Routing Multiple Work Teams to Minimize Latency in Post-disaster Road Network Restoration[1]

Meraj Ajam[a], Vahid Akbari[b], F. Sibel Salman[2c]

[a]*Haskayne School of Business, University of Calgary, Calgary T2N 1N4, Alberta, Canada*
[b]*Nottingham University Business School, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, United Kingdom*
[c]*College of Engineering, Koç University, 34450 Sariyer, Istanbul, Turkey*

Abstract

After a disaster, often roads are damaged and blocked, hindering accessibility for relief efforts. It is essential to dispatch work teams to restore the blocked roads by clearance or repair operations. With the goal of enabling access between critical locations in the disaster area in shortest time, we propose algorithms that determine the schedule and routes of multiple work teams. We minimize the total latency of reaching the critical locations, where the latency of a location is defined as the time it takes from the start of the operation until its first visit by one of the work teams. Coordination among the teams is needed since some blocked edges might be opened by a certain team and utilized by other teams later on. First, we develop an exact mathematical model that handles the coordination requirement. After observing the intractability of this formulation, we introduce two heuristic methods and a lower bounding procedure. In the first method, we develop a mathematical model based on a novel multi-level network representation that yields solutions with disjoint paths. Given that it does not coordinate the teams, we present a matheuristic based on a cluster-first-route-second approach embedded into a local search algorithm together with an additional coordination step to obtain alternative solutions with higher quality and in a shorter time. We test our heuristics on data sets coming from a real network from the literature (180 instances) and randomly generated ones (640 instances) and observe the superiority of the solutions obtained by incorporation of coordination.

Keywords: Humanitarian logistics; disaster response; road clearance; network restoration; minimum latency; matheuristic

1. Introduction

Natural disasters are known to damage infrastructure systems, which in turn obstructs the relief efforts. Often, road and highway segments, bridges, and viaducts get damaged to the degree of rendering these structures impassable by vehicles. In addition, debris of collapsed buildings, fallen trees, displaced cars and flooding commonly cause disconnectedness in the road networks leading to the isolation of people in need of immediate help or casualty locations. Moreover, this situation impedes accessibility to critical locations

[1]DOI: https://doi.org/10.1016/j.ejor.2021.07.048
[2]Corresponding Author
Email addresses: meraj.ajam@ucalgary.ca (Meraj Ajam), vahid.akbari@nottingham.ac.uk (Vahid Akbari), ssalman@ku.edu.tr (F. Sibel Salman[2])

such as hospitals, relief supply facilities, local points of dispense and transportation hubs, that are strategic for the success of the relief efforts. Therefore, one of the first immediate response actions taken to mitigate the negative impacts of a disaster is to open the roads on the routes between these critical locations. While many road segments may be blocked or damaged, restoring a group of them is crucial for the first-responders to reach people in need. Therefore, which blocked roads to open in the immediate stage should be selected carefully and the routes of the work teams must be determined to reach each critical location in the shortest time. Prior to dispatching the work teams, information regarding the blocked roads should be obtained. This can be done by means of drones, helicopters, satellite images or volunteers. With this information, clearance or repair, that is, the total restoration time of a road segment or structure can be estimated, and as a result, the routes of the work teams can be optimized.

We define the road restoration problem as follows. We represent the roadway by a network and the so-called "critical locations" constitute a subset of nodes that should be reached by the work teams. An edge represents a connection between two nodes via a path. Each edge has a traversal time. A subset of the edges are "blocked" and an additional "restoration time" is required to complete the necessary tasks to make each blocked edge operational. An edge that is restored can be traversed by the same or other teams afterwards. Each work team is initially located at a depot node. The optimal routes should specify which roads should be restored by which work team in what order, so that each critical node is accessed by a work team. Since timely access is of utmost importance for the success of first response operations, we select the objective of the optimization problem as minimizing the summation of the latency of all the critical nodes. The latency of a critical node is defined as the time elapsed between the beginning of the operation and the time when the corresponding critical node is reached for the first time. As such, the latency of a critical node is the waiting time until that node is reached and our goal is to make the waiting times of the critical nodes as small as possible, rather than just the total time of the operation. Ajam et al. (2019), Berktaş et al. (2016), Sahin et al. (2016) studied the problem of minimizing total latency of the critical nodes with a single work team under different problem names. Ajam et al. (2019) called this problem the Minimum Total Latency in Road Clearance Problem (ML-RCP) and provided an example to clarify why minimizing total latency is more preferable over alternative objectives such as minimizing the makespan.

We study a generalization of the ML-RCP with multiple work teams and call the addressed problem the Multiple Team Minimum Latency in Road Clearance Problem (MML-RCP). An important distinction is that in the case with a single work team, coordination concerns regarding traversal of blocked edges by different teams is not an issue, whereas in our problem multiple teams navigate the same network. Coordination among the teams is needed since some blocked edges might be restored by a certain team and utilized by other teams later on without spending time to restore it. Clearly, having multiple work teams working in parallel gives an opportunity to minimize the latency values further in comparison to the single work team case, hence accelerating the response efforts. On the other hand, having multiple work teams in the same network complicates the optimization problem significantly due to the coordination need. A few studies in road restoration (Akbari and Salman 2017a,b, Akbari et al. 2021a) have already considered the coordination requirement with multiple teams but under different objectives. To the best of our knowledge, this is the

first study to address the road restoration problem with minimum latency objective and multiple teams. We address this problem both when all the work teams are pre-positioned in the same depot and in the more general case where multiple depots exist so that each team can initiate its route from a different node.

In the routing literature, it has been observed that the latency objective makes the problem more challenging compared to the usual objective of minimizing the length of the routes. The problem of finding a single tour visiting each node in the graph and minimizing the total latency of the nodes is called the Minimum Latency Problem (MLP) (Sahni and Gonzalez 1976). Despite having similarities with the Traveling Salesman Problem (TSP), the MLP is known to be more difficult to solve computationally compared to the TSP (Goemans and Kleinberg 1998). The problem with multiple vehicles originating from the same depot and total latency minimization objective has been tackled only under the requirement that all nodes should be visited (Angel-Bello et al. 2017). Since their context is not post-disaster planning, Angel-Bello et al. (2017) did not consider blocked roads and the decisions on which ones to unblock and in which order. Hence, the inherent difficulty of our problem due to dynamically changing shortest path distances while the roads are restored does not exist in their study. Furthermore, for our study, visiting all nodes is not practical, especially in the aftermath of a disaster in which time plays an essential role in the rate of casualties. Thus, it is necessary to visit a set of pre-identified nodes, referred to as critical nodes, while other nodes may also be visited if this provides a better solution. This property also adds a different dimension to our problem compared to the problem in Angel-Bello et al. (2017).

In this article, we first develop a mathematical model (denoted by M1) that solves the MML-RCP, where the coordination requirement is satisfied. However, M1 is able to solve the MML-RCP for very small sized instances (with our computational platform, instances having up to only 13 nodes, 4 critical nodes and 2 work teams could be solved optimally). Hence, due to the need to solve larger instances, we introduce two heuristic procedures yielding solutions with different natures. The first one, called H1, is based on a multi-level network model designed for a restricted variation of the MML-RCP, namely, the case in which the routes of the work teams are generated to be disjoint from each other. In the solution of the H1, since the routes of the teams are disjoint, there is no coordination between the teams. In our second heuristic, called H2, we incorporate coordination among the teams. H2 is a matheuristic, that may find solutions with overlapping routes. H2 is based on a cluster-first-route-next approach that is embedded into a local search procedure augmented with iterative mathematical model solutions and a coordination sub-routine that dynamically updates the road network when a blocked edge is recovered by one of the teams. In H2, a mixed integer programming (MIP) model is solved for the single team routing subproblems after each inter-route move. In order to investigate the efficacy of H1 and H2, and to show how coordination can enable us to obtain better solutions, we test them with instances both from the literature based on a real city network and those generated randomly, where the depot node(s), critical nodes, the number of blocked edges and their locations differ instance by instance. Our main contribution is developing the two solution approaches that are capable of solving instances of practical sizes due to their small run times. We also develop a lower bounding procedure in order to measure how far the solutions output by the heuristics are from the optimal values. The lower bounding procedure solves a series of MIP models.

The paper is organized as follows. A review of the related literature is given in Section 2. Section 3 provides the problem description. The heuristic approaches and the lower bounding procedure are presented in Section 4. Sections 5 and 6 describe the data sets and present the computational results, respectively. Finally, conclusions are given in Section 7.

2. Literature review

First, we discuss studies associated with road clearance/restoration in disaster logistics having different objectives. Second, we focus on the Minimum Latency Problem (MLP) in the routing literature.

2.1. Road clearance and network restoration

An important vein of studies in disaster logistics focuses on optimizing road clearance and restoration operations. The early studies have generally focused on finding the route of a single work crew along with determining which edges should be cleared after a disaster. As one of the first studies, Duque et al. (2016) developed an exact Dynamic Programming (DP) algorithm, which is able to provide the schedule and route of a repair crew for small scale instances. The authors also developed an iterated greedy-randomized constructive procedure to solve large scale instances. Kim et al. (2018) built upon the problem studied by Duque et al. (2016) and considered the case in which damage impact may vary over time after the initial disaster has occurred. In addition, they developed an Ant Colony Optimization (ACO) algorithm to tackle the problem with multiple repair crews.

The latency objective was undertaken in studies by Sahin et al. (2016), Berktaş et al. (2016) and Ajam et al. (2019), where all focused on a single work crew that should visit a set of critical nodes. The problem that we study in this article, MML-RCP, generalizes these studies by generating the routes of multiple teams positioned at different depots. Sahin et al. (2016) proposed both a mathematical model and a heuristic approach, minimizing total time until all critical locations are visited. Berktaş et al. (2016) formulated a mathematical model and developed a heuristic approach, which are more efficient than the ones in Sahin et al. (2016) in terms of computational time. However, the number of critical nodes in their test instances were quite small (i.e., up to 7 critical nodes), which eludes justifying the applicability of their methods for instances arising in larger networks found in practice. Ajam et al. (2019) proposed a matheuristic and a metaheuristic that largely outperform the methods proposed for the latency minimization problem in Berktaş et al. (2016). Furthermore, the authors verified the strength of their proposed algorithms on larger instances derived from real networks by introducing a novel lower bounding approach, which is based on solving MIP models iteratively. They solved instances with up to 30 critical nodes with their heuristic algorithm.

In another direction, several researchers focused on damaged networks decomposed to isolated components, where the goal is to restore the connectivity of the network. Among those, Vodák et al. (2018) developed a metaheuristic based on ACO that reconnects all isolated components by dispatching a single work crew. The authors solve the problem by reducing the size of the networks, by only considering the boundary nodes of each component. Kasaei and Salman (2016) defined two problems optimizing the route of a single work team that renders the network re-connected. The first problem aims to minimize the total

time elapsed until all disconnected components get connected. For the second problem, the objective is to maximize the total prize collected from reconnecting the components within a specified time limit. The authors developed two exact formulations along with metaheuristics. Different variations and generalizations of these problems were addressed in other studies. Akbari et al. (2021b) studied the online variation of the first problem in which the unblocking time of the blocked edges is not known in advance and is only revealed after close observation and estimation of the road restoration team. Akbari and Salman (2017b) developed a mathematical model for the generalization of the first problem studied in Kasaei and Salman (2016) and considered multiple work teams. The authors developed a matheuristic approach to solve larger instances, which is based on the relaxation of their mathematical model, followed by a procedure to restore feasibility along with local search as the final step. In another study, Akbari et al. (2021a) developed a decomposition-based heuristic algorithm and successfully applied it to solve larger instances of this problem with multiple work teams. For the generalization of the second problem with multiple work teams that was defined in Kasaei and Salman (2016), Akbari and Salman (2017a) formulated a mathematical model and derived an efficient heuristic based on Lagrangian relaxation. Note that when multiple work crews operate in the same network, coordination becomes the most challenging aspect of these problems. That is, when a blocked edge is on the route of multiple work crews, the crew that arrives first to the edge should unblock the edge while other crews may need to wait to pass along that edge. Both Akbari and Salman (2017b) and Akbari and Salman (2017a) resolved this issue in their algorithms. In another study, Morshedlou et al. (2018) proposed two mathematical models associated with the coordinated routing problem such that the number of crews designated for each disrupted component has been considered as a decision variable. In a recent study, Moreno et al. (2020) addressed a heterogeneous multi-crew scheduling and routing problem, where the goal is to restore the damaged nodes used in the paths connecting a source node to demand nodes in a network. The objective is to minimize the time that the demand nodes remain disconnected from the source node. The authors developed three mathematical models together with some valid inequalities.

Some variations of the debris clearance/road restoration problem were also studied recently. Li et al. (2020), integrated logistic support scheduling with a repair crew scheduling and routing problem with two stages. The authors proposed a novel non–linear programming model for the two stages. In addition, by means of a case study, the authors showed how the repair delays can reduce when the two stages get integrated. Lorca et al. (2017) developed a mathematical model for managing debris collection, disposal operations and transportation, where the goal is to reduce the conflict between objectives such as cost and duration. In addition, the authors provided a user-friendly decision support tool.

In several recent studies, road clearance and relief distribution operations are envisioned to be conducted by the same team. Shin et al. (2019) developed an MIP and an ACO algorithm to restore the network after a disaster. In addition to scheduling and routing of the repair crew, the authors incorporated the transportation of relief goods in their algorithms. Moreno et al. (2019) proposed a branch-and-Benders-cut (BBC) algorithm such that the scheduling decisions are obtained from a master problem and the subproblems optimize the routes. To improve the performance of their BBC algorithm, the authors used various feasibility cuts and valid inequalities. Li and Teo (2019) developed a multi-period bi-level programming model for the post-

disaster road network repair work scheduling and relief logistics problem. At the first level, the goal is to improve the accessibility of the road network over the periods by optimizing the road restoration decisions, while at the second level, the goal is to maximize the satisfaction of relief material demand in the given network. The authors developed a genetic algorithm for this problem in order to obtain good solutions for different instance sizes within a limited time.

After an investigation of the literature on road clearance/restoration, we see that the latency objective has not been addressed under the existence of multiple work crews. To the best of our knowledge, our study considers both multiple crews and the latency objective for the first time. We provide two alternative heuristic algorithms for the solution of this challenging problem.

2.2. Minimum latency problem

The latency objective was first brought to attention in the routing literature. In the Minimum Latency Problem (MLP), a vehicle is assigned to visit all nodes in a complete network, where all of the edges are intact. The objective of the MLP is to minimize the total waiting time of all the nodes instead of the total time of the vehicle's route as in the classical Traveling Salesman Problem (TSP). Since some of the literature on MLP pays particular attention to repair services at each node of the network, MLP is also referred to as the Traveling Repairman Problem (TRP). Note that MLP is also known as the Delivery Man Problem (DMP) and the Cumulative Vehicle Routing Problem (CVRP).

Exact methods for the MLP (with a single vehicle) have managed to solve larger instances over time. Sarubbi et al. (2008) suggested an exact formulation based on the model that had been developed for the TSP by Picard and Queyranne (1978). The authors used Newton's Barrier Method to derive lower bounds for the problem. Méndez-Díaz et al. (2008) formulated a three-index mathematical model that can solve instances having up to 40 nodes. They took advantage of effective valid inequalities. Angel-Bello et al. (2013) developed two different mathematical models for the MLP that are based on a multi-level network structure. The small elapsed time when solving these models shows the strength of their formulations. Bulhões et al. (2018) proposed a branch-and-price (BP) algorithm for the MLP, which is able to solve the unsolved instances introduced in Roberti and Mingozzi (2014).

Many researchers have attempted to solve the MLP/DMP (with a single vehicle) by developing heuristics and metaheuristics. Salehipour et al. (2011) proposed a metaheuristic based on the Greedy Randomized Adaptive Search Procedure (GRASP) to generate initial solutions, followed by Variable Neighborhood Descent (VND) and Variable Neighborhood Search (VNS) for the improvement phase. The authors provided lower bounds based on the minimum spanning tree and upper bounds by means of the nearest neighbor heuristic to test the effectiveness of their algorithm. Silva et al. (2012) also proposed a metaheuristic based on GRASP followed by VND with a different shaking mechanism for the initial solution, which is called double-bridge perturbation. Mladenović et al. (2013) developed a General VNS based on the classical neighborhood structures for the MLP that outperforms the algorithms proposed by Salehipour et al. (2011). Some researchers addressed DMP with profit, whose aim is to find a path maximizing the total revenue within a given time limit. Dewilde et al. (2013) employed VNS and Tabu search to solve the variation of the

MLP with time dependent profit based objective. Avci and Avci (2017) suggested a metaheuristic based on GRASP and ILS for the initial and improvement of solutions, respectively.

Some researchers investigated the capacitated variation of this problems referred to as Capacitated CVRP (CCVRP). This variation has been addressed first by Ngueveu et al. (2010) where the authors proposed a Memetic algorithm to tackle this problem with instances having up to 199 nodes. Ribeiro and Laporte (2012) suggested an Adaptive Large Neighborhood Search (ALNS) heuristic that outperforms the computational results provided by Ngueveu et al. (2010). Lysgaard and Wøhlk (2014) developed a Branch-and-cut-and-price redalgorithm which is capable of solving CCVRP instances with up to 70 nodes within reasonable computational times. Rivera et al. (2016) developed two MIP models (able to solve instances up to 40 nodes) to solve multi-trip CCVRP in a disaster context where a single vehicle performs multiple trips to serve a set of affected sites. Sze et al. (2017) developed a min-max CCVRP where the objective is to minimize the maximum arrival time, solved by a two-stage adaptive VNS (AVNS) algorithm that incorporates Large Neighbourhood Search (LNS). More recently, Lalla-Ruiz and Voß (2020) extended CCVRP where multiple depots exist as starting points of routes, solved by an MIP and a matheuristic approach for larger instances.

Several studies have been devoted to the multi-vehicle MLP as well. Luo et al. (2014) introduced a new branch-and-price-and-cut algorithm for the multi-vehicle MLP under a distance constraint for each vehicle. The authors tested the algorithm with instances having up to 50 nodes, where the algorithm could obtain optimal solutions for all the instances except one of them within a 3-hour time limit. Nucamendi et al. (2015) developed a mathematical model based on a single-commodity flow formulation to find m disjoint routes to visit all the nodes collectively, where m is the number of vehicles originating from a depot. With this formulation, they were 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 which are based on the distance of the nodes to the depot node and incorporated 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 utilized. They compared 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 given in Nucamendi et al. (2015). To solve larger instances, they developed a metaheuristic based on an iterated greedy algorithm for constructing an initial solution. For the improvement phase, they employed five local search strategies which are arranged in two intra- and inter-route composite moves. In order to understand the performance of their metaheuristic for larger instances, they derived strong lower bounds from the CCVRP in the literature, where they relaxed the constraints associated with demand and supply nodes' capacities. Bang (2018) developed a metaheuristic algorithm which is based on GRASP for initial solution construction. Furthermore, the author implemented a VND algorithm for improvement by considering a distance constraint for each vehicle. 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 of the mentioned studies, the authors developed their models based on the assumption that a disjoint path is assigned to each vehicle. However, in our problem we relax this assumption. Very recently, Muritiba et al. (2021) proposed a tailored Branch-and-cut and ILS for the weighted k-TRP, which is able to solve instances having up to 50 nodes for the case of maintenance of speed cameras.

Thus far, none of the studies associated with the multi-vehicle MLP incorporated restoration of a damaged network. In addition, in the multi-vehicle MLP, all nodes must be visited. However, in the MML-RCP, only the critical nodes should be visited while it is optional to visit the other nodes. Moreover, we incorporate multiple depots in our model to reduce the latency of reaching the critical locations.

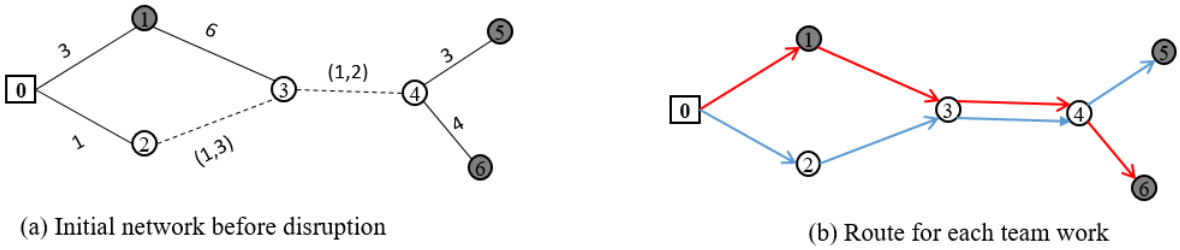
3. Problem definition

We represent the road network in the disaster area by an undirected graph $G = (V, E)$, where V and E denote the set of nodes and the set of edges, respectively. A traversal time t_{ij} associated with edge $(i, j) \in E$ is given for each edge. Blocked edges are known and represented by the set $B \subseteq E$. In addition, an extra time for clearing/unblocking a blocked edge (i, j) is given as u_{ij}. 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 work teams (crews) are positioned in the depot node(s) at the beginning of the operations. Each work team departs from a specific depot node to visit a subset of the critical nodes and finishes its route at one of the critical nodes while it may open a number of blocked edges to facilitate access to the critical nodes. Each work team must visit at least one critical node through its route and each critical node must be visited by at least one work team at the end of the operations. Non-critical nodes may be visited by the work teams through their routes. These are called "intermediate" nodes. It is possible that one or more work teams visit a critical node more than once. However, the first visit time of each critical node is taken into account as the latency of that node. In addition, during the operation, due to the clearance of some blocked edges, the shortest path lengths between the nodes may change, indicating that in the MML-RCP, we cannot pre-process the problem and only consider the critical nodes. The MML-RCP aims to find a route for each work team such that the total latency of all critical nodes is minimized. In addition, the solution specifies which blocked edges should be opened in which order by each work team.

In MML-RCP, we make the assumption that only the first team that arrives to a blocked edge opens it because we assume that each team is equipped with all the necessary machinery to open a blocked edge and a cooperation in this sense cannot improve the unblocking time. Even if a second team at the scene would work and speed up the process, it is very difficult to assess realistically how the speed up can be measured.

Figure 1(a) shows an example of the MML-RCP with two work teams positioned in the depot node indexed by 0. In this example, both of the work teams are positioned in the same node, while it does not have to be so in general. Nodes 0, 1, 5 and 6 are critical and the other nodes (2, 3 and 4) 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. The route of the first work team (shown in Figure 1(b) with blue arrows) includes nodes 0-2-3-4-5, where the latency of critical node 5 is 11 and the intermediate nodes 2, 3 and 4 have been visited in the route. Moreover, this work team cleared blocked edges (2,3) and (3,4) within its route, making these edges available for next traversals. The route of the second work team (shown in Figure 1(b) with red arrows) includes nodes 0-1-3-4-6, where the latency of critical nodes 1 and 6 are 3 and 14, respectively. In this route, nodes 3 and 4 are intermediate nodes. Note that since the first work team arrives to node 3 earlier than the second work team, blocked edge (3,4) would be already cleared by the first work team. Therefore, when the second work team arrives to node 3, it traverses the opened edge (3,4) in 1 unit of time. At the end, the work teams have visited all critical nodes with a total latency of 28 (11+3+14), which includes both traversal and clearing times in the objective value. This example shows our understanding of coordination between the work teams in the sense that edge (3,4) was opened by one team and used by another.



(a) Initial network before disruption

(b) Route for each team work

Figure 1: Example of MML-RCP

4. Solution methods

The MML-RCP 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 and since Sahni and Gonzalez (1976) proved that MLP is NP-hard, so is the multi-vehicle MLP and hence the MML-RCP is also NP-hard.

In order to tackle the MML-RCP, we first developed a mathematical model (M1) in the presence of coordination/synchronization such that when a work team is cleaning a blocked edge, the other work teams are not allowed to traverse that edge until it is unblocked completely. In later traversals, only the traversal time of that edge should be taken into account (i.e., the clearing time of a blocked edge incurs only once). M1 is adapted from the mathematical model suggested in Akbari and Salman (2017b) for a different objective. Incorporation of the latency objective in that formulation required us to further modify it by adding a number of constraints. In particular, M1 must calculate the first visiting time of a critical node as its latency. To do so, we had to add multiple constraints. We present M1 in the Appendix. As expected, M1 is only capable of solving small instances. We have also tried to relax the most time-consuming constraints to get some meaningful solutions (similar to what Akbari and Salman (2017b) did). However, the constraints associated with visiting time of each critical node have huge obstacles, preventing the model to get solutions for larger instances. Moreover, since we use the visiting time of each critical node in the objective function, the

objective value becomes zero if we remove the visiting time constraints. Thus, even modified methods proposed in Akbari and Salman (2017b) are not applicable to this problem and we had to develop H1 and H2 algorithms. We were able to solve instances with up to 13 nodes, 4 critical nodes and 2 work teams with our computational platform. Once we add more nodes, the model could not be solved even within a 5-hour time limit. In order to find feasible solutions to the problem, we first assume that the paths of the work teams should be disjoint (similar to the studies related to the multi-vehicle MLP). We develop a model that can be solved for larger instances by enforcing this assumption. In this way, we obtain the heuristic H1. Given that this assumption prevents coordination among the work teams, we develop a matheuristic (H2) in Section 4.2 that relaxes this assumption and addresses the underlying synchronization issue as well.

4.1. Mathematical model based on a multi-level network (H1)

In order to find feasible solutions for the MML-RCP within a reasonable time limit, we first assume that the paths of the work teams must be disjoint. We propose a mathematical model defined on a multi-level network as in Angel-Bello et al. (2017), but the network structure is different from those given in Angel-Bello et al. (2017). First, we briefly explain the multi-level network proposed by Angel-Bello et al. (2017) and then describe our problem with an example and highlight the differences.

Figure 2 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 work teams, respectively. Level 1 consists of a copy of the set V except the depot node. Levels 2, 3, ... , N consist of nodes that are copies of those in V. Finally, level N+1 consists of a 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, ... , N+1) and visits the lower levels and reaches level 1 for the final node. Each work team must visit a single node in each level indexed lower than the starting node. For instance, if a work team 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 work teams depart from the depot node, which is positioned in levels 2, 3, ... and $N + 1$.

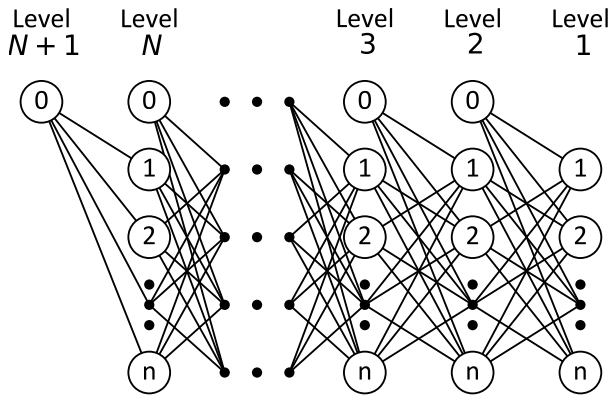


Figure 2: Graphical representation of the multi-level network in Angel-Bello et al. (2017)

Figure 3 demonstrates a feasible solution for the multi-level network problem described in Figure 2. There are 8 nodes (including the depot node 0) in this example, where all of them must be visited by two

work teams. Work team 1, whose route is shown by narrow lines, starts its route from the depot node in level 5 and visits nodes 1, 4, 7 and 6. On the other hand, work team 2, whose route is shown by thick lines, departs from the depot node in level 4 and visits nodes 3, 2 and 5. Thus, all of the nodes are visited by the work teams.

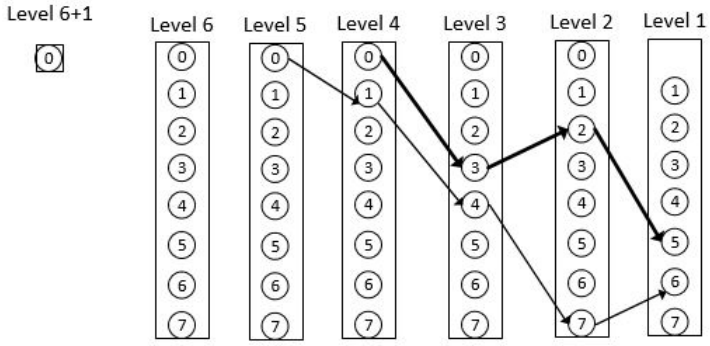


Figure 3: A feasible solution to the multi-level network problem in Angel-Bello et al. (2017)

In the following, we describe the multi-level network associated with our problem by listing its differences from the network given in Figure 2:

1. Unlike Angel-Bello et al. (2017)'s network, we define levels only for the critical nodes and we refer to them as *critical levels*. Such critical levels are indexed from 0 to $N = |V_C| - m$, where V_C is the set of critical nodes including the depot node.

2. We allow visiting 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 in the routes, the definition of decision variables and constraints changes significantly.

4. Our multi-level network can model instances of the MML-RCP with multiple depots (see Section 4.1.1).

Figure 4 (a) gives a small instance of the described network. It 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 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 visited through the routes. Two work teams are assigned to visit all the critical nodes collectively, whose routes are depicted with narrow and thick black lines. The number of critical levels is $N = 5 - 2 = 3$. 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 levels n and $n + 1$.

Parts (b) and (c) of Figure 4 illustrate some of the feasible routes provided by the MML-RCP. As we observe, each work team visits at least one critical node. A work team may or may not visit the non-critical nodes throughout its route. For instance, in Figure 4 (b), the work team whose route is shown by the narrow arrows starts its route from level 3+1, visits node 6 and then the critical node 7. This work team then directly

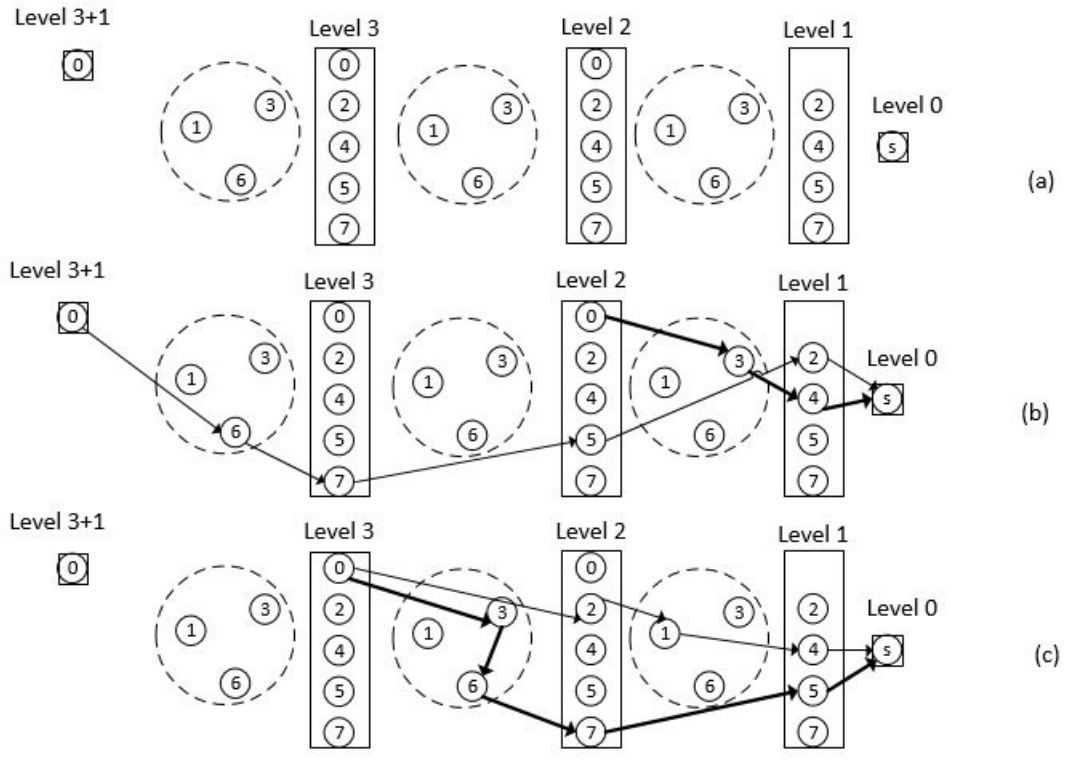


Figure 4: An example of the multi-level network structure

continues to critical node 5 without visiting any non-critical nodes. In another example depicted in Figure 4 part (c), the work team whose route is given by the thick arrows, 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 work team continues on the path to reach a critical node. We have developed this model to address the difficulties brought by the latency objective. We note that models that are typically used to solve general node routing problems, such as the VRP, do not perform well when used on latency objectives (Goemans and Kleinberg 1998).

We next introduce our mathematical model for the restricted version of the MML-RCP. We refer to this model as H1-MIP. In this model, we assume that only a single depot exists. We then explain in Section 4.1.1 how to generalize this model for the case with multiple depots.

In addition to the notation given in the problem definition, we define s as a dummy sink node. Recall that we added a restriction to the MML-RCP by assuming 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.

Decision variables:

$y^r_{ijkl} = 1$, 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 relax the integrality of this variable in the model)

$x^r_{ikl} = 1$, 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

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

s.t.

$$\sum_{l\in V_C\setminus\{0,k\}} \sum_{r=1}^{N-1} x^r_{kkl} + x^0_{kks} = 1 \qquad \forall k \in V_C \setminus \{0\} \tag{2}$$

$$\sum_{k\in V_C\setminus\{0\}} x^0_{kks} = m \tag{3}$$

$$\sum_{l\in V_C\setminus\{0\}} \sum_{k\in V_C, k\neq l} \sum_{r=1}^{N} x^r_{ikl} \leq 1 \qquad i \in V \setminus V_C, i \neq s \tag{4}$$

$$\sum_{l\in v_C} \sum_{k\in V_C\setminus\{0\}, k\neq l} \sum_{i\in (V\setminus V_C)\cup\{l\}, i\neq s} y^1_{iklk} = m \tag{5}$$

$$\sum_{r=1}^{N} \sum_{j\in (V\setminus V_C)\cup\{l\}, j\neq s} \sum_{l\in V_C\setminus\{0\}} y^r_{0j0l} = m \tag{6}$$

$$\sum_{j\in (V\setminus V_C)\cup\{l\}, j\neq i,s} y^r_{ijkl} = x^r_{ikl} \qquad r = 1, 2, \ldots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l,$$
$$i \in (V \setminus V_C) \cup \{k\}, i \neq s \tag{7}$$

$$y^r_{0j0l} + \sum_{i\in (V\setminus V_C)\cup\{k\}, i\neq j,s} y^r_{ijkl} = x^r_{jkl} \qquad r = 1, 2, \ldots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l,$$
$$j \in (V \setminus V_C) \cup \{l\}, j \neq s \tag{8}$$

$$\sum_{i\in (V\setminus V_C)\cup\{l\}} \sum_{l\in V_C, l\neq k} y^r_{iklk} = \sum_{w\in V_C\setminus\{0\}, w\neq k} x^{r-1}_{kkw} \qquad k \in V_C \setminus \{0\}, r = 2, 3, \ldots, N-1 \tag{9}$$

$$\sum_{i\in (V\setminus V_C)\cup\{l\}, i\neq s} y^1_{iklk} = x^0_{kks} \qquad k \in V_C \setminus \{0\}, l \in V_C, k \neq l \tag{10}$$

$$\sum_{l\in V_C, l\neq k} x^1_{klk} = x^0_{kks} \qquad k \in V_C \setminus \{0\} \tag{11}$$

$$x^r_{kkl} = x^r_{lkl} \quad r = 1, 2, \ldots, N-1, k \in V_C, l \in V_C \setminus \{0\}, k \neq l \tag{12}$$

$$x^N_{00l} = x^N_{l0l} \qquad l \in V_C \setminus \{0\} \tag{13}$$

$$\sum_{i\in (V\setminus V_C)\cup\{0\}} y^N_{ij0l} = x^N_{j0l} \qquad l \in V_C \setminus \{0\}, j \in (V \setminus V_C) \cup \{l\}, j \neq s \tag{14}$$

$$\sum_{j\in (V\setminus V_C)\cup\{l\}} y^N_{ij0l} = x^N_{i0l} \qquad l \in V_C \setminus \{0\}, i \in (V \setminus V_C) \cup \{0\}, i \neq s \tag{15}$$

$$\sum_{k\in V_C\setminus\{0,l\}} x^{N-1}_{llk} = x^N_{l0l} \qquad l \in V_C \setminus \{0\} \tag{16}$$

$$\sum_{i\in (V\setminus V_C)\cup\{0\}} y^N_{ik0k} = \sum_{w\in V_C\setminus\{0\}, w\neq k} x^{N-1}_{kkw} \qquad k \in V_C \setminus \{0\} \tag{17}$$

$$y^r_{ijkl} \geq 0 \qquad i, j \in V,\ k, l \in V_C,\ k \neq l,\ r = 1, 2, \ldots, N \tag{18}$$

$$x^r_{ikl} \in \{0, 1\} \qquad i \in V,\ k, l \in V_C,\ k \neq l,\ r = 0, 1, \ldots, N \tag{19}$$

The objective function (1) minimizes the total latency of all critical nodes. Constraints (2) ensure that each critical node is visited exactly once. Constraint (3) guarantees that all work teams must go to the dummy sink node at the end of their routes. The non-critical nodes may be visited at most once due to Constraints (4). Constraint (5) forces all work teams to visit level 1 at one of the critical nodes as a destination node. Constraint (6) ensures that exactly m disjoint paths are assigned to the work teams. Constraints (7)-(17) are the connectivity constraints that guarantee the continuity of the paths between each consecutive critical level. We divide the connectivity constraints into two groups. The first group (7)-(12) consists of the connectivity constraints belonging to levels 0, 1, 2, ... and N. Constraints (7)-(12) are the outgoing and incoming flow balance equations. Since in level N+1 only the depot node exists, we have to consider separate constraints for this level (Constraints (13)-(17)) to ensure the connectivity of each path. Moreover, these constraints guarantee the continuity of the paths in case any non-critical nodes are visited between critical levels. Constraints (18) and (19) define the domains of the variables.

In our computational tests we observed H1-MIP to be quite efficient as it satisfies a number of features: 1) There is no index defined associated with work teams, 2) Level numbers have been used in calculating total latency directly in the objective function. Thus, we do not need constraints associated with calculating the visiting times of each critical node, 3) Variable y can be relaxed to reduce the computational time.

Proposition 1. *Decision Variable y can be relaxed in H1-MIP.*

Proof. For any feasible solution, x^r_{ikl} must be binary because of the decision variable definition associated with H1-MIP. Now, if we replace the x^r_{ikl}s with their values (0 or 1), H1-MIP becomes an LP problem, which can be solved by the Simplex method. The right hand side of each constraint for this LP problem can be zero, one or m (number of vehicles). Moreover, all coefficients associated with y variables (inside the simplex table) are zero or one. Thus, for the first iteration, the minimum ratio is $\min\{0, 1, U\} = 0$ such that U is an integer number. Note that for the next iterations, we need to avoid degeneracy. In this case, the minimum ratio will be $\min\{1, U'\} = 1$ such that U' is an integer number. Therefore, for each iteration the basic feasible solutions are 0 or 1, indicating that y variables can be relaxed in model H1. □

We note that the solutions provided by H1-MIP are feasible to M1, and thus our problem. In fact, by assuming disjoint paths, synchronization between work teams is not required anymore. However, this assumption may affect the quality of the solutions obtained by solving H1-MIP.

4.1.1. H1-MIP with multiple depots

In this section, we adapt H1 to address the generalization with multiple depots. That is, each of the work teams may start its route from a different depot. We also note that multiple work teams might be pre-positioned in the same depot. In order to handle this generalization, some properties associated with H1-MIP presented in Section 4.1 should be modified. Before giving the modified mathematical model, we should modify our multi-level network first. We define a depot node set, namely D, including all depots in the network. Moreover, the number of work teams pre-positioned in each depot should be given as a parameter. Since there are multiple depots, a copy of all depots should be put in each level except level 1.

Similar to the multi-level network in Figure 2, a copy of only the depot nodes exists in level $N + 1$. Figure 5 depicts two feasible solutions, where the depot set is $D = \{\alpha, \beta, \gamma\}$. Nodes 2, 4, 5 and 7 are critical, and nodes 1, 3 and 6 are non-critical nodes. Two work teams exist in this example, where the routes of the work teams are represented by narrow and thick black lines. In the upper graph, the work teams are pre-positioned in different depots (α and β), whereas in the lower graph, all work teams are pre-positioned in the same depot (α).

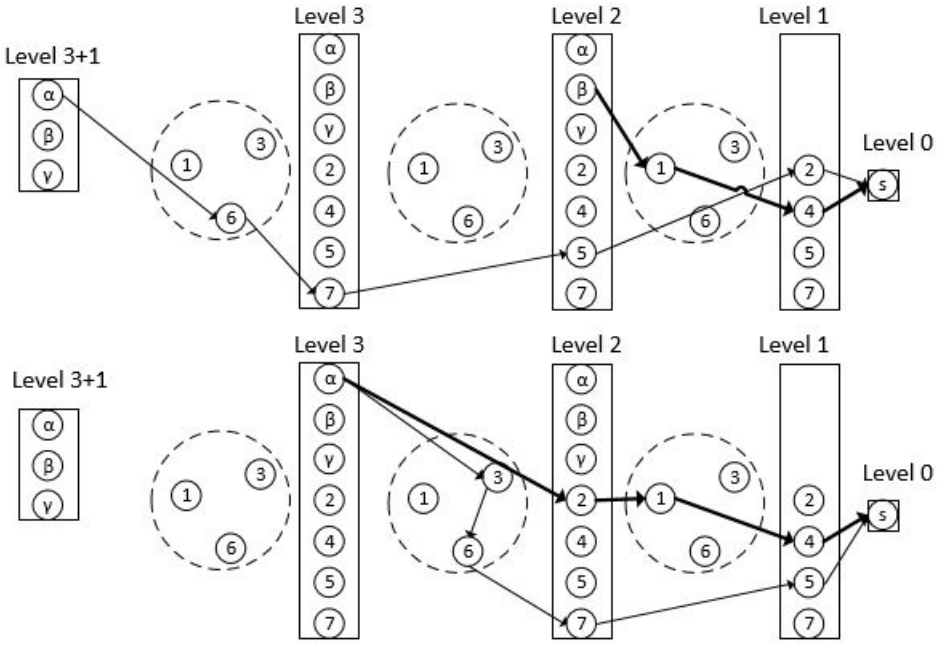


Figure 5: An example of the multi-level network structure with multiple depots

In the following, we present the necessary modifications to H1-MIP given in Section 4.1. First, we need to update the critical node set. That is, all of the depots should be added to the set V_C. Second, we replace index 0 with index $d \in D$ in all constraints given in Section 4.1. We define a parameter N_d, $d \in D$, which denotes the number of work teams pre-positioned in the depot node $d \in D$, where $\sum_{d \in D} N_d = m$ (m is the total number of work teams). Then, we eliminate Constraint (6) and instead add the following constraints to the mathematical model:

$$\sum_{r=1}^{N} \sum_{j \in (V \setminus V_C) \cup \{l\}, j \neq s} \sum_{l \in V_C \setminus D} y^r_{djdl} = N_d, \qquad d \in D \tag{20}$$

Constraints (20) set the number of work teams designated to each depot. By means of this new modified model, we can obtain feasible solutions to the MML-RCP instances having multiple depots.

4.2. A matheuristic for the MML-RCP (H2)

Given that in H1 the coordination of the work teams is prevented by assuming disjoint paths for the teams, in this section, we develop a matheuristic algorithm (H2) that allows the paths of the teams to have

common nodes and edges. In H2, we ensure that a blocked edge is opened only once and the teams can traverse an opened blocked edge without the need to recover it again. The H2 heuristic starts with a K-means clustering step for the assignment of critical nodes to work teams, followed by the resolutions of a mathematical model to determine the route of each work team separately. An improvement phase is also implemented. In order to obtain the corresponding objective function of the coordinated routes, we use the subroutine algorithm presented in section 4.2.2. The advantages of this heuristic are twofold; firstly, it generates coordinated routes that allows the teams to travel blocked edges without unblocking them once they are opened and its second advantage is its high speed in finding high quality solutions (within a 30-second time limit). The main reason for its high speed is that this heuristic approach is applied on a transformed graph of smaller size. We have conducted a number of computational experiments presented in Appendix B and observed that 30 seconds is in fact more than enough for H2 to find its best solutions. Since we solve a mathematical sub-problem for each work team, we refer to this algorithm as a matheuristic and denote it by H2.

We first form a complete graph, namely $G_C = (V_C, E_C)$, in which the node set consists of all critical nodes and the depot(s). We find the distance of the shortest path between each pair of critical nodes $i, j \in V_C$, namely SP_{ij}, in the original graph $G = (V, E)$ and set this value as the travel time of the edge (i, j) in the transformed graph. For calculation of this shortest paths, we set the traversal time of the intact edges $(E \setminus B)$ equal to t_{ij} and the traversal time of the blocked edges equal to $t_{ij}+u_{ij}$, which corresponds to the required time to traverse and unblock a blocked edge, $(i, j) \in B$. By doing so, the size of the network reduces considerably. Also, there is a possibility that some edges in the original graph exist in different shortest paths between critical nodes. As a result, H2 captures solutions that H1 may not. In this section, our default graph is set as the transformed graph.

4.2.1. Clustering: Assignment of critical nodes to work teams

We assign the critical nodes to work teams using the K-means clustering algorithm. K-means clustering algorithm, first introduced by Macqueen (1967), is a simple and popular method which is commonly used to partition a data set into m groups. It starts by choosing m initial cluster centers (centroids). Each node is assigned to its nearest centroid. Then, each centroid is updated to the mean of all nodes belonging to the same cluster. These steps are repeated until there is no further change in the cluster assignments, indicating the convergence of the algorithm. We use the the travel times in the reduced graph as distances in the algorithm. In addition, the initial centroids are selected randomly among the critical nodes. The number of clusters is set equal to the number of work teams. In our implementation, we used the K-means clustering module embedded in Scikit learn using Python.

4.2.2. The coordination subroutine

Once the assignment of nodes and the order in which they should be visited is known to the work teams, their routes should be coordinated to avoid over-estimating the objective function by reopening blocked edges that have been opened by other work teams. This is also important because the shortest path distances between critical nodes changes dynamically when blocked edges are opened. The steps of this coordination

subroutine are given in Algorithm 1.

The most important input of the coordination subroutine is the order of visiting critical nodes for each of the work teams. In this procedure, the first work team that arrives to a blocked edge unblocks it. If multiple work teams arrive to the same blocked edge simultaneously, without loss of generality, we assume that the team with the lower index unblocks it. The algorithm works based on identifying the next event for each of the work teams. These events are the arrival of the work teams to the end node of their last traversed edge. It could be either traversing intact edges, traversing a blocked edge that has been already opened or traversing and unblocking a blocked edge. In the initiation step of the algorithm, the first node on the path to the first critical node for each team is extracted. Then, in each step of the algorithm, the work team with the closest event is identified and the network is updated accordingly. The output of this subroutine is the coordinated latency of input order in which the order of visiting the critical nodes for each of the work teams is given.

4.2.3. Routing and improvement steps

After identifying which critical nodes will be visited by which teams, a second step is needed to find the order of visiting the critical nodes by each work team. Also, it is important to change the assignments designated to each work team to capture more diverse and better solutions. This can be done by a local search procedure consisting of inter-route moves. We use a mathematical model derived from Ajam et al. (2019), which has been designed to solve the single team problem, for each cluster. The model yields the optimal order of critical nodes to be visited by each work team. By doing so, we do not need to have intra-route moves for each work team. Furthermore, we keep a pool of currently investigated assignments so that we avoid recalculation of the total latency value for the same assignment. In other words, before applying the mathematical model to find the optimal orderings for these new assignments we check in the pool of already investigated assignments if they were generated before. As a result, we only need the inter-route moves between the work teams. The inter-route moves implemented in our algorithm are described below. Note that for each solution (that gives the order in which critical nodes should be visited for each work team), the coordination subroutine will be applied to coordinate the routes of the work teams such that additional unblocking times are removed if a blocked edge is traversed more than once.

1. **Swap** (N_1): A critical node is selected randomly. Then, one of the critical nodes assigned to another team is randomly chosen. The assignments of these nodes are swapped between the two work teams. First, the mathematical model is solved to find the best order whenever the assignments changes, then the coordination subroutine is applied to evaluate the corresponding latency of the new solution (the same procedure holds for the next moves as well). For instance, suppose critical nodes, given with their optimal orderings, $\{5, 4, 6\}$ and $\{9, 7, 8\}$ are assigned to work teams 1 and 2, respectively. We randomly pick node 5 from work team 1 and node 7 from work team 2 and exchange them. We check this new assignment in the pool of already investigated assignments if they were generated before. Then, we implement the mathematical model to find the optimal visiting order of the critical nodes with respect to the new assignments and finally once the optimal order of visiting the critical nodes is obtained, we implement the coordination subroutine to find the corresponding latency of the new

Algorithm 1 The coordination subroutine

Input: $G = (V, E), B, t_{ij}, u_{ij}, V_C, O_m, P_d$

▷ O_m : the order of visiting critical nodes for work team $m \in M$.

▷ P_d : the set of work teams initially positioned in the depot $d \in D$

$OFV = 0$ ▷ OFV : the objective function value of the coordinated routes.

$G' = (V, E \setminus B)$ ▷ G' : the post-disaster graph that will get updated with opening blocked edges.

$N = \emptyset$ ▷ N : set of visited critical nodes.

$L_m = d, m \in M$ ▷ L_m : position of work team $m \in M$.

$UT_{ij} = \infty, (i, j) \in B$ ▷ UT_{ij} : unblocking time of blocked edge (i, j).

$T_m = 0, m \in M$ ▷ T_m : time of the last event for work team m.

for $m = 1$ to $|M|$ **do**

  a_m : first node in $O_m \notin N$ ▷ a_m : first critical node in O_m that has not been visited.

  $b_m = SP(L_m, a_m, G')$

  ▷ b_m : first node on the shortest path from current position of work team m to a_m on G'.

  if $(L_m, b_m) \in B$ & $UT_{L_m,b_m} = \infty$ **then**

    $T_m = t_{L_m,b_m} + u_{L_m,b_m}$

    $UT_{L_m,b_m} = T_m$

  else if $(L_m, b_m) \in B$ & $UT_{L_m,b_m} < \infty$ **then**

    $T_m = UT_{L_m,b_m} + t_{L_m,b_m}$

  else

    $T_m = t_{L_m,b_m}$

  end if

end for

while $N \neq V_C$ **do**

  $\ell = \underbrace{\arg\min}_{m}\{T_m, \ \ m \in M\}$ ▷ ℓ : the work team that has the closest next event.

  if $(L_\ell, b_\ell) \in B$ **then**

    $B = B \setminus (L_\ell, b_\ell), \ \ G' = (V, E \setminus B)$

  end if

  if $b_\ell = a_\ell$ **then**

    $N = N \cup a_\ell, \ \ OFV = OFV + T_\ell$

    a_ℓ : first node in $O_\ell \notin N$,

    if a_ℓ does not exist **then**

      $T_\ell = \infty$

    else

      $L_\ell = b_\ell, \ \ b_\ell = SP(L_\ell, a_\ell, G')$

      if $(L_\ell, b_\ell) \in B$ & $UT_{L_\ell,b_\ell} = \infty$ **then**

        $T_\ell = T_\ell + t_{L_\ell,b_\ell} + u_{L_\ell,b_\ell}$

        $UT_{L_\ell,b_\ell} = T_\ell$

      else if $(L_\ell, b_\ell) \in B$ & $UT_{L_\ell,b_\ell} < \infty$ **then**

        $T_\ell = \max(T_\ell, UT_{L_\ell,b_\ell}) + t_{L_\ell,b_\ell}$

      else

        $T_\ell = T_\ell + t_{L_\ell,b_\ell}$

      end if

    end if

  end if

end while

return OFV

assignments and visiting orders. Thus, the new optimal orders for work teams 1 and 2 are $\{4, 7, 6\}$ and $\{8, 9, 5\}$, respectively.

2. **Random remove-insert** (N_2): We randomly select and remove a node from its path associated with a work team and add it randomly to a position in the path of another work team that is chosen randomly.

3. **Remove-insert from max to min latency** (N_3): We randomly select and remove one node from the path having maximum total latency value. Then, we add that node to the path of the work team that has the minimum total latency value.

4. **Double swap** (N_4): We randomly select 3 nodes from the paths of 3 work teams (we do not select more than one node from the same path). Then, they are removed from their paths and inserted back, again randomly with equal probabilities (the current positions are not possible for the removed nodes), in the empty positions. For instance, we select and remove nodes 5, 8 and 3 from the paths of $\{5, 4, 6\}$, $\{9, 7, 8\}$ and $\{1, 3, 2\}$ belonging to work teams 1, 2 and 3, respectively. Node 5 is moved to the path of work team 3. Then, node 8 is moved to the path of work team 1. Finally, node 3 is moved to the path of work team 2. The new optimal orders are found as $\{4, 8, 6\}$, $\{9, 3, 7\}$ and $\{1, 2, 5\}$.

Within a given time limit, these moves are repeated to find a better solution. We keep the order of the moves the same as presented. In Appendix B we provide computational experiments to show that this identified order is the most efficient order and all the identified and utilized neighborhood moves are effectively improving the solutions. Similar to the Variable Neighbourhood Structure (VNS) algorithm (Mladenović et al. 2013), if a move 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 move (N_1). In order to avoid a local optimum, we also add a perturbation procedure such that if there is no improvement in i_{max} consecutive iterations, we implement the double swap move N_{DS} consecutive times to avoid being trapped in a local optimal solution. In order to identify the efficient i_{max} and N_{DS} values we have conducted a number of computational experiments that are presented in Appendix B. Our results showed that setting the $i_{max} = 200$ and $N_{DS} = 5$ would yield the best solutions. Every time the perturbation step is implemented, we consider the incumbent solution obtained from the perturbation for the rest of the local search procedure. The outline of H2 is given in Algorithm 2, where i and k are the indices associated with the number of iterations and moves, respectively.

Note that H2 provides coordinated routes to MML-RCP as it extracts the solutions from the coordination subroutine in which blocked edges are only opened once and work teams can traverse blocked edges after they are recovered by only spending t_{ij} unit of time on them. We also note that the solution obtained from a road restoration model with a different objective, such as in Moreno et al. (2020), may be utilized as an initial solution, and then an improvement algorithm may be implemented starting with this solution.

4.3. A lower bound for MML-RCP

In this section, in order to evaluate the quality of the solutions provided with H1 and H2, we suggest a lower bounding approach which is based on makespan minimization. Recall that due to the intractability

Algorithm 2 The matheuristic (H2) associated with MML-RCP implemented on the transformed network (G_C)

Let $Coord(x)$ be the the latency of a solution x extracted from Algorithm 1.

Obtain x_G ▷ x_G : the initial solution obtained from the the K-means clustering algorithm and the mathematical model mentioned in this section.
$x_I = x_G$ ▷ x_I : the incumbent solution.
$x_B = x_G$ ▷ x_B : the best found solution.
while time limit is not over **do**
    $k \leftarrow 1$
    $i \leftarrow 1$
    while $i \leq i_{max}$ **do**
        local search: $x' \leftarrow N_k(x_I)$
        if $Coord(x') < Coord(x_I)$ **then**
            $x_I \leftarrow x'$
            $x_B \leftarrow x'$
            $i \leftarrow 1$
            $k \leftarrow 1$
        else
            $i \leftarrow i + 1$
            $k \leftarrow k + 1$ (if $k = k_{max}$ then, $k \leftarrow 1$)
        end if
    end while
    Perturb: $x_I \leftarrow x_I$
end while
return $Coord(x_B)$

of M1, we obtain very weak lower bounds from M1. In the proposed lower bounding approach we solve an optimization model that finds the minimum time required to visit a certain number of critical nodes with the given work teams. In other words, the objective of this MIP model is to minimize the total time until a given number of critical nodes is visited by m work teams. We call this MIP, which is solved consecutively with different input parameters, the Multiple Makespan Minimization MIP (3M-MIP). Note that contrary to the MML-RCP, 3M-MIP minimizes the latency of the last visited critical node. We give the 3M-MIP model in section C of the Appendix. In Algorithm 3, we present the outline of our lower bounding approach, where 3M-MIP(n, m) returns the objective function of the makespan minimization problem of visiting n critical nodes with m work teams. Let us call this lower bound obtained by this approach as the Consecutive Makespan Lower Bound (CMLB).

Algorithm 3 The lower bounding approach

Set $LB_m = 0$
for $n = 1, 2, \ldots, |V_C|$ **do**
    $LB_m = LB_m$ + 3M-MIP(n, m)
end for
return LB_m

Proposition 2. *CMLB is a lower bound for the total latency of the MML-RCP.*

Proof. Let $L_m^{[i]}$ be the latency of the critical node in position i (where m work teams exist). 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 MML-RCP) such that $LB_m^{[i]} \leq L_m^{[i]}\ \forall i = 1, 2, \ldots, |V_C|$. Thus, we can conclude that $\sum_{i=1}^{i=|V_C|} LB_m^{[i]} \leq \sum_{i=1}^{i=|V_C|} L_m^{[i]}$, indicating that CMLB is a lower bound to the objective function of MML-RCP. □

5. Data sets

In order to evaluate the performance of the proposed heuristic approaches, namely, the mathematical model based on the multi-layer network representation (H1-MIP) and the matheuristic (H2), that yield solutions with different characteristics, we used two different data sets. The first one is based on the Kartal region of Istanbul, Turkey, which is subject to major earthquake risk. This data set has been used previously in several related articles. In addition, we use random networks generated according to different characteristics to test different topologies.

5.1. Kartal data

The Kartal road network was first generated by Kilci et al. (2015) as a complete network with 45 nodes, together with real road distances, for a shelter location problem. A data set was derived from it by Sahin et al. (2016) for the road clearance problem, where the number of critical nodes was limited to seven. Later, Ajam et al. (2019) generated further instances with eleven and fifteen critical nodes. In this article, we use the Kartal network with 7, 11 and 15 critical nodes listed in Table 1. Additionally, for each instance, we run our algorithms with 2, 3, 4 and 5 work teams pre-positioned in the depot(s). Apart from the critical nodes, the data set consists of 20 instances (k1, k2,...,k20) in which the clearing times and the set of blocked edges differ. Given the severity of the earthquake (SOE), these 20 instances are categorized into 4 groups. For instance, SOE=1 refers to the least severe earthquake while SOE=4 is associated with the severest earthquake, causing more blocked edges with higher clearing times. Instances in each SOE group (e.g., k1, k2,...,k5) have the same number of blocked edges but their locations and the clearing times differ instance by instance. Table 2 describes the Kartal instances, stating the corresponding SOE and the number of blocked edges.

The traversal times are calculated based on the shortest path distances between the nodes over the road network. The clearing times are determined according to $u_{kl} = SOE * t_{kl} + U[0, max\ t_{ij}\ \forall (i, j) \in E]$, where a random number based on a uniform distribution is added.

Table 1: Selected critical nodes in the Kartal network

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

5.2. Random networks

We generate Euclidean networks with 20, 30, 40 and 50 nodes (640 instances in total, having up to 5 work teams), where in each instance the set of critical nodes, the traversal and the clearing times, and the

Table 2: Kartal instance names, the corresponding SOE and the number of blocked edges

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

number and location of blocked edges differ. The location of each node is defined by selecting coordinates randomly with equal probability in a plane. Then, the traversal time between each pair of nodes is calculated by the Euclidean distance. The clearing times are calculated according to: $u_{ij} = t_{ij}.U, \quad \forall (i,j) \in B$, where U has a uniform distribution between 1 and 20. Moreover, the percentage of the blocked edges vary from 5 to 60. Similar to the Kartal data sets, we test the model with 7, 11 and 15 critical nodes. These random data sets can be found in the website https://figshare.com/s/19531120f12b5a0f835f. Data consists of the set of nodes and critical nodes, the set of blocked edges with their clearing times, and the edge traversal times for each instance.

6. Computational results

This section examines the performance of H1 and H2 presented in Sections 4.1 and 4.2, respectively. We report the objective function values of the solutions, the gap of the solutions with respect to the lower bounds (CMLB) obtained by the procedure described in Section 4.3, and the run times associated with the Kartal data and the random network instances. We run H2 ten times over each of the instances to take advantage of its randomness. In both data sets, we test each instance with up to 5 work teams. We also examine cases having multiple depots.

We coded our algorithms with Python 2.7.12, solved the models using Gurobi 8.0, and ran them on a computer with 32 GB RAM and two Intel Xeon E5-2643 CPU @ 3.30 gigahertz processors, under the Windows 7 operating system.

6.1. Kartal data results

In this section, we report the results of the tests with Kartal data in Tables 3 for 7 critical nodes, 4 and 5 for the cases with 11 and 15 critical nodes, respectively, where we run H1 and H2 with 7, 11 and 15 critical nodes, for each of the 20 instances. Moreover, we conduct experiments by varying the number of work teams. For 7, 11 and 15 critical nodes, we employ up to 5 work teams. In our tables, in the columns denoted by H2, we give the average objective function value from the ten repetitions of the algorithm and denote it by "Avg", and also show the best solution among the ten repetitions by "Best". In the columns denoted by H1 we give the objective function value obtained from this algorithm by "H1" and the CPU run time in second by "CRT". In the column denoted by "LB", we report the lower bound obtained from our lower bounding algorithm given in section 4.3. We also report the optimality gaps using the best solution between H1 and H2. Denoting the best solution by OFV, we then calculate the gaps using $Gap = \frac{OFV-LB}{OFV} \times 100$ formula.

Tables 3, 4 and 5 represent the results of testing H1 and H2 algorithms on the Kartal instances with 7, 11 and 15 critical nodes, respectively. Recall that in H1-MIP, we have not defined an index associated with the work teams. Moreover, increasing the number of work teams decreases the total number of critical nodes designated to each work team. As a result, as the number of work teams increases, the average computational time of H1-MIP decreases in most of the instances (i.e., Average CRT decreases from 491.1 to 142 seconds when the number of work teams changes from 2 to 4 in Table 5). We can take advantage of this property if a high number of work teams are available. In other words, having more work teams not only decreases the total latency, but also decreases the computational times in most of the instances.

As we see in Tables 3, 4 and 5, H1 and H2 results are close to each other when 2 work teams are employed. However, when the number of work teams increases, H2 provides better solutions than those in H1. Due to an increase in the number of work teams, the chance of traversing an edge by different work teams increases. Therefore, H1 misses better solutions because of the disjoint-path assumption.

Recall that the SOE value increases for every 5 instance in Kartal data from 1 to 4; thus, increasing the number of blocked edges. Correspondingly, we observe that an increase in SOE value in Kartal data with 15 critical nodes and 2 work teams increases computational times of some instances (as instance numbers increase) associated with H1 model. In addition, the locations of the blocked edges can change the computational times. For instance, in Table 2, although the instances from 16 to 20 have 806 blocked edges, their computational times are different for any number of critical nodes that we have tested for the Kartal data.

Table 3: Results of Kartal data sets for 7 critical nodes

	m = 2						m = 3					
	H2		H1		LB	Gap	H2		H1		LB	Gap
Instance	Avg	Best	H1	CRT			Avg	Best	H1	CRT		
1	100	100	100	12.7	84	16%	85	85	85	5.6	78	8%
2	98	98	98	11.2	81	17%	82	82	82	5.6	78	5%
3	99	99	99	10.3	83	16%	84	84	84	5.9	79	6%
4	98	98	98	12.5	81	17%	82	82	82	5.5	77	6%
5	98	98	98	12	81	17%	82	82	82	5.4	77	6%
6	104	104	104	11.2	89	14%	87	87	87	5.7	77	11%
7	105	105	107	6.7	90	14%	87	87	87	5	79	9%
8	113	113	113	5.9	102	10%	103	103	103	5.2	93	10%
9	108	108	108	6.4	99	8%	94	94	94	5.5	86	9%
10	104	104	116	19.3	93	11%	91	91	95	5.5	82	10%
11	130	130	134	10.8	115	12%	110	110	110	5.1	100	9%
12	134	134	134	6.7	127	5%	115	115	115	5.5	103	10%
13	121.6	121	121	8.4	114	6%	103	103	103	8.7	92	11%
14	117	117	117	8.8	103	12%	103	103	103	5.2	95	8%
15	105	105	105	6.4	90	14%	91	91	91	5.4	82	10%
16	281	281	292	18.5	272	3%	240	240	250	8.7	234	3%
17	198	198	208	10.8	197	1%	173.1	173	186	11.1	166	4%
18	243	243	243	6.2	236	3%	205.4	205	207	5.8	193	6%
19	238	238	228	16.3	214	6%	188	188	190	11.4	179	5%
20	196	196	186	6.2	171	8%	150	149	149	5	141	5%
Average	139.5	139.5	140.5	10.4	126.1	11%	117.8	117.7	119.3	6.3	109.6	8%

Table 4: Results of Kartal data sets for 11 critical nodes

	m = 2						m = 3						m = 4					
	H2		H1		LB	Gap	H2		H1		LB	Gap	H2		H1		LB	Gap
Instance	Avg	Best	H1	CRT			Avg	Best	H1	CRT			Avg	Best	H1	CRT		
1	178	178	178	24.5	166	7%	150	150	150	20.7	135	10%	138	138	138	15.9	122	12%
2	174	174	174	24.7	163	6%	140	140	140	19.7	132	6%	133.4	132	132	21.8	122	8%
3	172	172	172	26.3	163	5%	146	146	146	20.9	135	8%	136	136	136	15.9	123	10%
4	173	173	173	32.7	161	7%	139	139	139	19.5	131	6%	131.5	131	131	26.8	121	8%
5	173	173	173	34.8	161	7%	139	139	139	19.5	131	6%	131.6	131	131	26.4	121	8%
6	181	181	181	21.9	174	4%	148	148	148	18.7	136	8%	135	135	135	16.8	123	9%
7	211.8	211	211	23.8	194	8%	171.6	171	171	18.7	152	11%	151	151	151	16.3	137	9%
8	201	201	209	44.2	192	4%	168.4	168	171	32.9	155	8%	153.4	153	153	16.4	140	8%
9	205.4	205	205	78.9	193	6%	166.1	164	164	21.1	154	6%	147	147	163	16	136	7%
10	194	194	204	23.9	182	6%	155	155	161	18.7	148	5%	141	141	147	17.2	132	6%
11	245	245	240	60.5	223	7%	192.3	192	192	19.7	174	9%	170	170	170	16.5	153	10%
12	249	249	249	23.9	239	4%	193.2	192	192	19.5	180	6%	165	165	165	16.6	159	4%
13	242.2	238	240	63.3	223	6%	183	183	183	31.3	172	6%	158.4	158	160	26.8	146	8%
14	214	214	212	68.9	199	6%	181.9	180	180	29.1	162	10%	164	164	164	17.1	151	8%
15	200	195	197	33.1	181	7%	159.8	159	159	19.8	146	8%	145.3	145	145	17	131	10%
16	464.8	456	469	56.3	439	4%	390.4	386	409	69.3	370	4%	351.8	350	375	44.3	337	4%
17	322	322	322	31.9	311	3%	270.4	267	274	71.5	251	6%	238.8	237	250	23.8	224	5%
18	417	417	441	69.1	398	5%	328	328	353	53.9	311	5%	289.4	287	309	16.7	273	5%
19	321	318	337	74.7	304	4%	254.7	254	257	62.6	233	8%	223	222	222	17.1	208	6%
20	334	334	334	32.9	311	7%	257.5	257	257	18.5	240	7%	222.2	222	222	15.6	206	7%
Average	243.6	242.5	246.1	42.5	228.9	6%	196.7	195.9	199.3	30.3	182.4	7%	176.3	175.8	180.0	20.1	163.3	8%

Table 5: Results of Kartal data sets for 15 critical nodes

	m = 2						m = 3						m = 4						m = 5					
	H2		H1		LB	Gap	H2		H1		LB	Gap	H2		H1		LB	Gap	H2		H1		LB	Gap
Instance	Avg	Best	H1	CRT			Avg	Best	H1	CRT			Avg	Best	H1	CRT			Avg	Best	H1	CRT		
1	332	332	327	616.9	293	10%	253.8	253	253	193.8	229	9%	224.4	223	223	149.8	199	11%	208	204	204	95.7	187	8%
2	342	333	332	633.9	297	11%	257.3	251	251	167.6	232	8%	225	225	222	249.4	202	9%	206.2	206	206	85.9	188	9%
3	332.9	323	323	513.4	294	9%	255.6	252	252	204.8	234	7%	229.1	229	227	105.6	203	11%	210	210	210	127.5	190	10%
4	351	337	337	464	307	9%	262.8	256	256	137.9	236	8%	225.2	220	220	76.2	204	7%	204	204	204	59.8	191	6%
5	353.2	344	337	465.2	307	9%	262.8	256	256	141.8	236	8%	228	220	220	76.2	204	7%	204.8	204	204	60.3	191	6%
6	386	386	366	222.9	343	6%	278.4	278	278	112.1	257	8%	239.2	237	237	56.8	220	7%	219.2	218	218	49.9	199	9%
7	366	366	366	311.2	339	7%	289	289	289	305.4	254	12%	250.9	250	250	116.2	224	10%	233.9	232	232	164	206	11%
8	364	364	370	481.8	338	7%	292	291	284	342.8	261	8%	243	243	243	111.2	229	6%	226.6	225	225	77.8	212	6%
9	385	385	385	638.8	353	8%	284	284	284	156	269	5%	248.7	248	249	176.1	233	6%	233	233	224	82.4	211	6%
10	363	363	377	745	335	8%	280.3	280	286	283.8	252	10%	241.5	237	244	191.4	219	8%	218	218	224	72.5	202	7%
11	412	412	412	512	386	6%	329.8	309	309	199.6	289	6%	272.2	271	269	133.3	246	9%	246.6	246	246	83.7	224	9%
12	454.2	452	452	236.9	433	4%	332	332	332	176.9	319	4%	291.8	291	285	219.2	267	6%	249	249	249	49.3	239	4%
13	418	418	419	671.4	385	8%	316.7	308	308	270.8	286	7%	265.5	265	265	138.6	246	7%	239.5	237	239	94.8	221	7%
14	364	364	364	456.3	347	5%	301.9	297	287	115.4	269	6%	266.7	261	261	75.2	244	7%	250.2	249	248	169.9	229	8%
15	380	380	358	443.4	326	9%	279.2	276	276	192	254	8%	244.8	244	244	207.5	219	10%	224.9	224	224	92.5	204	9%
16	708.7	707	691	402	669	3%	575.4	565	577	348.9	541	4%	515.5	510	516	175.3	488	4%	485.8	483	494	67.7	457	5%
17	637	637	637	207.7	597	6%	508.3	498	502	326.7	464	7%	429.8	420	443	222.5	397	5%	390.7	390	413	152.3	365	6%
18	693	693	716	791.2	638	8%	521.8	517	547	176.2	481	7%	447.8	447	478	144	416	7%	407.5	406	438	182.8	381	6%
19	599.2	591	621	593.4	561	5%	430.1	428	444	202.6	407	5%	365.4	365	371	130.3	347	5%	329.8	329	331	49.5	314	5%
20	735	735	684	414.9	626	8%	485.8	482	493	111.7	455	6%	403.8	400	407	84.5	379	5%	345.8	342	349	46.5	327	4%
Average	448.8	446.1	443.7	491.1	408.7	7%	339.9	335.1	338.2	208.3	311.3	7%	292.9	290.3	293.7	142.0	269.3	7%	266.7	265.5	269.1	93.2	246.9	7%

6.2. Kartal data results with multiple depots

We have already described in Section 4.1.1 how to handle the problem when the work teams are pre-positioned in more than one depot node. In this section, we report results of the experiments with multiple depots, implemented by modified H1. Figure 6 presents the results of Kartal data with 15 critical nodes, where other than depot node 16, nodes 45 and 20 are added to the depot set (D). We assume that 2, 1 and 1 work teams are pre-positioned in depot nodes 16, 45 and 20, respectively, where 4 work teams are available in total. In addition, when 5 work teams are available, we assign 2, 2 and 1 work teams to depot nodes 16, 45 and 20, respectively. In each part of Figure 6, two data sets for the single depot versus multiple depot cases are given. The total latency and computational times with 4 and 5 work teams is given in these figures. 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.), which increases the computational times. Note that also the locations of the depot nodes may affect 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 be obtained in higher time. As we see in Figure 6, although the average computational times of instances with multiple depots are higher than the computational times of instances with a single depot, we notice a decrease in the objective value over all instances, indicating that using multiple depots besides multiple work teams may yield better solutions.

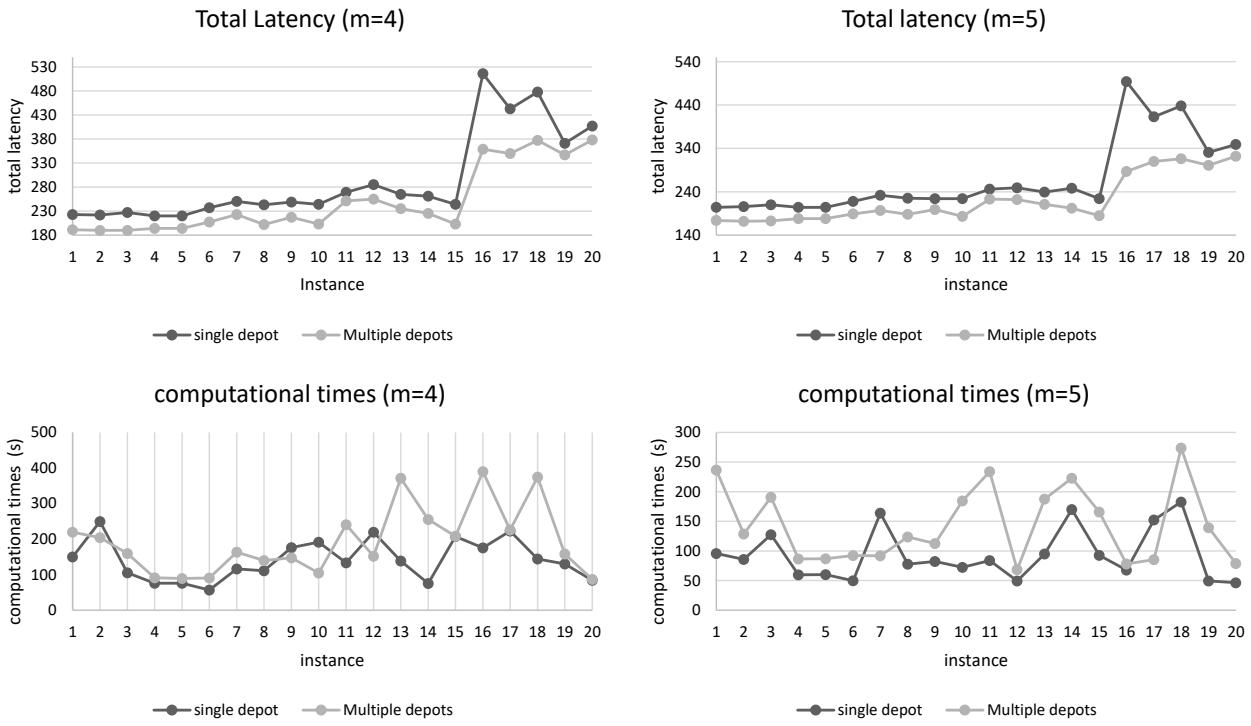


Figure 6: Comparison of multiple and single depot cases

6.3. Random data results

To investigate the performance of H1 and H2 further on other networks, we carry out experiments with random data sets (640 instances in total) where the locations and number of the depot(s), critical nodes, blocked edges, and traversal and clearing times differ in each instance. For all the instances, H2 was tested for ten repetitions and the reported results are the average of those ten repetitions. In our computational experiments, we set the number of critical nodes similar to the Kartal data. Nevertheless, we vary the number of total nodes as 20, 30, 40 and 50. In Tables 6, we give the results of testing both H1 on H2 on instances with 7 critical nodes. In Tables 7 and 8, we give the optimality gaps obtained for the instances with 11 and 15 critical nodes, respectively and in Figures 7 and 8, we compare the performance of the H1 and H2 on the instances with 11 and 15 critical nodes, respectively. Using this comparison, we highlight the importance of incorporating coordination operations. In these data sets, although we use various values for

the parameters (number of blocked edges and clearing times), the average computational time of H1 remains under 1 hour. For example, in the largest instances having 15 critical nodes and 50 nodes in the graph, the average computational time (with 2 work teams) is around 1000 seconds, showing the capability of H1 on the data sets generated randomly. We note that once the number of teams increases, or the number of critical nodes decreases, the CPU run time of H1 also decreases. Given this observation, we have not reported the CPU run time of the algorithms and instead, our main consideration is towards comparison of the results achieved from the H1 and H2 algorithms and presenting the quality of the obtained solutions through the optimality gaps.

The result of the random data sets with 7 critical nodes is given in Table 6. In this table, the columns denoted by D, give the percentage difference between obtained solutions using H1 and H2 for each scenario with certain number of teams (m) and nodes (n). If we denote the solution obtained from H1 by Z_{H1} and the solution obtained from H2 by Z_{H2}, then column D is calculated as $D = \frac{Z_{H1} - Z_{H2}}{Z_{H1}}$. The Gap columns are calculated using the same method as explained for Table 3. The average row at the bottom of the table, gives the average of the D and Gap values over the 20 tested instances.

As seen in Table 6, the average results of the H1 and H2 algorithms are close to each other in most of the instances in the presence of 7 critical nodes with 2 and 3 work teams. Here, we point out that if D is positive over an instance, then H2 was able to find a better solution compared to H1 and if this value is negative, then H1 was able to find a better solution. As it can be observed, in most of the instances H1 was not able to find a better solution and by considering coordination in the operations, we were able to find much better solutions in some of the instances like instance 8 with 3 teams and 30 nodes were consideration of coordination enabled us to improve the solutions by more than 37%.

Table 6: Results of random data sets with 7 critical nodes

Instance	m=2								m=3							
	n=20		n=30		n=40		n=50		n=20		n=30		n=40		n=50	
	D	Gap	D	Gap	D	Gap	D	Gap	D	Gap	D	Gap	D	Gap	D	Gap
1	0.00	0.46	0.00	2.59	0.00	3.49	0.00	0.00	0.00	0.49	0.00	0.63	0.00	0.01	0.00	4.73
2	0.00	0.00	0.00	0.28	0.00	1.38	0.00	2.52	0.00	2.88	0.00	0.35	0.00	0.11	0.00	0.41
3	0.00	2.22	0.00	1.62	0.76	11.82	-1.70	0.72	0.86	2.30	0.00	8.25	0.00	12.07	-2.02	5.04
4	0.00	5.82	0.00	0.01	0.00	9.28	0.00	4.49	0.00	8.02	0.40	2.01	0.00	0.26	0.00	1.32
5	0.00	7.00	0.00	0.99	0.00	4.46	0.00	6.83	0.00	6.35	0.00	1.47	0.00	4.89	4.07	8.54
6	0.00	2.26	0.00	3.65	0.00	8.37	5.25	5.48	0.00	0.00	0.00	4.81	0.00	2.94	5.70	0.64
7	0.00	15.83	0.00	10.79	0.00	0.87	0.00	12.25	0.00	0.00	5.17	3.07	0.00	3.55	0.00	10.25
8	0.00	0.83	0.00	0.00	0.00	5.84	0.03	3.33	0.00	8.63	37.96	0.01	0.00	8.08	0.00	2.84
9	0.00	0.00	0.00	3.65	0.21	1.06	3.41	4.05	0.00	0.00	16.23	2.54	0.00	0.45	0.94	0.00
10	0.00	12.89	0.00	8.57	0.00	8.97	0.00	2.23	0.00	13.58	0.00	0.00	0.00	9.90	0.00	6.77
11	0.00	3.10	0.00	2.59	0.00	2.36	0.00	4.46	0.00	3.52	0.00	4.92	0.00	8.38	0.00	3.21
12	0.00	4.28	0.00	2.66	0.20	1.94	0.00	0.50	0.00	3.86	0.00	10.22	0.23	1.42	0.00	2.39
13	0.00	4.09	0.00	2.27	0.00	3.30	0.00	2.05	0.00	2.59	0.00	0.00	0.00	2.53	0.00	6.11
14	0.00	0.00	1.30	12.93	0.00	0.54	0.00	0.98	0.00	6.17	0.00	0.03	0.00	0.36	0.00	3.22
15	0.00	0.01	0.00	0.42	0.00	8.28	0.00	0.00	2.85	0.00	0.00	5.70	0.00	0.00	0.00	1.32
16	0.00	0.00	0.00	4.24	0.00	3.48	0.00	1.17	0.00	0.00	0.00	0.88	0.00	3.07	0.00	1.31
17	0.00	0.00	0.00	3.82	0.00	0.00	0.00	5.76	0.08	0.00	0.00	6.67	0.00	0.00	0.00	3.02
18	0.00	2.71	0.00	5.92	0.00	1.14	0.00	2.21	0.00	0.50	0.00	1.44	0.00	0.75	0.00	7.10
19	0.00	2.70	3.81	3.63	0.00	4.03	0.00	2.67	0.00	2.85	0.00	1.97	0.00	4.51	0.00	2.54
20	2.98	0.00	0.00	0.00	0.00	4.52	0.00	0.22	0.00	1.90	2.13	0.00	0.00	5.65	0.00	2.54
Average	0.15%	3.21%	0.26%	3.53%	0.06%	4.26%	0.35%	3.10%	0.19%	3.18%	3.09%	2.75%	0.01%	3.45%	0.43%	3.66%

The results of the random data sets with 11 critical nodes are presented in Table 7 and Figure 7, where the obtained optimality gaps and a comparison between the H1 and H2 algorithms are presented, respectively. As it can be observed in Table 7, the average gap remained under 6.75% and this gap decreases once the

number of work teams increases from 2 to 4. For example the average gap over all the instances with 2 teams is 6.05% and it decreases to 4.06% and 3.92% for the instances with 3 and 4 work teams, respectively.

Table 7: Obtained gaps for the random data sets with 11 critical nodes

	m=2				m=3				m=4			
Instance	n=20	n=30	n=40	n=50	n=20	n=30	n=40	n=50	n=20	n=30	n=40	n=50
1	0.08	0.10	0.05	0.10	0.00	0.07	0.03	0.09	0.03	0.02	0.05	0.11
2	0.03	0.01	0.03	0.04	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.01
3	0.02	0.05	0.07	0.07	0.01	0.06	0.02	0.03	0.00	0.02	0.03	0.07
4	0.08	0.12	0.02	0.04	0.01	0.03	0.03	0.01	0.02	0.02	0.04	0.04
5	0.04	0.01	0.06	0.04	0.02	0.00	0.04	0.01	0.05	0.01	0.06	0.07
6	0.01	0.05	0.01	0.07	0.07	0.03	0.00	0.08	0.06	0.05	0.01	0.08
7	0.09	0.07	0.01	0.05	0.09	0.01	0.00	0.07	0.01	0.07	0.04	0.03
8	0.12	0.03	0.02	0.09	0.01	0.03	0.06	0.10	0.12	0.04	0.02	0.07
9	0.06	0.05	0.05	0.07	0.05	0.00	0.06	0.07	0.05	0.01	0.01	0.04
10	0.05	0.14	0.04	0.06	0.05	0.07	0.08	0.02	0.03	0.07	0.03	0.02
11	0.10	0.06	0.07	0.00	0.07	0.10	0.06	0.00	0.06	0.01	0.06	0.05
12	0.07	0.00	0.08	0.08	0.11	0.05	0.10	0.03	0.06	0.02	0.10	0.02
13	0.09	0.04	0.05	0.01	0.02	0.06	0.02	0.02	0.01	0.02	0.01	0.02
14	0.07	0.05	0.12	0.04	0.13	0.04	0.01	0.05	0.03	0.02	0.08	0.02
15	0.05	0.07	0.05	0.06	0.00	0.01	0.06	0.01	0.01	0.04	0.03	0.04
16	0.08	0.08	0.07	0.02	0.01	0.06	0.05	0.02	0.15	0.04	0.05	0.04
17	0.12	0.03	0.04	0.06	0.06	0.01	0.01	0.02	0.01	0.02	0.07	0.03
18	0.03	0.13	0.11	0.06	0.02	0.09	0.06	0.03	0.01	0.02	0.11	0.03
19	0.02	0.07	0.02	0.05	0.01	0.01	0.00	0.01	0.01	0.07	0.04	0.01
20	0.03	0.19	0.19	0.08	0.04	0.10	0.23	0.01	0.03	0.09	0.08	0.05
Average	6.13%	6.75%	5.83%	5.50%	3.94%	4.28%	4.66%	3.37%	3.74%	3.30%	4.53%	4.13%

Figure 7 gives the results of the comparisons between H1 and H2 based on the number of nodes and the number of work teams. Overall, we can observe that incorporating coordination can improve the solutions significantly. This observation is more significant when the number of work teams increases. For example, for the case with two work teams, over all the different values of n, the average coordination improvement is 0.81% but this values increases to 6.17% for the case with three work teams and 8.37% for the case with 4 work crews respectively. This could be expected since using more work teams provides more opportunities for coordination among them. Moreover, H1 was only able to find a better solution compared to H2 in one of the instances with four teams and 50 nodes and this improvement was only for 0.6%. For instances with three teams however, H1 was not able to find a better solution compared to H2 in none of the tested instances. In the case with two teams, H1 found a better solution compared to H2 for only one instance with 20 nodes and 2 instances with 50 nodes.

The results of the random data sets with 15 critical nodes are presented in Table 8 and Figure 8. Table 8 gives the obtained optimality gaps over different instances with different number of work teams ($m = 2, 3, 4, 5$) and number of nodes ($n = 30, 40, 50$). As it can be observed over these instances, the average gap decreases when the number of work teams increases. For the case with two work teams, the average gap is 6.74, for the case with three teams the average gap is 6.50% and this gap decreases to 3.95% and 3.41% for the cases with four and five work teams respectively. This is a similar trend as to what we have observed for the random instances with 11 critical nodes.

Figure 8 provides the comparison between H1 and H2 algorithms for the random instances with 15

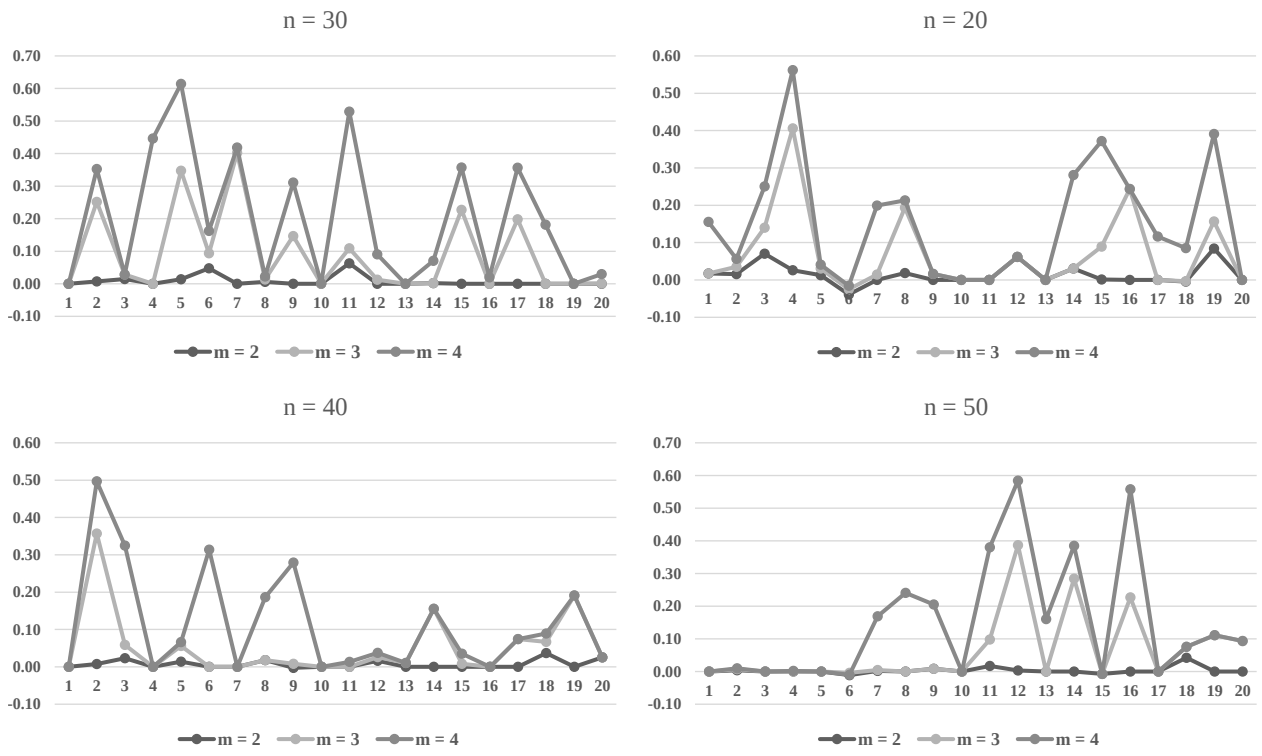


Figure 7: Comparison of H1 and H2 for random instances with 11 critical nodes

Table 8: Obtained gaps for the random data sets with 15 critical nodes

	m=2			m=3			m=4			m=5		
Instance	n=30	n=40	n=50	n=30	n=40	n=50	n=30	n=40	n=50	n=30	n=40	n=50
1	0.07	0.03	0.04	0.08	0.03	0.04	0.04	0.14	0.02	0.02	0.03	0.02
2	0.12	0.04	0.14	0.09	0.03	0.07	0.03	0.08	0.02	0.03	0.05	0.07
3	0.06	0.04	0.08	0.05	0.06	0.08	0.01	0.04	0.03	0.02	0.04	0.07
4	0.11	0.05	0.03	0.12	0.05	0.04	0.01	0.01	0.06	0.03	0.03	0.01
5	0.04	0.09	0.07	0.06	0.08	0.07	0.03	0.07	0.04	0.06	0.02	0.07
6	0.09	0.09	0.04	0.06	0.09	0.04	0.08	0.04	0.01	0.01	0.03	0.01
7	0.06	0.05	0.04	0.01	0.05	0.07	0.02	0.04	0.05	0.02	0.06	0.03
8	0.06	0.08	0.10	0.08	0.09	0.10	0.07	0.01	0.02	0.00	0.02	0.05
9	0.09	0.09	0.00	0.06	0.06	0.01	0.03	0.04	0.04	0.02	0.03	0.02
10	0.10	0.10	0.07	0.06	0.08	0.06	0.05	0.08	0.09	0.03	0.06	0.04
11	0.04	0.07	0.03	0.06	0.05	0.05	0.01	0.03	0.03	0.03	0.06	0.01
12	0.09	0.07	0.09	0.14	0.07	0.11	0.01	0.03	0.03	0.08	0.02	0.02
13	0.08	0.02	0.09	0.07	0.04	0.12	0.02	0.02	0.06	0.02	0.04	0.01
14	0.07	0.08	0.10	0.07	0.11	0.09	0.06	0.09	0.08	0.03	0.04	0.04
15	0.07	0.05	0.12	0.06	0.07	0.09	0.01	0.01	0.03	0.02	0.03	0.02
16	0.09	0.01	0.02	0.08	0.04	0.01	0.03	0.03	0.01	0.03	0.02	0.03
17	0.04	0.11	0.06	0.07	0.07	0.05	0.05	0.04	0.03	0.03	0.03	0.08
18	0.13	0.07	0.03	0.09	0.08	0.03	0.04	0.03	0.06	0.06	0.03	0.02
19	0.06	0.05	0.03	0.04	0.08	0.02	0.03	0.00	0.03	0.06	0.03	0.05
20	0.10	0.05	0.03	0.06	0.06	0.06	0.02	0.09	0.06	0.02	0.05	0.03
Average	7.95%	6.20%	6.08%	6.99%	6.44%	6.07%	3.18%	4.66%	4.00%	3.10%	3.57%	3.58%

critical nodes. This figure is divided to four parts based on the number of work teams. As it can be observed when the number of work teams increases the impact of the coordination considerations becomes more significant. In the instances with two work teams the coordination makes an improvement of 0.54% but this

value increased to 1.05%, 13.54% and 14.50% for the instances with three, four and five teams, respectively. The coordination consideration was able to improve the solutions by up to 30% for instance 3 with four teams and 40 nodes. H1 was only able to provide better solutions in a number of instances with two work teams and only one instance with three teams. However, in all the instances with four and five teams, H2 found a better solution compared to H1 in all the tested instances.

Given the observations associated with the random network data, number of work teams, number of critical nodes, number of total nodes can affect the solutions provided by H1 and H2. For small networks with up to 3 work teams and a few critical nodes (up to 7 critical nodes), both H1 and H2 yield close solutions. However, when the number of work teams increases, the possibility of traversing an edge with different work teams increases as well. Thus, H1 cannot provide high-quality solutions (given its disjoint-path assumption) compared to those in H2.

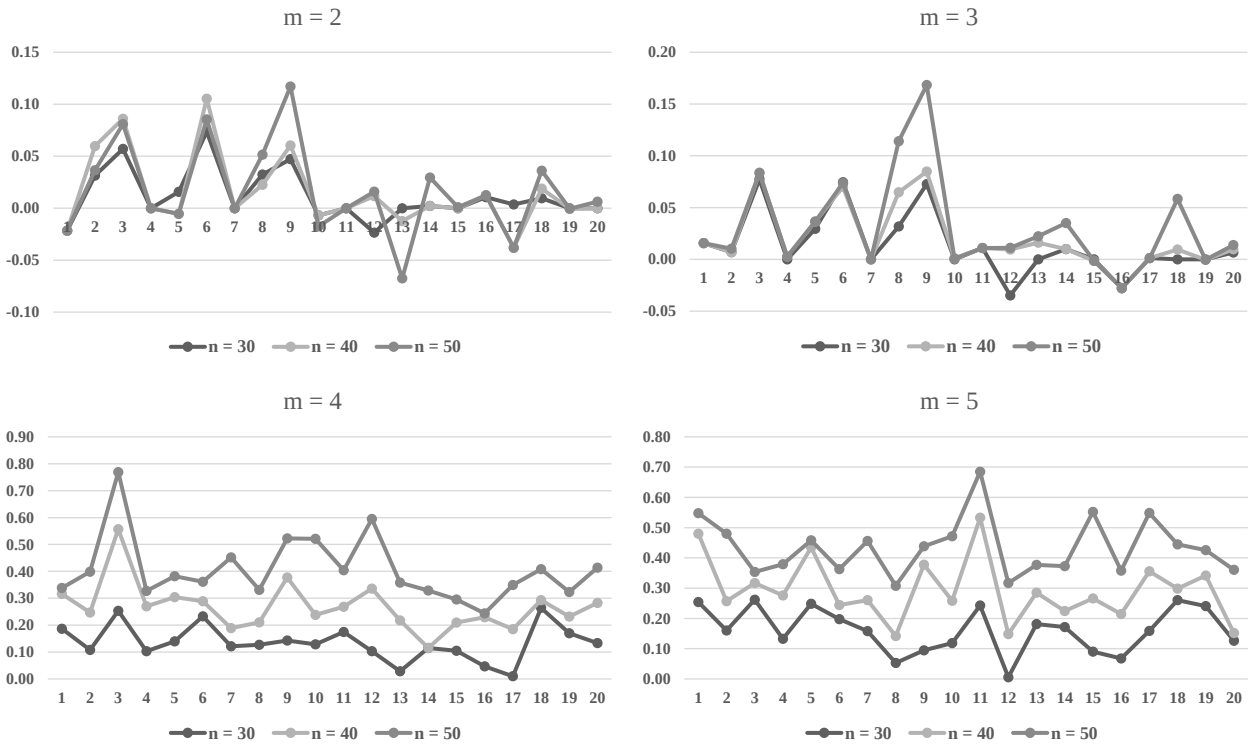


Figure 8: Comparison of H1 and H2 for random instances with 15 critical nodes

7. Conclusions

In this article, we studied the road clearance/restoration problem in disaster management with multiple work teams, with the objective of minimizing total latency of reaching a given set of critical nodes for the first time. We developed a mathematical model (M1) that handles the complicating synchronization requirement among the work teams. However, this model is able to solve instances with a limited size (having up to 12 nodes). Thus, we proposed two alternative heuristic approaches. We introduced a novel multi-level network structure to solve which problem as a heuristic approach through which multiple work teams positioned at

multiple depots can be handled as well as visiting a subset of nodes. Based on the novel multi-level network, we proposed a mathematical model (H1-MIP) that is able to solve instances up to 45 nodes and 15 critical nodes in short computational times. In addition, we introduced a matheurisitc (H2) which has different properties from the ones in H1, namely the disjoint paths requirement has been relaxed in H2. H2 works on a transformed network having the set of critical nodes as the set of nodes. H2 is designed based on a coordinated subroutine in which if a blocked edge is unblocked by a work team, it will be available for the next traverses by the same or the other work teams. In small networks with limited number of critical nodes and work teams both H1 and H2 find solutions close to each other. However, we recommend to use H2 when the number of work teams and critical nodes are large.

We conducted experiments with two types of data sets to compare the two heuristics and observe optimality gaps with respect to the lower bounds obtained by the proposed procedure. The first one is based on a real data of a district in Istanbul that is prone to earthquake risk and the second one is generated randomly with different problem sizes. We observed reasonable computational times with H1. The advantage of this model is that since we have intentionally not defined an index associated with work teams, computational times decrease when the number of work teams increases, since the number of critical nodes designated to each work team decreases. On the other hand, H2 is faster since it is implemented on a reduced graph. Also, the paths it generates are not necessarily disjoint, which may enable obtaining smaller objective values. We note that having small computational time is extremely important for implementing solutions in the aftermath of a disaster in the immediate response phase.

We believe that due to the tractability of both H1 and H2, organizations in charge of disaster response can utilize these methodologies for a more effective response operation. Analyses with what-if questions to guide the decision makers would also be possible due to short computational times of the developed methodologies. As a future work, heterogeneous work teams with different speeds for clearing blocked edges can be considered. Incorporating the uncertainty in road clearing/restoration times is another direction for future work.

References

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., Sadati, M. E. H., and Kian, R. (2021a). A decomposition-based heuristic for a multicrew coordinated road restoration problem. *Transportation Research Part D: Transport and Environment*, 95:102854.

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.

Akbari, V., Shiri, D., and Salman, F. S. (2021b). An online optimization approach to post-disaster road restoration. *Transportation Research Part B: Methodological*, 150:1–25.

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.

Avci, M. and Avci, M. G. (2017). A grasp with iterated local search for the traveling repairman problem with profits. *Computers & Industrial Engineering*, 113:323–332.

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.

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.

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.

Goemans, M. and Kleinberg, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming, Series B*, 82(1-2):111–124.

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.

Kim, S., Shin, Y., Lee, G. M., and Moon, I. (2018). Network repair crew scheduling for short-term disasters. *Applied Mathematical Modelling*, 64:510–523.

Lalla-Ruiz, E. and Voß, S. (2020). A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem. *Optimization Letters*, 14(3):671–691.

Li, S., Ma, Z., and Teo, K. L. (2020). A new model for road network repair after natural disasters: Integrating logistics support scheduling with repair crew scheduling and routing activities. *Computers & Industrial Engineering*, 145:106506.

Li, S. and Teo, K. L. (2019). Post-disaster multi-period road network repair: work scheduling and relief logistics optimization. *Annals of Operations Research*, 283(1):1345–1385.

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.

Lysgaard, J. and Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research*, 236(3):800–810.

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

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., Urošević, D., and Hanafi, S. (2013). Variable neighborhood search for the travelling deliveryman problem. *4OR*, 11(1):57–73.

Moreno, A., Alem, D., Gendreau, M., and Munari, P. (2020). The heterogeneous multicrew scheduling and routing problem in road restoration. *Transportation Research Part B: Methodological*, 141:24–58.

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.

Muritiba, A. E. F., Bonates, T. O., Da Silva, S. O., and Iori, M. (2021). Branch-and-cut and iterated local search for the weighted k-traveling repairman problem: an application to the maintenance of speed cameras. *Transportation Science*, 55(1):139–159.

Ngueveu, S. U., Prins, C., and Calvo, R. W. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 37(11):1877–1885.

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.

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.

Ribeiro, G. M. and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728–735.

Rivera, J. C., Afsar, H. M., and Prins, C. (2016). Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. *European Journal of Operational Research*, 249(1):93–104.

Roberti, R. and Mingozzi, A. (2014). Dynamic ng-Path Relaxation for the Delivery Man Problem. *Transportation Science*, 48(3):413–424.

Sahin, 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.

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.

Sze, J. F., Salhi, S., and Wassan, N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transportation Research Part B: Methodological*, 101:162–184.

Vodák, R., Bíl, M., and Křivánková, 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.

Online Supplements

Appendices

A. Mathematical model associated with synchronized MML-RCP (M1)

The decision variables are presented in Table 9. In addition, there are extra sets and parameters used in the following formulation as defined below.

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 work teams which are initially positioned at depot $d \in D$. In addition K is the set of work teams.

Table 9: Decision variables used in the model associated with synchronization

Notation	Type	Definition
x^k_{ijl}, $\forall (i,j),(j,l) \in E, \forall k \in K$	binary	If work team k crosses (j,l) right after (i,j) in its walk
z^k_{ijl}, $\forall (i,j) \in B,(j,l) \in E, \forall k \in K$	binary	If (i,j) is unblocked by work team k
y^k_{ijl}, $\forall j \in V_C, \forall i:(i,j) \in E, \forall l:(j,l) \in E, \forall k \in K$	binary	If work team k reaches critical node j for the first time while crossing (j,l) right after (i,j) in its walk
f^k_{ij}, $\forall (i,j) \in E, \forall k \in K$	continuous	The flow corresponding to work team k's route on (i,j) from node i to j
v^k_i, $\forall i \in V \cup \{n+1\}, \forall k \in K$	continuous	Number of times work team k visits node i
t^k_{ijl}, $\forall (i,j),(j,l) \in E, \forall k \in K$	continuous	The time that work team k arrives to node j from i before going to l
s^k_{ijl}, $\forall (i,j),(j,l) \in E$	continuous	If arc (i , j) is in the walk of work team k ; it is the earliest time that work team k can enter { i , j }. Otherwise, it is 0
τ^k_{ijl}, $\forall j \in V_C, \forall i:(i,j) \in E, \forall l:(j,l) \in E, \forall k \in K$	continuous	The time that work team k arrives to node j from i before going to l. Otherwise becomes M
γ^k_{ijl}, $\forall j \in V_C, \forall i:(i,j) \in E, \forall l:(j,l) \in E, \forall k \in K$	continuous	The **first** time that work team k arrives to node j from i before going to l
T_j, $\forall j \in V_C$	continuous	latency of node j

$$\min \quad \sum_{j \in V_C} T_j \tag{21}$$

$$T_j \leq \tau^k_{ijl} \qquad \forall j \in V_C, i:(i,j) \in E, l:(j,l) \in E, \forall k \in K \tag{22}$$

$$\sum_{l:(j,l) \in E} \sum_{i:(i,j) \in E} \sum_{k \in K} y^k_{ijl} = 1 \qquad \forall j \in V_C \tag{23}$$

$$\tau^k_{ijl} = M\left(1 - x^k_{ijl}\right) + t^k_{ijl} \qquad \forall j \in V_C, i:(i,j) \in E, l:(j,l) \in E, \forall k \in K \tag{24}$$

$$\gamma^k_{ijl} \geq \tau^k_{ijl} - M\left(1 - y^k_{ijl}\right) \qquad \forall j \in V_C, i:(i,j) \in E, l:(j,l) \in E, \forall k \in K \tag{25}$$

$$\gamma^k_{ijl} \leq \tau^k_{ijl}, \qquad \forall j \in V_C, i:(i,j) \in E, l:(j,l) \in E, \forall k \in K \tag{26}$$

$$T_j = \sum_{l:(j,l) \in E} \sum_{i:(i,j) \in E} \sum_{k \in K} \gamma^k_{ijl} \qquad \forall j \in V_C \tag{27}$$

$$y^k_{ijl} \leq x^k_{ijl} \qquad \forall j \in V_C, i:(i,j) \in E, l:(j,l) \in E, \forall k \in K \tag{28}$$

$$x^k_{ijl} \leq \sum_{h:(l,h) \in E} x^k_{jlh} \qquad \forall (i,j) \in E, k \in K, l:(j,l) \in E, \forall k \notin P_i \tag{29}$$

$$x^k_{ijl} \leq \sum_{h:(h,i) \in E} x^k_{hij} \qquad \forall (i,j) \in E, \forall k \in K, l:(j,l) \in E, \forall k \notin P_i \tag{30}$$

$$\sum_{h:(h,i) \in E} x^k_{hij} \leq 1 \qquad \forall (i,j) \in E, \forall k \in K \tag{31}$$

$$\sum_{l:(j,l)\in E} x^k_{ijl} \le 1 \qquad \forall (i,j)\in E, \forall k \in K \tag{32}$$

$$\sum_{j:(d,j)\in E} \sum_{l:(j,l)\in E} x^k_{djl} = 1 \qquad \forall d \in D, \forall k \in P_d \tag{33}$$

$$\sum_{(i,j)\in E} x^k_{ij(n+1)} = 1 \qquad \forall k \in K \tag{34}$$

$$\sum_{l:(j,l)\in E} x^k_{ijl} \ge z^k_{ij} \qquad \forall (i,j)\in B, k \in K \tag{35}$$

$$\sum_{l:(j,l)\in E} x^k_{ijl} + \sum_{h:(i,h)\in E} x^k_{jih} \le 2\left(\sum_{\kappa=1}^{K} z^\kappa_{ij} + \sum_{\kappa=1}^{K} z^k_{ji}\right) \qquad \forall (i,j)\in B, \forall k \in K \tag{36}$$

$$\sum_{\kappa=1}^{K} \left(z^\kappa_{ij} + z^\kappa_{ji}\right) \le 1 \qquad \forall (i,j)\in B \tag{37}$$

$$\sum_{j:\in V\cup\{(n+1)\}:(i,j)\in E} \left(f^k_{ij} - f^k_{ji}\right) = -v^k_i \qquad \forall k \in K, \forall i \in V\cup\{(n+1)\}\backslash D \tag{38}$$

$$\sum_{j\in V\cup\{(n+1)\}} \left(f^k_{dj} - f^k_{jd}\right) = \sum_{i\in V\cup\{(n+1)\}\backslash\{d\}} v^k_i \qquad \forall k \in P_d, \forall d \in D \tag{39}$$

$$\sum_{j\in V} f^k_{j(n+1)} = 1 \qquad \forall k \in K \tag{40}$$

$$f^k_{ij} \le (n-1) \sum_{l:(j,l)\in E} x^k_{ijl} \qquad \forall k \in K, \forall (i,j)\in E,\ i,j \in V\cup\{(n+1)\} \tag{41}$$

$$f^k_{ij} \ge \sum_{l:(j,l)\in E} x^k_{ijl} \qquad \forall k \in K, \forall (i,j)\in E,\ i,j \in V\cup\{(n+1)\} \tag{42}$$

$$\sum_{j:(j,i)\in E} \sum_{l:(i,l)\in E} x^k_{jil} = v^k_i \qquad \forall k \in K, \forall i \in V\cup\{(n+1)\} \tag{43}$$

$$t^k_{ijl} \le M x^k_{ijl} \qquad \forall k \in K, i:(i,j)\in E, \forall l:(j,l)\in E \tag{44}$$

$$s^k_{ij} \le \sum_{h:(h,i)\in E} t^k_{hij} + M\left(1 - z^k_{ij}\right) \qquad \forall k \in K, \forall (i,j)\in B \tag{45}$$

$$s^k_{ij} \ge \sum_{h:(h,i)\in E} t^k_{hij} \qquad \forall k \in K, \forall (i,j)\in B \tag{46}$$

$$s^k_{ij} \ge \sum_{l:(j,l)\in E} t^\kappa_{ijl} - M\left(1 - z^\kappa_{ij}\right) - 2M\left(1 - \sum_{h:(h,i)\in E} x^k_{hij}\right) \qquad \forall k,\kappa \in K, \forall (i,j)\in B, \kappa \ne k \tag{47}$$

$$\sum_{l:(j,l)\in E} t^k_{ijl} \ge s^k_{ij} + u_{ij} z^k_{ij} + \sum_{l:(j,l)\in E} x^k_{ijl} t_{ij}, \forall k \in K \quad \forall (i,j)\in B \tag{48}$$

$$\sum_{l:(j,l)\in A} t^k_{ijl} \ge \sum_{h:(h,i)\in E} t^k_{hij} + \sum_{l:(j,l)\in A} x^k_{ijl} t_{ij} \qquad \forall k \in K, \forall (i,j)\in E\backslash B \tag{49}$$

$$x^k_{ijl} \in \{0,1\}, \forall k \in K, \quad i:(i,j)\in E, \quad \forall l:(j,l)\in E \tag{50}$$

$$z^k_{ij} \in \{0,1\}, \quad \forall (i,j)\in B, \quad \forall k \in K \tag{51}$$

$$f^k_{ij} \ge 0, \quad \forall (i,j)\in E, \quad \forall k \in K \tag{52}$$

$$v^k_i \ge 0, \quad \forall i \in V, \quad \forall k \in K \tag{53}$$

$$t^k_{ijl} \ge 0, \forall k \in K, \quad i:(i,j)\in E, \quad l:(j,l)\in E \tag{54}$$

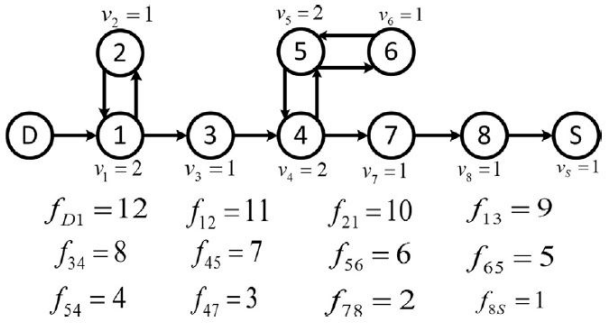


Figure 9: Demonstration of the flow formulation (Akbari and Salman 2017b)

$$s_{ij}^k \geq 0, \quad \forall (i,j) \in B, \quad \forall k \in K \tag{55}$$

The objective function calculates the total latency of all critical nodes. Note that all walks end in the sink node indexed as $n + 1$. Constraints (22)-(27) force the model to pick the minimum visiting time of critical node j as the value of its latency. Constraints (22) guarantee that the latency of critical node j should be less than or equal to all visiting times of node j by all work teams. Constraints (23) ensure that only one of the visiting times of critical node j should be considered as its latency (given the other constraints, the model picks the earliest time that critical node j is visited). In addition, it implies that all critical nodes must be visited. Constraints (24) provide the visiting time of critical node j for each work team that visits this critical node (on the other hand, if critical node j is not visited by a work team, its visiting time becomes equal to big M.). By constraints (25) and (26), LL_{ijl}^k finds the minimum time to reach critical node j over all work teams and routes. As a result, Constraints (27) calculate the latency of critical node j. If no work team traverses node j, then variable y should be zero as well, which is shown by Constraints (28). Constraints from (29) to (32) represent the flow balance equations. First, when a work team enters an arc and it must leave it in the next step of its walk. Moreover, the work team is allowed to leave the edge if it traverses that edge in the previous step of the walk, which is stipulated by Constraints (29) and (30). Moreover, Constraints (31) and (32) enforce that each arc is traversed at most once. Constraints (33) and (34) make each work team start its walk in its corresponding depot and ends it in the sink node ($n + 1$). Constraints (35) force that a blocked edge gets cleared only if a work team traverses it. Constraints (36) ensure that a work team cannot traverse an edge unless it is cleared by a work team. By (37), at most one work team is allowed to clear a blocked edge. We use flow variables f_{ij}^k for every work team and for each arc $(i, j) \in E$, to guarantee the connectivity of the walks. The net flow out of a depot node is the total number of visits to all nodes except the depot. For other nodes, net flow equals to the number of visits to the corresponding node. Figure 9 (taken from Akbari and Salman (2017b)) gives an example of the usage of the flow variables and how they enable the route of a work team to be connected. In the given walk in Figure 9, 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 consumed. Finally one unit of flow enters the sink node. Constraints (38) and (39) make a work team leave one unit of flow whenever it visits a node. Constraints (40) ensure that walks end in the sink node. Constraints (41) do not provide flow on an edge if it is not traversed. Constraints (42) force a positive amount of flow which passes through

an edge whenever that edge is traversed. Constraints (43) calculate the number of times work team k visits node $i \in V$. Constraints sets (44) to (49) are for setting time limits. Constraints (44) ensure that, if $x^k_{ijl} = 0$, then $t^k_{ijl} = 0$, indicating that edge (j, l) is not visited right after (i, j), so that the corresponding time to reach node j must be zero. Constraints (45), (46) and (47) calculate s^k_{ij} values. Variable s^k_{ij} shows the earliest time that edge $(i, j) \in B$ is traversable by work team k, if work team k is planned to cross it. On the other hand, s^k_{ij} should be forced to be zero, if work team k does not traverse $(i, j) \in B$. Traversing a blocked edge can only start by a work team when either the particular work team is the opener of the edge, or the edge is already unblocked by another work team. Due to constraints (45) and (46), if work team 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 $z^k_{ij} = 0$, then (45) is redundant, and by (46) work team k can start traversing (i, j) no sooner than its arrival time to node i. If a work team traverses a blocked edge without opening it, it is only possible if the edge is unblocked earlier by another work team. In addition, $s^k_{ij} = 0$ should be satisfied, if work team k is not planned to cross $(i, j) \in B$. Constraints sets (48) and (49) calculate the arrival times to an edge (i, j) for $(i, j) \in B$ and $(i, j) \in E \setminus B$, respectively. Constraints (50)-(55) define the domains of the decision variables.

B. Parameter tuning for H2

In the H2 algorithm given in section 4.2, a number of parameters and certain setting was utilized. In this section, we conduct a number of computational experiments to justify our choices. For all the experiments in this section, we have analyzed all the 20 instances from the Kartal data set with 15 critical nodes and 3 work teams. For each instance, we run the H2 algorithm 10 times to address the randomness of our algorithm.

We have used four neighborhood search moves in H2 denoted by N_1, N_2, N_3 and N_4. We have used them exactly in the same order in our algorithm. Table 10 gives the results of testing different orders including N1-N2-N3-N4, N2-N3-N1-N4 and N1-N4-N2-N3. In this table, the Average OFV column gives the average objective function value obtained from 10 repetitions of running H2 on that instance and Best OFV gives the best objective function value found in those instances. The Average LIT column gives the time in seconds in which the last improvement was achieved. As it can be observed in this table, the N1-N2-N3-N4 finds the best objective function values on average. Moreover, we observe that the time in which the last improvement is made is always less than 10 seconds, hence we have set a time limit of 30 seconds for H2 in our computational experiments.

We have also conducted a number of experiments to see whether all the moves are actively improving the objective function or not. For the same three different orders, under the same scenarios as in Table 10, we have also observed the total number of times a successful move from each of the neighborhood moves was executed. These results are presented in Table 11. As it can be observed in this table, all the moves were successful in finding better solutions. While N3 seems to have the most number of successful executions, N4 had the lowest successful utilization among the moves. Nevertheless, all the moves had numerous successful executions.

In our local search, when an incumbent solution is not improved in i_{max} iteration, a perturbation step is implemented to avoid the local optima. In Table 12, we have investigated the impacts of changing the value

Table 10: Impact of the neighborhood orders on the obtained solutions

Order	N1-N2-N3-N4			N2-N3-N1-N4			N1-N4-N2-N3		
Instance	Average OFV	Best OFV	Average LIT	Average OFV	Best OFV	Average LIT	Average OFV	Best OFV	Average LIT
1	253.8	253	4.15	254.8	253	3.29	253.8	253	4.00
2	261.5	251	3.57	257.3	251	3.74	258.3	251	2.95
3	257.8	252	3.73	254.1	252	4.07	258.4	252	4.09
4	266.2	256	3.53	258.7	256	3.52	262.1	256	2.14
5	262.1	256	1.63	266.2	256	3.27	259.4	256	3.66
6	278.6	278	5.54	278.2	278	3.82	278.2	278	4.47
7	291.9	289	2.29	291.2	289	6.49	289.4	289	4.22
8	291.1	291	5.57	292.5	291	2.89	292	291	4.42
9	284	284	5.20	285.4	284	6.23	286.8	284	4.74
10	280.5	280	1.92	280.8	280	1.31	280.6	280	1.94
11	328	309	2.31	331.9	331	1.32	332.5	331	3.51
12	332	332	3.89	332	332	5.41	334	332	5.15
13	314.4	308	6.13	313.7	308	5.26	314.4	308	4.46
14	303.8	297	4.65	302.5	297	4.69	302	297	3.70
15	277.2	276	6.57	277.2	276	6.65	276	276	7.62
16	581.2	573	5.86	576.6	573	4.04	576.5	573	6.28
17	504.4	498	4.45	509	502	3.82	500.9	498	4.61
18	520.4	517	5.86	520.8	517	6.21	525.2	517	6.62
19	428	428	4.88	430.1	428	6.25	432.2	428	4.71
20	484.7	482	5.32	488.9	482	5.22	486.1	482	6.10
Average	340.08	335.5	4.35	340.10	336.8	4.38	339.94	336.6	4.47

of i_{max}. We have tested different values in the range from 100 to 300. Based on our observations, while the impacts of this parameter are not significant given the selected range, the average Best OFV and the Average of the Avg OFV values are both in their minimum state when $i_{max} = 200$. As a result, we have set $i_{max} = 200$ in all our computational experiments.

In the perturbation step of the H2 algorithm, the double swap move is applied N_{DS} times on the incumbent solution. We have also investigated the impacts of changing the value of N_{DS} on the obtained solutions. The results of these experiments are presented in Table 13. We have considered different values from 3 to 7 for our experiments. As it can be observed, when $N_{DS} = 5$, both the average of the Best OFV and the average of the average OFV is improved. As a result, for our computational experiments, we set $N_{DS} = 5$.

C. Multiple Makespan Minimization MIP (3M-MIP)

The Multiple Makespan Minimization MIP (3M-MIP) finds the minimum time in which a certain number of critical nodes can be visited by the available work teams. For instance 3M-MIP(n, M) is the makespan minimization problem in which M work teams visit n critical nodes. Similar to MML-RCP, the route of a work team starts in its depot and ends in a dummy sink node indexed by N+1. We also add a dummy edge from each node in V to the dummy sink node and set its cost equal to 0. Similar to MML-RCP, P_d is the set of work teams positioned in the depot $d \in D$. In the following, we first give the decision variables of the

Table 11: Total number of successful neighborhood moves

Order	N1-N2-N3-N4				N2-N3-N1-N4				N1-N4-N2-N3			
Instance	N1	N2	N3	N4	N1	N2	N3	N4	N1	N2	N3	N4
1	12	38	26	2	16	22	24	4	12	27	23	5
2	15	11	24	0	9	15	34	2	12	13	26	0
3	23	9	22	2	24	22	25	2	17	25	26	2
4	6	13	25	0	5	14	31	1	8	19	23	3
5	4	15	28	2	4	14	26	4	7	16	30	3
6	8	23	32	3	11	13	35	1	13	24	36	0
7	9	19	29	4	20	23	39	8	14	18	29	2
8	13	6	24	3	12	10	27	1	14	13	24	3
9	17	3	23	5	17	11	24	3	16	8	23	6
10	4	8	33	0	7	10	29	0	7	13	36	4
11	14	17	28	0	4	9	27	0	9	6	29	1
12	15	18	27	3	18	9	30	3	16	19	21	3
13	9	7	31	10	10	12	29	4	14	10	30	7
14	11	19	25	2	12	20	26	4	9	15	37	5
15	14	39	24	3	18	31	35	4	26	42	24	1
16	19	18	38	13	17	15	35	10	24	21	37	11
17	15	22	28	3	12	18	29	1	16	13	19	2
18	23	29	30	12	17	20	37	10	35	15	37	12
19	15	21	41	7	13	20	34	9	24	15	38	9
20	22	22	35	11	15	19	29	13	17	19	42	9

Table 12: Analysis of the impact of i_{max} on the results

i_{max}	100		150		200		250		300	
Instance	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV
1	253	253.8	253	254.8	253	253.6	253	255	253	254.8
2	251	261.5	251	259.4	251	261.1	251	264.6	251	260.6
3	252	257.8	252	255.6	252	254.3	252	256.3	252	259.2
4	256	266.2	256	257.7	256	267.9	256	264.5	256	259.4
5	256	262.1	256	263.8	256	262.8	256	269.6	256	260.4
6	278	278.6	278	278	278	278	278	278.8	278	278
7	289	291.9	289	289.7	289	289.9	289	290	289	289.7
8	291	291.1	291	292.7	291	291.5	284	290.3	291	293
9	284	284	284	284	284	284	284	284	284	284
10	280	280.5	280	280.2	280	280.5	280	281	280	280.4
11	309	328	331	333.1	309	324.1	331	332.7	332	333.2
12	332	332	332	332	332	332	332	332	332	332
13	308	314.4	308	313.7	308	314.4	308	316.2	308	317.1
14	297	303.8	297	304.8	297	300.7	297	304.1	287	301.2
15	276	277.2	276	277.9	276	276.8	276	276.8	276	283.6
16	573	581.2	573	580.1	565	573.6	565	575.2	573	577.3
17	498	504.4	498	505.1	498	505.9	498	504.6	498	504
18	517	520.4	517	524	517	523.8	517	523.6	517	521.3
19	428	428	428	432.2	428	431.5	428	437.8	428	436.1
20	482	484.7	482	489.5	482	491.5	482	488.5	482	489.6
Average	335.5	340.08	336.6	340.415	335.1	339.895	335.85	341.28	336.15	340.745

Table 13: Analysis of the impacts of N_{DS} on the results

N_{DS}	3		4		5		6		7	
Instance	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV	Best OFV	Avg OFV
1	253	254.4	253	254.6	253	253.6	253	253.4	253	254.2
2	251	255.2	251	255.2	251	261.1	251	262.5	251	267.8
3	252	259.9	252	257	252	254.3	252	261.4	252	257
4	256	260.2	256	266.2	256	267.9	256	259.4	256	263.8
5	256	264.5	256	263.9	256	262.8	256	262.8	256	265.5
6	278	278	278	278	278	278	278	278.6	278	279.4
7	289	290.3	289	290	289	289.9	289	292.3	289	290.9
8	291	292.5	291	291.8	291	291.5	291	292	291	294.3
9	284	285.4	284	284	284	284	284	285.8	284	285.4
10	280	280.2	280	280.6	280	280.5	280	280.2	280	280.3
11	309	328	309	326.3	309	324.1	332	332.3	309	328
12	332	332	332	332	332	332	332	332	332	332
13	308	315.2	308	313.6	308	314.4	308	317.3	308	314.8
14	297	307.3	297	304.3	297	300.7	297	303.1	297	302.5
15	276	278.4	276	276	276	276.8	276	276	276	276
16	570	574.1	565	574.2	565	573.6	565	572.8	565	572.7
17	498	507.3	504	509.4	498	505.9	498	507.5	502	505.9
18	517	522.6	517	525.1	517	523.8	517	522.6	517	523
19	428	434.3	428	432.2	428	431.5	428	429.4	428	438.3
20	482	482	482	487.4	482	491.5	482	485.2	482	487.4
Average	335.35	340.09	335.4	340.09	335.1	339.895	336.25	340.33	335.3	340.96

3M-MIP and then give the objective function and constraints associated with 3M-MIP with m work teams and visiting n critical nodes.

Table 14: Decision variables of the 3M-MIP

Notation	Type	Definition
$x_{ij}^m, \forall (i,j) \in E, \forall m \in M$	binary	If work team m goes from nodes i to j.
$z_{ij}^m, \forall (i,j) \in B, \forall m \in M$	binary	If work team m unblocks blocked edge (i, j).
$f_{ij}^m, \forall (i,j) \in E, \forall m \in M$	continuous	The flow corresponding to work team m's route on (i, j) from node i to j
$v_i^m, \forall i \in V \cup \{N+1\}, \forall m \in M$	continuous	Number of times work team m visits node i
$y_c, c \in V_c$	binary	If critical node $c \in V_c$ is visited or not.
W	continuous	The value of the objective function

$$Min \;\; W \tag{56}$$

$$W \geq \sum_{(i,j)\in E} t_{ij} x_{ij}^m + \sum_{(i,j)\in B} u_{ij} z_{ij}^m, \;\; m \in M \tag{57}$$

$$\sum_{j \in V \cup \{(n+1)\}:(d,j)\in E} (x_{dj}^m - x_{jd}^m) = 1, \quad \forall d \in D, \;\; \forall m \in P_d \tag{58}$$

$$\sum_{j \in V \cup \{(N+1)\}:(i,j)\in E} (x_{ij}^m - x_{ji}^m) = 0, \;\; m \in M, \;\; \forall i \in V \backslash D \tag{59}$$

$$\sum_{j \in V} x_{j(N+1)}^m = 1, \;\; m \in M \tag{60}$$

$$x_{ij}^m + x_{ji}^m \geq z_{ij}^m, \quad m \in M, \quad \forall (i,j) \in B \tag{61}$$

$$x_{ij}^m + x_{ji}^m \leq 2 \sum_{l \in M} z_{ij}^l, \quad m \in M, \quad \forall (i,j) \in B \tag{62}$$

$$\sum_{m \in M} z_{ij}^m \leq 1, \quad \forall (i,j) \in B \tag{63}$$

$$\sum_{j:(j,i) \in E} x_{ji}^m = v_i^m, \quad m \in M, \quad \forall i \in V \cup \{(N+1)\} \tag{64}$$

$$f_{ij}^m \leq (|V|-1) x_{ij}^m, \quad m \in M, \quad \forall (i,j) \in E \tag{65}$$

$$f_{ij}^m \geq x_{ij}^m, \quad m \in M, \quad \forall (i,j) \in E \tag{66}$$

$$\sum_{j:\in V \cup \{(N+1)\}:(i,j) \in E} (f_{ij}^m - f_{ji}^m) = -v_i^m, \quad m \in M, \quad \forall i \in V \cup \{(N+1)\} \setminus D \tag{67}$$

$$\sum_{j \in V \cup \{(N+1)\}} (f_{dj}^m - f_{jd}^m) = \sum_{i \in V \cup \{(N+1)\} \setminus \{d\}} v_i^m, \quad \forall m \in P_d, \quad \forall d \in D \tag{68}$$

$$\sum_{j \in V} f_{j(N+1)}^m = 1, \quad m \in M \tag{69}$$

$$y_c \leq \sum_{m \in M} v_c^m, \quad c \in V_c \tag{70}$$

$$\sum_{c \in V_c} y_c = n \tag{71}$$

$$x_{ij}^m \in \{0,1\}, \quad (i,j) \in E, \quad m \in M \tag{72}$$

$$z_{ij}^m \in \{0,1\}, \quad (i,j) \in B, \quad m \in M \tag{73}$$

$$f_{ij}^m \geq 0, \quad (i,j) \in E, \quad m \in M \tag{74}$$

$$v_i^m \geq 0, \quad i \in V, \quad m \in M \tag{75}$$

$$y_c \in \{0,1\}, \quad c \in V_c \tag{76}$$

Given constraints (56) together with (57) we set the makespan minimization objective function. By constraints (58), (59) and (60) we ensure that work teams start their routes from the depot and end them in the dummy sink node while considering the flow balance for them. Constraints (61), (62) and (63) are to ensure that blocked edges are not traversed unless they are opened and they cannot be opened unless they are traversed by one of the work teams. Moreover, only one work team can be assigned to opening a blocked edge. Constraint (64) counts the number of times a node is visited by a work team. Constraints from (65) to (69) are similar to flow constraints in the MML-RCP and ensure that the routes of the vehicles are not disjoint (sub-tour elimination). Constraint (70) and (71) are to fix the number of critical nodes that are visited by work teams to n. The remaining constraints are the variable restrictions.

D. Computational results associated with M1

In this Section, we provide the results associated with M1 with total numbers of nodes 8, 10 and 13, where 2 work teams are employed. Moreover, we report the results of M1 for 3 and 4 critical nodes. Recall that M1 is able to solve small instances having up to 13 nodes, 4 critical nodes and 2 work teams. We

also compare the M1 solutions with proposed algorithms associated with H1 and H2. In total, there are 60 instances such that all of their parameters have been randomly generated. Tables 15 and 16 present the results associated with M1 and its comparison with the solution obtained by H1 and H2. Under column M1, the optimal total latency corresponding to each instance is given. Columns D_{H1} and D_{H2} provide the difference percentage with M1 results, such that $D_{H1} = \frac{D_{H1}-M1}{D_{H1}} \times 100$ and $D_{H2} = \frac{D_{H2}-M1}{D_{H2}} \times 100$. Recall that in these instances the number of both total and critical nodes are small, meaning that the possibility of finding optimal solutions with disjoint-path assumption are low. Thus, as we observe in these tables, H2 is able to find better (or the same) solutions than ones in H1 in all instances.

Table 15: Computational results associated with M1 with 3 critical nodes

NO. Nodes	8			10			13		
Instance	M1	D_{H1}	D_{H2}	M1	D_{H1}	D_{H2}	M1	D_{H1}	D_{H2}
1	45	0	0	55	2.1	0	68	2.2	0
2	39	0	0	42	0	0	71	3.8	0
3	42	5.9	0	57	0.9	0	63	0	0
4	51	4.6	0	52	0	0	57	0	0
5	38	0	0	49	0	0	75	4.2	0
6	37	0	0	59	1.6	0	69	0	0
7	54	3.8	0	38	0	0	56	1.5	0
8	58	0	0	46	0	0	78	6.8	3.9
9	32	0	0	43	0.5	0	49	0	0
10	48	0	0	58	1.4	0	63	2.7	1.9
Average		1.4	0		0.7	0		2	0.6

Table 16: Computational results associated with M1 with 4 critical nodes

NO. Nodes	8			10			13		
Instance	M1	D_{H1}	D_{H2}	M1	D_{H1}	D_{H2}	M1	D_{H1}	D_{H2}
1	53	0	0	71	3.8	0	95	3.6	1.9
2	61	0	0	80	5.2	0.8	89	3.2	0
3	45	0	0	67	1.6	0	101	6.9	2
4	49	0	0	75	4.1	1.1	76	0.5	0
5	58	2.3	0	59	0	0	89	0	0
6	69	5.9	1.2	46	0.8	0	93	7.4	2.8
7	47	0	0	45	0	0	72	0	0
8	52	0	0	82	7.4	2.7	88	3.4	0.2
9	61	2.6	1.4	87	3.5	3.4	105	5.7	4.5
10	66	1.2	0	67	0	0	74	0.6	0
Average		1.2	0.3		2.7	0.8		3.2	1.1