the concluding remarks.

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. 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.

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 matheuristic 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.

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. 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. 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%.

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. Nucamendi-Guillén et al. (2016) studied MLP with multiple vehicles, citing applications in humanitarian aid distribution and personnel transportation. They presented an MIP that outperforms the branch-and-price-and-cut algorithm in the literature (Luo et al. (2014)) in terms of computational time. They also proposed a metaheuristic to solve the problem for larger instances, which reached to optimality in 386 out of 389 instances in a short time. Recently, Angel-Bello et al. (2017) introduced five mathematical models for the multi-vehicle MLP. The first three models are obtained from the classical and flow-based formulations while the last two ones are generalizations of the time-dependent MLP formulation. They tested the models with data having up to 80 nodes and 16 vehicles, where the largest instance is solved in 86 seconds. The authors showed that the last two models perform better in terms of computational time.

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, 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.

3. Problem description

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

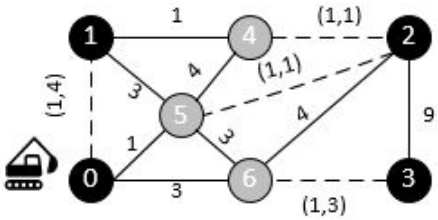

Figure 2: Example of ML-RCP

refer to this problem as the Minimum Total Latency in Road Clearance Problem (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. 2 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.

4. 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 Section 7, 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 single 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 $1, 2, \ldots, n$

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

$$\min \sum_{i \in V_C} \sum_{j \in V_C, j \neq i} sp_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_{i \in V_C \setminus \{0\}} x_{0,i} = 1 \tag{2}$$

$$\sum_{i \in V_C, i \neq D} x_{iD} = 1 \tag{3}$$

$$x_{Di} = 0 \quad \forall i \in V_C \setminus \{0\}, i \neq D \tag{4}$$

$$\sum_{i \in V_C, i \neq j} x_{ij} - \sum_{k \in V_C \setminus \{0\}, k \neq j} x_{jk} = 0 \quad \forall j \in V_C \setminus \{0\}, j \neq D \tag{5}$$

$$x_{ij} + x_{ji} \leq 1 \quad \forall (i, j) \in E_C \tag{6}$$

$$\sum_{j \in V_C, j \neq i} x_{ji} = y_i \quad \forall i \in V_C \setminus \{0\} \tag{7}$$

$$\sum_{i \in V_C \setminus \{0\}} f_{0,i} = ICN + 1 \tag{8}$$

$$\sum_{i \in V_C, i \neq D} f_{iD} = 1 \tag{9}$$

$$\sum_{i \in V_C, i \neq j} f_{ij} - \sum_{k \in V_C \setminus \{0\}, k \neq j} f_{jk} = y_j \quad \forall j \in V_C \setminus \{0\}, j \neq D \tag{10}$$

$$f_{ij} \leq (ICN + 1) x_{ij} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \tag{11}$$

$$f_{ij} \geq x_{ij} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \tag{12}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V_C, \forall j \in V_C, i \neq j \tag{13}$$

$$y_i \in \{0, 1\} \quad \forall i \in V_C \setminus \{0\} \tag{14}$$

$$f_{ij} \geq 0 \quad \forall i \in V_C, \forall j \in V_C, i \neq j \tag{15}$$

Objective function (1) is to minimize the total time of the path from 0 to D. Constraint (2) guarantees that the vehicle leaves the depot only once by visiting one of the critical nodes. Constraint (3) ensures that the destination node must be visited once at the end of the route. By Constraints (4) once the destination node is visited, the route ends. Constraints (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 (6). Constraints (7) define if node i is visited or not. Constraints (8-10) are flow balance equations, where a unit flow is left at each intermediate critical node. In Constraint (8), the net flow out of the depot node should be equal to the total number of visits to all other nodes. Constraint (9) forces the model to send a unit flow to the destination node. Constraints (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 (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 (12) ensure that if edge $\{i, j\}$ is used, a positive flow should be assigned to that edge. Constraints (13-15) 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} ($ICN = 0, 1, \ldots, |V_C|-2$). 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 (a) shows the same example that has been represented in Section 3. The transformed graph is represented in Fig. 3 (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, 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 $Z_M(ICN, 3) = \{Z_M(0, 3) = 7, Z_M(1, 3) = 11, Z_M(2, 3) = 15\}$, respec-

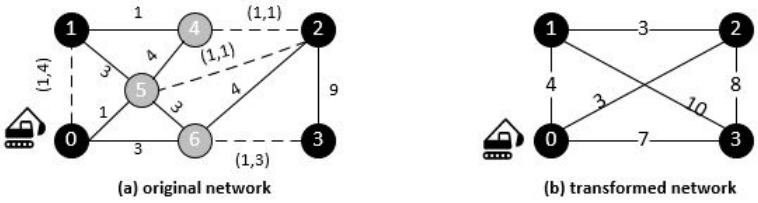

Figure 3: CMLB example

tively. 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 destination 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 $\eta_{ICN} = \mathbf{min}_{D \in V_C \setminus \{0\}} Z_M(ICN, D)$. 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 ($i = 0, 1, \ldots, |V_C| - 2$). 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 $i = 0, 1, \ldots, |V_C| - 2$), 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 $P = 0[1][2] \ldots [n]$ 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, $\eta_{[i]} = \mathbf{min}_{D \in V_C \setminus \{0\}} Z_M(i, D)$, it is less than or equal to $L_{[i]}$. In fact, $\eta_{[i]}$ takes the minimum latency value in po-

sition 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 $i = 0, 1, 2, \ldots, |V_C| - 2$, 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. □

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

Input:

The original graph G and data associated with G

Create a complete graph, $G_C = (V_C, E_C)$ on the set of critical nodes and the depot node.
Find the shortest path time sp_{ij}, for all $(i, j) \in E_C$.
for D in $V_C \setminus \{0\}$ **do**
    for ICN= $0, 1, \ldots, |V_C| - 2$ **do**
        Solve MM-MIP(ICN,D)
        Implement the objective correction procedure.
    end for
end for
for ICN= $0, 1, \ldots, |V_C| - 2$ **do**
    $\eta_{ICN} = \mathbf{min}_{D \in V_C \setminus \{0\}} Z_M(ICN, D)$
end for
Calculate the total latency value ($\sum_{ICN=0}^{|V_C|-2} \eta_{ICN}$) and assign it to CMLB.

5. 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 matheuristic that solves an MIP on a transformed graph. Second, we develop a metaheuristic which does not need a commercial solver.

5.1. Matheuristic approach

In this section, we suggest a matheuristic approach to solve instances of ML-RCP with high number of critical nodes. Similar to Section 4, 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 4.

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 $1, 2, \ldots, n$.

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

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;

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

$$\min z = n \sum_{i \in V_C \setminus \{0\}} sp_{0,i} x_{i,1} + \sum_{k \in K \setminus \{n\}} \sum_{i \in V_C \setminus \{0\}} \sum_{j \in V_C \setminus \{0\}, j \neq i} (n-k) sp_{ij} y_{ijk} \tag{16}$$

s.t.

$$\sum_{k \in K} x_{ik} = 1 \quad \forall i \in V_C \setminus \{0\} \tag{17}$$

$$\sum_{i \in V_C \setminus \{0\}} x_{ik} = 1 \quad \forall k \in K \tag{18}$$

$$\sum_{j \in V_C \setminus \{0\}, j \neq i} y_{ijk} = x_{ik} \quad \forall i \in V_C \setminus \{0\}, k \in K \setminus \{n\} \tag{19}$$

$$\sum_{j \in V_C \setminus \{0\}, j \neq i} y_{ijk} = x_{i,k+1} \quad \forall i \in V_C \setminus \{0\}, k \in K \setminus \{n\} \tag{20}$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in V_C \setminus \{0\}, k \in K \tag{21}$$

$$y_{ijk} \geq 0 \quad \forall i \in V_C \setminus \{0\}, j \in V_C \setminus \{0\}, j \neq i, k \in K \setminus \{n\} \tag{22}$$

The objective is to minimize the total latency. Constraints (17) guarantee that each critical node must occupy a single position. Constraints (18) force that each position is allowed to be captured by a single critical node. By Constraints (19), only one arc can leave from position k. Constraints (20) ensure that only one arc can arrive at position $k+1$. Finally, Constraints (21) and (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 4.

We use the same example shown in Fig. 3 to illustrate the matheuristic. 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 matheuristic approach is 25, which is equal to the optimal solution of ML-RCP.

The steps of the matheuristic algorithm are given in Algorithm 2.

Algorithm 2 Steps of the matheuristic 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.

5.2. 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.

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.

5.2.1. GRASP

GRASP is a metaheuristic approach that was first introduced by Feo and Resende (1995). 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 4. 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 α 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 5.1).
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

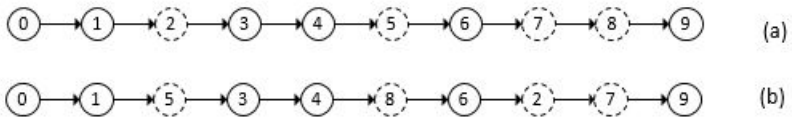

Figure 4: An example illustrating the shaking procedure

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).

5.2.2. VNS

For the improvement procedure, a local search based on VNS is executed. This metaheuristic 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 Section 7, 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. 4 depicts an example of the shaking procedure in which nodes 2, 5, 7 and 8 are removed from the current solution (Fig. 4(a)) and then inserted in different positions (Fig. 4(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 $N_1, N_2, ..., N_{k_{max}}$ 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 $\alpha_1, \alpha_2, ..., \alpha_p$ 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.

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

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 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.

Table 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

- **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.

6. 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.

6.1. Kartal data

The Kartal data is taken from Kılcı et al. (2015) and Şahin et al. (2016) and is based on a complete network with 45 nodes. Table 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 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 2 presents the SOE and the corresponding number of blocked edges in these instances.

Table 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

For example, instances k1,...,k5 have the same number of blocked edges (124) with SOE=1, while the set of blocked edges are different.

6.2. 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). 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. 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: $u_{ij} = t_{ij}.X$, 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.

7. Computational results

In this section, we compare the performance of the mathematical model proposed by Berktaş et al. (2016), the matheuristic 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.

The computational results are summarized in Tables 3 - 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 matheuristic 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 matheuristic 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 5.2, 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 (We explained the time intervals in Section 5.2.2).

7.1. Kartal data results

Table 3 shows the computational times of Berktaş et al. (2016)'s MIP model (T_E), the matheuristic (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 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 matheuristic 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 matheuristic 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 matheuristic (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. 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 matheuristic 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 matheuristic 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 matheuristic 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 matheuristic 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 matheuristic. 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 matheuristic (T_H) is still very small, namely around 1s. Additionally, the average $D\,(\%)$ between the two methods (matheuristic and metaheuristic) is reasonably small and the average gap between the lower bound and the upper bound provided by the matheuristic 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 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.

7.2. 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 matheuristic and the metaheuristic

Table 3: Results of Kartal data sets for the mathematical model, CMLB and the matheuristic

No.CN	3			4				7			
I.N	$T_E\,(s)$	$T_H\,(s)$	$T_{CMLB}\,(s)$	$T_E\,(s)$	$T_H\,(s)$	$T_{CMLB}\,(s)$	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

Table 4: Results of Kartal data sets for CMLB, matheuristic 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

for this data set. The results for both simplified and southwestern Istanbul networks are summarized in Tables 5 - 8. In each table the instance name is stated in the first column. Similar to Section 7.1, 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 matheuristic finds a better solution. On the other hand, if $D(\%)$ is negative, the metaheuristic obtains a better solution in comparison to the matheuristic. 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 5 - 6, the average $D(\%)$ increases once the number of critical and intermediate nodes increases indicating that the solutions of the matheuristic 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 metahuristic up to 30 minutes. It is also worth mentioning that the average gap (%) between the matheuristic and CMLB is at most 11.8%.

Tables 7 and 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 matheuristic approach in the post-disaster response stage. Note that since the $D(\%)$ values in Tables 5 - 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. 5 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.

7.3. 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 3. We also coded the matheuristic 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

Table 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 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 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 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

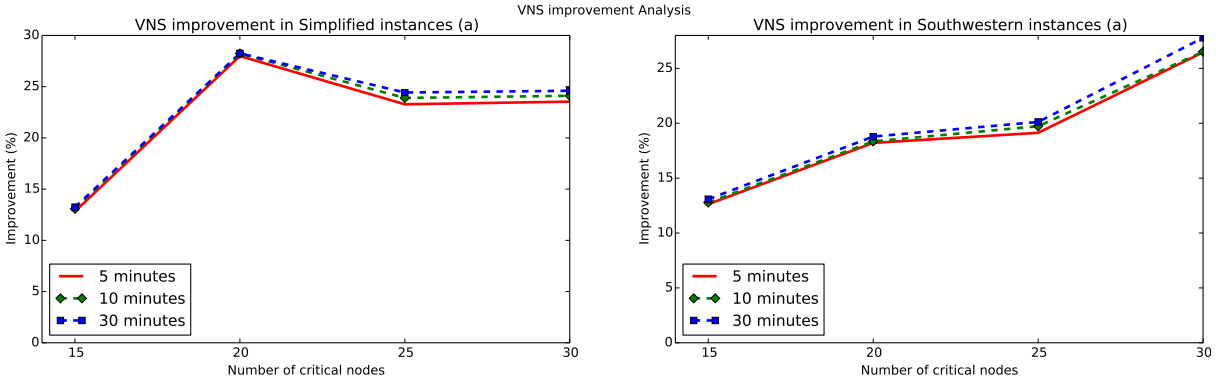

Figure 5: VNS analysis for simplified and southwestern Istanbul

tested their model with up to 15 critical nodes for the matheuristic method. We present their results in Table 9. For 11 critical nodes, both methods reached the optimal solution in all instances, with their corresponding computational times shown as $T_{\text{Berktas et al}}$ and T_H. (T_H shows the computational time of our matheuristic 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 matheuristic. 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 9. In contrast, the MIP in our method is solved exactly in an average time ($T_{total} = T_H + T_{CMLB}$) of 26.2 seconds. Note that Berktaş et al. (2016) have not considered an objective correction procedure.

8. 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

Table 9: Heuristic results of Berktaş et al. (2016) for Kartal data sets

No.CN	11		15 (within 1-hour time limit)		
I.N	$T_H\,(s)$	$T_{\text{Berktas et al}}\,(s)$	$T_{total}\,(s)$	gap (%)	Berktas 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

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 matheuristic.

$$\sum_{i \in V_C \setminus \{0\}} sp_{0,i} x_{i,1} + \sum_{k \in K \setminus \{n\}} \sum_{i \in V_C \setminus \{0\}} \sum_{j \in V_C \setminus \{0\}, j \neq i} sp_{ij} y_{ijk} \leq T_{updated} \tag{23}$$

Constraint (23) 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 (23) 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

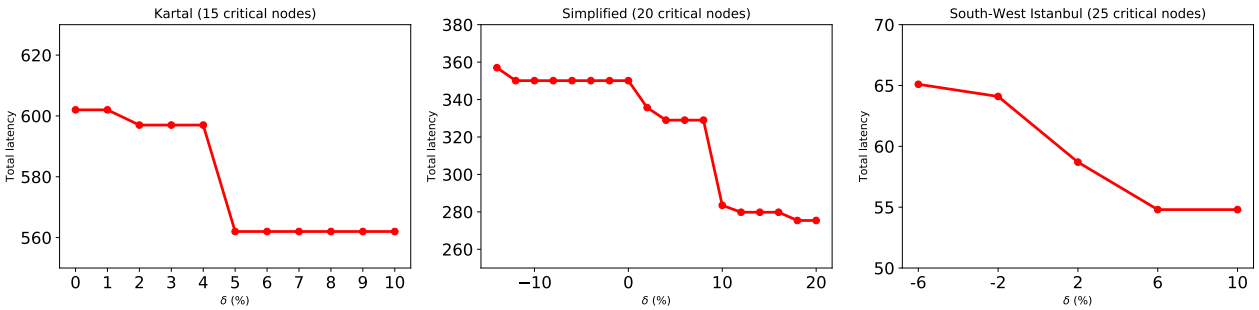

Figure 6: Results of ML-RCP under maximum latency constraint

value. However, in the matheuristic, the objective correction procedure is executed at the end. Thus, prior to running the new formulation, we add the clearing time every time the vehicle traverses the blocked edge in Constraint (23) 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. 6 shows the total latency versus the δ changes in the RHS of Constraint (23). 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 (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. 6, 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 between -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.

9. Conclusion

We proposed solution methods for the road clearance problem arising in the disaster response phase to minimize the total latency of reaching the critical locations. 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 matheuristic approach to find high quality feasible solutions within a short time. For both Kartal and Istanbul data sets, the matheuristic 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 matheuristic runs in much less time. We also compared our matheuristic with the matheuristic 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 matheuristic'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.

For future work, considering multiple vehicles dispatched from a single/multi depot would be an important extension that may better represent the situation after a disaster. In this regard, we note that solving multi-vehicle problems in ML-RCP is computationally much more challenging due to the need to coordinate the vehicles, which arises because an edge opened by a vehicle can later be traversed by other vehicles. Since the road clearance problem should be solved after a disaster, it is crucial to find a solution quickly. Therefore, an efficient solution method for the single vehicle problem is useful in a decomposition approach where the disaster affected region is partitioned into zones and a single vehicle problem is solved for each zone.

Acknowledgment

This research has been supported by TUBITAK grant 114M373.

References

Akbari, V. and Salman, F. S. (2017a). Multi-vehicle prize collecting arc routing for connectivity problem. *Computers & 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 Garca, 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.

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.

Çelik, M. (2016). Network restoration and recovery in humanitarian operations: framework, literature review, and research directions. *Surveys in Operations Research and Management Science*, 21(2):47–61.

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

Şahin, H., Kara, B. Y., and Karaşan, 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 & Operations Research*, 40(7):1700–1707.

Duque, P. M. and Sörensen, K. (2011). A grasp metaheuristic to improve accessibility after a disaster. *OR spectrum*, 33(3):525–542.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Gavish, B. and Graves, S. C. (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*, 82(1):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.

Kılcı, F., Kara, B. Y., and Bozkaya, B. (2015). Locating temporary shelter areas after an earthquake: A case for turkey. *European Journal of Operational Research*, 243(1):323–332.

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.

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.

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.

Özdamar, L., Aksu, D. T., and Ergüneş, 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.

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: A Quarterly Journal of Operations Research*, 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)*.

Silva, M. M., Subramanian, A., Vidal, T., and Ochi, L. S. (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 KÅivánková, 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*, 30(1):63–75.

Yan, S. and Shih, Y.-L. (2009). Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers & 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.