An online optimization approach for post-disaster relief distribution with online blocked edges[1]

Vahid Akbari[a,*], Davood Shiri[b]

[a]*Nottingham University Business School, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, United Kingdom*
[b]*Department of Industrial Engineering, Istanbul Medipol University, 34810 Istanbul, Turkey*

Abstract

Disasters can disrupt road networks by blocking some of the roads and consequently impeding accessibility to critical locations. In the immediate post-disaster response phase, while the blockage information is yet to be collected, relief distribution crews (RDCs) should dispatch from warehouses to supply critical locations with first aid items. The RDCs are not capable of unblocking damaged roads and should find a way to bypass them once such edges are observed in their routes. With the objective of minimizing total latency of the critical nodes, we study the problem that addresses the relief distribution operations with k non-recoverable online blocked edges. The online blocked edges are not known to the RDCs initially and the blockage of a blocked edge is revealed when one of the RDCs arrives at one of its end-nodes. Once one of the RDCs knows about a blocked edge, this information is communicated among the rest of the RDCs and they will all be informed about that blocked edge. We first investigate the worst-case performance of online algorithms against offline optimal solutions using competitive ratio. We then prove a lower bound on the competitive ratio of deterministic online algorithms. We also provide an upper bound on the competitive ratio of the optimal deterministic online algorithms by introducing a deterministic algorithm which achieves a bounded competitive ratio. We then develop three heuristic algorithms to solve this problem. One of our algorithms is based on solving an Integer Programming model to find the assignment of the nodes to the RDCs and then finding the routes dynamically. The other algorithms are not based on solving optimization models and hence are more efficient in terms of their computational time. We compare our proposed heuristic algorithms with the best known algorithms from the literature that are developed for a single RDC variation of the problem. Finally, we provide a through computational analysis of our algorithms on instances adopted from real-world road networks.

Keywords: Online optimization, relief distribution, competitive ratio, minimum latency problem, travelling repairman problem.

[1]DOI: https://doi.org/10.1016/j.cor.2021.105533
*Corresponding author
Email addresses: vahid.akbari@nottingham.ac.uk (Vahid Akbari), davood.shiri@medipol.edu.tr (Davood Shiri)

1. Introduction

Natural disasters can cause negative impact on both people and infrastructures. While the psychological stress level among impacted people rises, they seek assistance for first-aid items in emergency assembly nodes and pre-identified critical locations such as hospitals. In order to ensure that the demand of the casualties is met, relief distribution crews should dispatch from warehouses and supply the demand of these critical nodes. Meanwhile, the impact of the disaster can cause blockage in some of the roads. This could occur by accumulated debris from fallen buildings, twisted road signs and lamp posts, displaced cars or uprooted plantations. The relief distribution crews do not have the required equipment to unblock these blocked edges and should avoid them in their routes. Furthermore, in the immediate post-disaster stage, before blockage information is collected, the relief distribution crews do not have information about the location of the blockages as well.

Since providing timely service to casualties is of the utmost importance in immediate post-disaster scenarios, we have considered the latency minimization objective in which the summation of the time of the visits to the critical nodes is minimized. The latency minimization objective is known to be a service or customer oriented objective (Martínez-Salazar *et al.* 2015). Hence, this objective is a suitable option for relief distribution operations. The *minimum latency problem* (MLP) has been investigated extensively in operations research. In this problem, a traveling agent receives an undirected connected graph $G = (V, E)$ such that $V = \{v_0, v_1, v_2, ..., v_n\}$ and v_0 is the depot node where the agent is initially positioned. In addition, each edge in E has a non-negative edge distance (time) that is known to the traveling agent. In our study, we represent the latency of v_i by l_i which corresponds to the total traveled distance (time) from when the agent leaves the depot until v_i is visited. The goal of the MLP is to identify a tour for the agent that starts from v_0 and visits all the nodes in V while minimizing the summation of the latency values, i.e., $\sum_{i=1}^{n} l_i$.

This problem is also referred to as the *traveling repairman problem* (TRP) (Afrati *et al.* 1986), the *delivery man problem* (Fischetti *et al.* 1993) and the *cumulative traveling salesman problem* (Bianco *et al.* 1993). The combination of the Traveling Salesman Problem (TSP) and the MLP using a multi-objective model has been studied in Bock & Klamroth (2019) as well. The MLP has several applications in various topics including and not limited to disaster relief distribution (Ajam *et al.* 2019), machine scheduling (Picard & Queyranne 1978), minimizing the waiting times of the passengers in a public transportation system (Saharidis *et al.* 2014), post-disaster damage assessment (Bruni *et al.* 2020), and maintenance of speed cameras (Muritiba *et al.* 2021).

We focus on an online version of the MLP involving k non-recoverable blockages which are not known to the traveling agent from the beginning. In this problem, a subset of the edges in G denoted by B is assumed to be blocked but not known in advance. Moreover, we assume that these blocked edges are non-recoverable and $|B| = k$, i.e., the road network contains k blocked edges that are not recoverable. The blockage of

a blocked edge (i, j) is only revealed after the agent arrives to either of the nodes i or j. Similar to the literature (Zhang *et al.* (2019) and Zhang *et al.* (2016)), we refer to these blocked edges as *online blocked edges*. We assume that eliminating these online blocked edges from G, does not make the road network disconnected. In other words, $G_B = (E, V \setminus B)$ is a connected graph. Since the online blocked edges are non-recoverable, the connectivity of G_B is essential to make sure that accessibility to all the nodes is granted to the agent. The goal is to propose an online solution in which the traveling agent identifies a tour which does not traverse any blocked edges, starts from v_0 and minimizes $\sum_{i=1}^{n} l_i$. This problem is referred to as the *online minimum latency problem with edge uncertainty* (OMLP) and has been addressed in Zhang *et al.* (2019). This study modeled the real-time traffic uncertainties such as un-predicted traffic, extreme weather conditions or accidents as online blocked edges. In our study however, we focus on the disaster management application and the online blocked edges represent the unknown damaged edges that are blocked due to accumulated debris on the roads. We note that road clearance operations are not addressed in our study as the blocked roads are assumed to be non-recoverable. We also translate visiting each of the nodes as conducting the relief distribution operations in that specific node.

In our study, we generalize the OMLP and consider the problem with multiple agents. We also refer to an agent as a *Relief Distribution Crew* (RDC) hereafter. In the multiple OMLP, L ($L < k$) homogeneous RDCs are initially located at the depot node (v_0). The RDCs can communicate with each other in the sense that they can send and receive the revealed blockage information. A blocked edge is known to all the RDCs when at least one of them arrives at an end-node of that edge. Once blockage of an edge is revealed to one of the RDCs, it is then communicated among them immediately and all of them will be informed about the location of the blocked edge and should avoid it in their routes as these online blocked edges are non-recoverable. Hence, the RDCs may repeat an edge or a node in their routes when they update their routes with respect to the revealed blockage information. As mentioned above, we denote the latency of a node v_i by l_i which gives the distance (time) traveled before the first visit of v_i by one of the RDCs. The objective is to identify a tour for each of the RDCs that starts from v_0 such that the collection of the tours of the RDCs minimizes $\sum_{i=1}^{n} l_i$. We refer to this problem as the Multiple Online Minimum Latency Problem with Edge Uncertainty (MOMLP).

Online solutions take their input piece by piece and should be updated according to each new piece of input they receive. Their objective is to obtain a performance which is close to the optimal performance obtainable if the entire input is known from the beginning. The performance of online solutions is measured by means of *competitive analysis* in the literature. In competitive analysis approach, the quality of an online algorithm is evaluated by comparing its results with the offline optimal solutions using the competitive ratio measure (Sleator & Tarjan 1985). Online algorithms are classified into two types as deterministic and randomized. A randomized online algorithm is designed based on some probability distribution, whereas

a deterministic online algorithm does not depend on probabilistic outcomes (Albers 2003). For an online minimization problem, the competitive ratio of a deterministic online algorithm is the maximum ratio of the cost of that online algorithm to the cost of the offline optimum which is calculated over all problem instances. Similarly, the expected competitive ratio of a randomized online algorithm for a minimization problem is equivalent to the maximum ratio of the expected cost of the online algorithm to the cost of the offline optimum over all problem instances. In the offline variation of the MOMLP, the blockages are removed from the graph. Therefore, the solution of the offline problem can be obtained from the traditional multiple minimum latency problem where the blocked edges are eliminated from the road network.

The organization of the paper is as follows. In Section 2, we provide the related literature and highlight the contributions of our study. In Section 3, we present the theoretical competitive analysis of the MOMLP. In Section 4, we introduce the deterministic online heuristic algorithms which are developed to tackle the MOMLP. Data generation and the computational results are given in Section 5. Finally, concluding remarks are provided in Section 6.

2. Literature review

The focus of our study is on online routing problems involving online blocked edges which find applications in post-disaster relief distribution operations. Therefore, we divide the literature review into two sections. First, we review the literature of relevant routing problems that are investigated from online optimization and competitive analysis perspectives. Next, we briefly present the related works that address post-disaster relief distribution operations.

2.1. Related online routing problems

Different variations of the routing problems are widely analyzed within the framework of competitive analysis in the literature (see (Jaillet & Wagner 2008b), (Jaillet & Wagner 2006), and (Jaillet & Wagner 2008a)). In these studies, the locations to be visited are not known initially and are learned incrementally over time. Different from these studies, we focus on online routing problems in which the unknown parameter is a bounded number of online blocked edges that are unknown initially and are disclosed once their blockage is learned by an agent. Works on these problems are divided into two streams according to the number of agents defined in the problem. The first stream investigates problems involving one agent and the second stream studies problems with multiple agents considering their communication as well.

2.1.1. Problems with a single agent

The single-agent offline MLP which does not involve online blocked edges has been studied in the literature extensively. In order to solve this problem, exact approaches such as mathematical modeling (Angel-Bello *et al.* 2013) and branch-and-price algorithm (Bulhões *et al.* 2018) have been proposed and

tested. Several heuristic approaches such as general variable neighborhood search (Mladenović *et al.* 2013) and tabu search (Dewilde *et al.* 2013) have also been developed and tested on this problem and its variations as well. The main concentration of this section is to review the related articles to the MLP that involve online blocked edges.

Papadimitriou & Yannakakis (1991) was the first study which used online blockages to model edge uncertainty in the *Canadian Traveler Problem* (CTP). The CTP is a single-agent variant of the shortest path problem on graphs involving online blockages. Papadimitriou & Yannakakis (1991) showed that designing an online procedure which obtains a bounded competitive ratio for the CTP is PSPACE-complete. Westphal (2008) studied the online k-CTP, a version of the CTP with a fixed number of k online blocked edges. Westphal (2008) provided an optimal deterministic online algorithm together with a lower bound of $2k+1$ on the competitive ratio of deterministic algorithms. Also, Westphal (2008) proved that no randomized online algorithm obtains an expected competitive ratio better than $k+1$. Bender & Westphal (2015) proposed an online randomized procedure for the online k-CTP which is optimal on graphs containing at most two online blockages and edge-disjoint paths. Shiri & Salman (2019b) provided a randomized algorithm for the online k-CTP and showed its optimality on graphs containing arbitrary number of online blockages and edge-disjoint paths.

Liao & Huang (2014) studied a variant of the Traveling Salesman Problem (TSP) on a complete graph in which there exist k online blockages and named it the online *Covering Canadian Traveler Problem* (CCTP). They introduced a deterministic online procedure within an $o(\sqrt{k})-$competitive ratio for the CCTP. Zhang *et al.* (2015) investigated the Steiner TSP involving k non-recoverable online blockages in which a traveling agent should visit a certain subset of the nodes. Zhang *et al.* (2015) provided a lower bound on the competitive ratio of deterministic procedures together with an exponential time deterministic online algorithm which matches the given lower bound. Also, they presented an asymptotically optimal polynomial-time online deterministic procedure and evaluated its performance on randomly generated sparse graphs. Zhang *et al.* (2016) investigated a variant of the Steiner TSP having k non-recoverable advanced online blockages, in which the traveler learns about a blockage at a certain distance from it. Zhang *et al.* (2016) derived a lower bound on the competitive ratio of deterministic online procedures. Additionally, they introduced a polynomial-time deterministic online algorithm and tested it on sparse randomly generated graphs. Zhang & Xu (2018) analyzed the online covering salesman problem having k online blockages in which the goal is to identify a tour such that each node in the graph is either on the tour or within a predetermined distance from a node on the tour. Zhang & Xu (2018) proposed a lower bound for the competitive ratio of deterministic online procedures as well as a deterministic solution with a bounded competitive ratio.

As mentioned, Zhang *et al.* (2019) studied the single-agent minimum latency problem having k online blockages (i.e., OMLP). Zhang *et al.* (2019) showed a lower bound of $2k+1$ for the competitive ratio of

deterministic online solutions and introduced an exponential time deterministic heuristic procedure called GoodTreeTraversal which achieves a competitive ratio close to $2k + 1$ in special instances in which the number of blockages in the graph is sufficiently large. In addition, Zhang *et al.* (2019) provided a polynomial time deterministic heuristic procedure named Detour whose performance is evaluated on real city as well as randomly generated instances. Akbari & Shiri (2021) studied a variant of the OMLP involving k online blockages where priority weights are assigned to the nodes and the objective is to minimize the total weighted latency of nodes. They provided an optimal deterministic online procedure as well as a tight lower bound for the competitive ratio of online deterministic algorithms. They also proposed two efficient heuristic algorithms that are tested on real city instances.

2.1.2. Problems with multiple agents

The multi-agent offline variation of the MLP has also received significant consideration in the literature. In the offline problem, all the inputs of the problem are known to the agents in advance and hence there are no online blocked edges. Similar to the single-agent problem, several exact approaches such as mathematical modeling (Angel-Bello *et al.* 2019) and branch-and-price-and-cut algorithm (Luo *et al.* 2014) have been proposed for the multi-agent offline MLP. Heuristic approaches have also been developed and tested to address the multi-agent offline MLP as well. Ngueveu *et al.* (2010) utilized a memetic algorithm and Avci & Avci (2019) developed an adaptive large neighborhood search to tackle the multi-agent offline MLP. Similar to the single-agent MLP, the focus of this literature study is to investigate the utilization of online blocked edges in the MLP and other related problems.

A multi-agent problem involving online blocked edges is considered first in Zhang *et al.* (2013) where an extension of the online k-CTP involving multiple homogeneous travelers is studied. In the online multiple k-CTP, the travelers are initially located at a given depot node and the goal is to minimize the total time until the arrival of at least one of the agents to the destination node. Zhang *et al.* (2013) proposed lower bounds on the competitive ratio of online deterministic procedures for the cases with limited and complete communication between agents. They proposed a deterministic online heuristic procedure called *Alternating* which is optimal for special instances where there are only two travelers in the input graph. Shiri & Salman (2017) and Shiri & Salman (2019a) further investigated online deterministic and randomized procedures, respectively, for the online multiple k-CTP in different communication levels from a theoretical competitive analysis perspective. We note that, all the aforementioned studies consider homogeneous agents and are conducted from a theoretical worst-case competitive analysis perspective.

The online multiple k-CTP has been also studied from a computational perspective in Shiri & Salman (2020), where an efficient heuristic deterministic online procedure is introduced and experimented on real-world as well as random instances. Shiri *et al.* (2020) analyzed a search-and-rescue problem with multiple teams on graphs having non-recoverable online blockages where the service time of a node is unknown

initially and is disclosed online when the node is visited. Shiri *et al.* (2020) proved two lower bounds for the competitive ratio of online deterministic procedures as well as two efficient online deterministic heuristic policies tested on randomly generated graphs.

2.2. Related relief distribution problems

Ozdamar *et al.* (2004) provided a mathematical formulation which integrates the multi-commodity network flow and the vehicle routing problems to provide delivery schedules for the relief distribution vehicles and applied a Lagrangian relaxation method to solve this problem. Sheu (2007) proposed a three layered emergency response distribution using a fuzzy clustering-optimization approach. Tzeng *et al.* (2007) applied a multi-objective formulation with three objectives which are minimizing the total cost and travel times while maximizing the minimum satisfaction during the planning period, to find the optimal planning of a relief delivery system. Najafi *et al.* (2013) considered a multi-objective stochastic formulation to propose a robust decision planning in the earthquake response phase where the number of injured casualties from different categories is uncertain. Liu *et al.* (2019) provided a multi-commodity, multi-period formulation which considers both delivering of the distribution of relief items and serving injured people where the objective is the total weighted unmet demand.

Several articles used stochastic optimization methods to address the relief distribution operations while considering blocked edges on the post-disaster road network. Similar to our study, in these articles, the blockages are non-recoverable and should be avoided to process the relief distribution operations. Rawls & Turnquist (2010) used a two-stage stochastic formulation and provided a pre-positioning policy for placement of relief supplies. They applied a Lagrangian L-shaped procedure to solve larger instances and evaluated their models on real-world instances. Moreno *et al.* (2018) proposed a two-stage stochastic formulation to optimize location, transportation, and fleet sizing decisions by considering the uncertainties about the demand, incoming supply, and availability of the routes. They also presented matheuristic procedures to solve their problem and experimented their solutions on real-world instances. Hu *et al.* (2019) provided a multi-agent multi-stage stochastic formulation for relief delivery operations and developed a progressive hedging algorithm (PHA) to tackle larger instances. They measured the performance of their model and algorithm on real-world instances. The main difference between our study and the ones mentioned above is that we address an online optimization problem in which no prior probabilistic information about the blockage of the roads is available.

There are some other related articles which investigate post-disaster relief distribution in different contexts such as search and rescue operations (Shiri *et al.* (2020) and Liu *et al.* (2020)), evacuation and resettlement of victims (Hu *et al.* 2014), as well as integrating the location, inventory and routing decisions in the relief distribution operations (Wei *et al.* (2020), Pérez-Rodríguez & Holguín-Veras (2016), and Wang *et al.* (2014)).

2.3. *Our contributions*

In this article, we investigate the MOMLP for the first time in the literature. We focus on the application of the problem in post-disaster relief distribution. We consider multiple RDCs and online blocked edges which represent transportation links that are damaged by the disaster. First, we analyze this online optimization problem in a theoretical worst-case competitive analysis framework. We prove a lower bound on the competitive ratio of deterministic online algorithms for the MOMLP. A lower bound on the competitive ratio of deterministic algorithms for an online problem means that no deterministic algorithm can approach the offline optimum than the corresponding lower bound. This result demonstrates the value of having complete input information before the online problem is solved. Furthermore, to derive an upper bound for the competitive ratio of the optimal deterministic algorithm for the MOMLP, we provide a deterministic online algorithm which achieves a bounded competitive ratio. Applying the lower and upper bounds that we introduce, we specify an interval for the competitive ratio of the optimal deterministic online algorithm for the MOMLP. Our theoretical results improve and extend the previous results that are proposed in Zhang *et al.* (2019) for a special case of the MOMLP with a single RDC (i.e., the OMLP).

Moreover, with the aim of providing efficient solutions for real-world applications, we propose three heuristic deterministic online algorithms to solve the MOMLP. The first algorithm is based on solving an Integer Programming (IP) model that finds the assigned nodes and the order of their visit to each RDC. In this algorithm, the traversal of the nodes should be simulated dynamically to avoid traversing blocked edges. This is because in the IP model, we do not consider the blocked edges as they are only revealed online after a close observation by one of the RDCs. Since the IP-based algorithm is based on solving an optimization model, it requires access to an optimization solver and once the size of the network increases, its required computational time increases significantly. In order to address this issue, we developed two more algorithms that do not rely on solving an optimization model. In one of these algorithms, we use the K-means clustering algorithm to find an assignment of nodes to each RDC and then simulate their traversal dynamically to avoid traversing blocked edges. In the other algorithm, we use a greedy procedure to dynamically assign the closest node to each RDC. Once the next nodes that each RDC should visit are determined, their traversal is simulated. In order to evaluate these algorithms, we first compare them with the algorithms from the literature on instances with one RDC, i.e. the OMLP. Once we approve that our algorithms outperform those from the literature, we adopt real-life instances from the literature that are based on three different real-world networks generated from the road network of Istanbul. We conduct thorough computational experiments to test all our developed algorithms on these networks.

3. Competitive analysis

We analyze the MOMLP from a competitive analysis viewpoint. We first derive a lower bound on the competitive ratio of deterministic online solutions for the MOMLP. Next, we prove an upper bound on the competitive ratio of the optimal deterministic solution for this problem by proposing a deterministic online algorithm which achieves a bounded competitive ratio for the MOMLP. Here we emphasize that the blockage information is revealed incrementally in the MOMLP and the RDCs should update their routes whenever a new blockage information is obtained. Therefore, each edge or node can be visited more than once by the RDCs in the provided online solutions for the MOMLP.

Lemma 3.1. *The competitive ratio of no deterministic online procedure is less than* $2\lfloor\frac{k}{L}\rfloor+1$ *for the MOMLP, where k and L (k > L) are the number of blockages and RDCs, respectively.*

Proof. To derive the lower bound, we should show that for an arbitrary deterministic solution, *ALG*, there exists at least one instance of the MOMLP such that *ALG* cannot obtain a competitive ratio less than $2\lfloor\frac{k}{L}\rfloor+1$ on that instance. For our proof, we use the instance shown in Figure 1, where

$$V = \{v_0,\ v_{(k+1)(x+1)+1}\} \cup \{v_{(k+1)u+q} |\ u = 0, 1, ..., x, q = 1, 2, ..., k+1\}$$

and

$$E = \{(v_0, v_q),\ (v_{(k+1)u+q}, v_{(k+1)(u+1)+q}),\ (v_{(k+1)x+q}, v_{(k+1)(x+1)+1})\} |\ u = 0, 1, ..., x-1, 1 \le q \le k+1\}.$$

We set the travel times of the edges (v_0, v_q) to 1 for $q = 1, 2, ..., k+1$ and the travel times of the rest of edges to 0. Let $V^{'} = \{V \setminus \{v_0, v_1, v_2, ..., v_{k+1}\}\}$.

- **Cost of an arbitrary deterministic online algorithm.** We analyze an arbitrary deterministic solution, *ALG*, applied to the instance in Figure 1. In *ALG*, each RDC starts from v_0 and takes an edge (v_0, v_q) $(q = 1, 2, ..., k+1)$. Next, two scenarios may arise. In the first scenario, the RDC finds out that the edge $(v_q, v_{(k+1)+q})$ is open and traverses it. Then, the RDC takes all the edges with travel time zero to visit the $x(k+1)+1$ nodes in $V^{'}$. In the second scenario, the RDC finds out that the edge $(v_q, v_{(k+1)+q})$ is blocked, returns to v_0 and tests a new edge.

  Let $E^{'} = \{(v_q, v_{(k+1)+q}) | q = 1, 2, ..., k+1\}$. For *ALG*, we consider the instance where k edges in $E^{'}$ are blocked and only one of the edges in $E^{'}$ is open. Without loss of generality, let $(v_{k+1}, v_{(k+1)+k+1})$ be the only open edge in $E^{'}$. For *ALG*, we consider the instance where the RDCs encounter k blocked edges in $E^{'}$ before arriving at v_{k+1}. In this case, the L RDCs can visit at most L, $2L$, ..., $L\lfloor\frac{k}{L}\rfloor$ nodes from $\{v_1, v_2, ..., v_k\}$ by times 1, 3, 5, ..., $2\lfloor\frac{k}{L}\rfloor - 1$, respectively. The total latency of these nodes is at least $L\sum_{i=1}^{\lfloor\frac{k}{L}\rfloor}(2i-1)$.

Also, the remaining $k-L\lfloor\frac{k}{L}\rfloor$ nodes in $\{v_1, v_2, ..., v_k\}$, v_{k+1}, and the $x(k+1)+1$ nodes in $\{V\backslash\{v_0, v_1, v_2, ..., v_{k+1}\}\}$ cannot be visited earlier than time $2\lfloor\frac{k}{L}\rfloor + 1$. Hence, the total latency of *ALG* is at least

$$(L\sum_{i=1}^{\lfloor\frac{k}{L}\rfloor}(2i-1)) + (k - L\lfloor\frac{k}{L}\rfloor + 1 + x(k+1) + 1) \times (2\lfloor\frac{k}{L}\rfloor + 1).$$

- **Cost of the offline optimal solution.** In the offline optimal solution, one of the RDCs begins from v_0 and takes the edge (v_0, v_{k+1}) to arrive at v_{k+1} which is an end-node of the only open edge in $E^{'}$. Next, the RDC takes all the edges with travel time zero to visit the $x(k+1)+1$ nodes in $V^{'}$. Hence, the latency of v_{k+1} and the nodes in $V^{'}$ is one, i.e., the summation of the latencies of node v_{k+1} and the nodes in $V^{'}$ is $x(k+1)+2$. The other $L-1$ RDCs take $L-1$ other edges from $E^{'}$. Hence, the latency of $L-1$ nodes in $\{v_1, v_2, ..., v_k\}$ is also one. Therefore, the latency of $x(k+1)+L+1$ nodes is one and their total latency is $x(k+1)+L+1$.

  When the RDCs visit all the $x(k+1)+L+1$ number of nodes with latency one, they backtrack to v_0. At time 2, all the RDCs are at v_0. Then, the remaining unvisited $k-L+1$ nodes in $\{v_1, v_2, ..., v_k\}$ are visited such that L, $2L$, ..., $L(\lfloor\frac{k}{L}\rfloor - 1)$ nodes are visited by times 3, 5, ..., $2\lfloor\frac{k}{L}\rfloor - 1$, respectively. That is, $L(\lfloor\frac{k}{L}\rfloor - 1)$ nodes from the remaining (unvisited $k-L+1$ nodes at time 2) are visited by time $2\lfloor\frac{k}{L}\rfloor - 1$, i.e., the summation of the latencies of these $L(\lfloor\frac{k}{L}\rfloor - 1)$ nodes is $L\sum_{i=2}^{\lfloor\frac{k}{L}\rfloor}(2i-1)$. Finally, the remaining unvisited $k+1-L\lfloor\frac{k}{L}\rfloor$ nodes are visited at time $2\lfloor\frac{k}{L}\rfloor + 1$. Thus, the total latency of the offline optimal solution is

$$(x(k+1)+L+1) + (L\sum_{i=2}^{\lfloor\frac{k}{L}\rfloor}(2i-1)) + ((k+1-L\lfloor\frac{k}{L}\rfloor)(2\lfloor\frac{k}{L}\rfloor + 1)).$$

The lemma follows when x approaches $+\infty$.

□

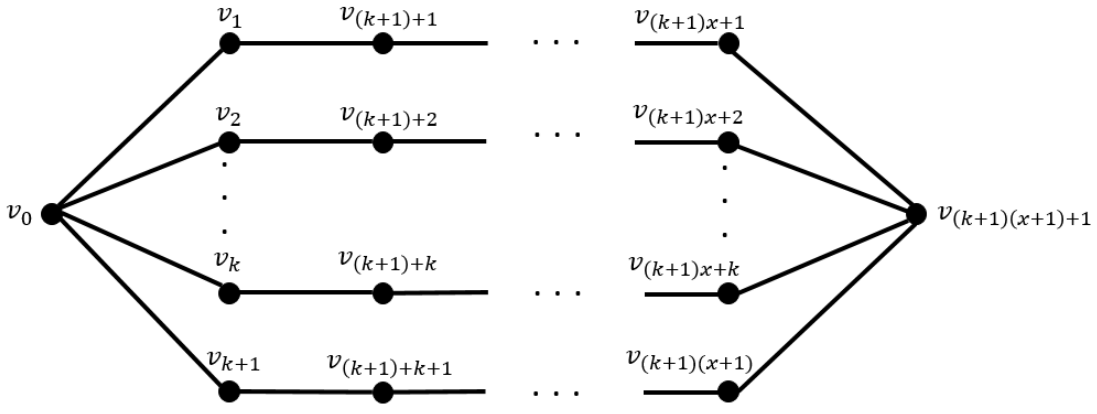

Figure 1: The instance for proving the lower bound of $2\lfloor\frac{k}{L}\rfloor + 1$

Lemma 3.1 states that the competitive ratio of the optimal deterministic online solution for the MOMLP is at least $2\lfloor\frac{k}{L}\rfloor + 1$. Next, we provide an upper bound of $2k + 1$ for the competitive ratio of the optimal deterministic online algorithm for the MOMLP by introducing a deterministic online algorithm which achieves the competitive ratio of $2k + 1$. We call this deterministic online algorithm the *multi-agent back-to-root* (MBR) algorithm. The MBR is an iterative algorithm and ends in at most $k + 1$ iterations. At the beginning of the $q^{th}(q \in \{1, 2, ..., k+1\})$ iteration, the RDCs eliminate the found blocked edges from the graph and apply an offline Integer Programming (IP) model given in Angel-Bello *et al.* (2019) (assuming there is no blocked edge in the graph) to compute a collection of tours such that a tour T_q^l which starts from v_0 is assigned to the l^{th} $(l \in \{1, 2, ..., L\})$ RDC and the collection of the routes of the RDCs minimizes the total latency of all the nodes in V for the MOMLP. We note that Angel-Bello *et al.* (2019) presented an IP formulation which is applicable to solve the offline versions of the MOMLP under the assumption that complete input information is available. We have given this IP formulation in section Appendix A. When T_q^l is constructed, the l^{th} RDC travels on T_q^l. Then, either all the RDCs complete the traversal of their assigned tours and visit all the nodes in V or at least one new blockage is found. In the former case, the iteration finishes and the algorithm terminates. In the latter case, all the RDCs backtrack to v_0 and the iteration finishes. The detailed steps of the MBR algorithm are presented in Algorithm 1.

Algorithm 1: MBR algorithm

Input:

a: $G = (V, E)$ ▷ initial graph containing the blocked edges

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

c: v_0 ▷ the root node of the RDCs

d: S ▷ set of destination nodes

1: **Initiation:**

a: $F = \emptyset$ ▷ F : set of found blocked edges.

b: $q = 1$ ▷ q : counter variable

c: $G_1 = G = (V, E)$ ▷ G_q : updated graph with revealed information in step q

d: flag = True ▷ flag: variable to terminate loops.

2: **While** flag **do:**

3: $T_q^l \leftarrow MMLP(G_q) : l \in \{1, ..., L\}$

▷ $MMLP(G_q)$: the exact IP model for the multiple MLP on graph G_q.

▷ T_q^l : the assigned tour to the l^{th} RDC at the q^{th}iteration by the IP model on G_q.

4: Traverse $T_q^l : l \in \{1, 2, ..., L\}$ ▷ the l^{th} RDC traverses T_q^l for $l = 1, 2, ..., L$.

5: **if** $\nexists\ (i, j) \in T_q^l : (i, j)$ is blocked for $l \in \{1, 2, ..., L\}$ **then:** ▷ no blocked edges in T_q^l for $l = 1, 2, ..., L$.

6: flag = 0 ▷ Termination Point.

7: **Else:**

8: the l^{th} RDC traverses T_q^l until at least one of the RDCs reaches a blocked edge b_q.

9: $F = F \cup b_q$

10: $q = q + 1$

11: $G_q = (V, E \setminus F)$

12: all the RDCs return to root node, v_0.

13: **end while**

Theorem 3.2. *The competitive ratio of the MBR algorithm is* $2k+1$ *for the MOMLP, where* k *is the number of blockages.*

Proof. Assume that the MBR algorithm ends at iteration q^* ($q^* \in \{1, 2, ..., k+1\}$). We define $l_i^{q^*}$ ($q^* \in \{1, 2, ..., k+1\}$) as the travel time between v_0 and v_i in the offline optimal solution, i.e., the total latency of the offline optimal solution is $\sum_{i=1}^{n} l_i^{q^*}$. Let $X_q^L \subset V$ represent the set of nodes that are visited for the first time at iteration q ($q \in \{1, 2, ..., q^*\}$) by all the L RDCs, i.e., the nodes in X_q^L are not visited before the q^{th} iteration by any of the RDCs. Let Δ_q^L denote the minimum travel time that one of the L RDCs has spent starting from v_0 to arrive at one of the end-nodes of the q^{th} found blockage at the q^{th} iteration for $q = 1, 2, ..., q^*-1$. We define l_i^q ($q \in \{1, 2, ..., q^*\}$) as the travel time between v_0 and v_i in the offline IP formulation that is solved at the beginning of the q^{th} ($q \in \{1, 2, ..., q^*\}$) iteration. The total latency of the MBR algorithm can be represented as

$$\sum_{q=1}^{q^*} \sum_{i \in X_q^L} l_i^q + \sum_{q=1}^{q^*-1} \sum_{u \in V - \sum_{j=1}^{q} X_j^L} 2\Delta_q^L = \sum_{q=1}^{q^*-1} \Big(\sum_{i \in X_q^L} l_i^q + \sum_{u \in V - \sum_{j=1}^{q} X_j^L} 2\Delta_q^L \Big) + \sum_{i \in X_{q^*}^L} l_i^{q^*}.$$

Since one edge is removed from the graph at the end of each iteration, we have

$$\sum_{i=1}^{n} l_i^1 \leq \sum_{i=1}^{n} l_i^2 \leq ... \leq \sum_{i=1}^{n} l_i^{q^*-1} \leq \sum_{i=1}^{n} l_i^{q^*}.$$

We point out that it holds that

$$\sum_{i \in X_q^L} l_i^q + \sum_{u \in V - \sum_{j=1}^{q} X_j^L} \Delta_q^L \leq \sum_{i=1}^{n} l_i^q \leq \sum_{i=1}^{n} l_i^{q^*}$$

for $q = 1, 2, 3, ..., q^*$. Hence, it holds that

$$\sum_{i \in X_q^L} l_i^q + \sum_{u \in V - \sum_{j=1}^{q} X_j^L} 2\Delta_q^L \leq 2 \sum_{i=1}^{n} l_i^q \leq 2 \sum_{i=1}^{n} l_i^{q^*}$$

for $q = 1, 2, 3, ..., q^* - 1$. Note that $\sum_{i \in X_{q^*}^L} l_i^{q^*} \leq \sum_{i=1}^{n} l_i^{q^*}$ and the total latency of the offline optimal solution is $\sum_{i=1}^{n} l_i^{q^*}$. Thus, the competitive ratio of the MBR procedure is at most $2(q^* - 1) + 1$ which is not more than $2k + 1$ since $q^* \in \{1, 2, ..., k+1\}$.

□

Corollary 3.2.1. *The optimal deterministic online solution for the MOMLP has a competitive ratio between* $2\lfloor \frac{k}{L} \rfloor + 1$ *and* $2k + 1$*, where* k *and* L *(*$k > L$*) are the number of blockages and RDCs, respectively.*

4. Algorithms to solve the MOMLP

In this section, we present three deterministic online algorithms to tackle the MOMLP. The first algorithm is based on solving an offline IP model iteratively and then utilizing the obtained routes to find a feasible solution to the MOMLP. We refer to this algorithm as the *IP-based* algorithm. Since the IP-based algorithm is based on solving an IP model, its CPU run time increases significantly when the size of the problem increases. For that, we propose two other polynomial time deterministic online algorithms. We call these algorithms the *Cluster-First-Route-Second (CFRS)* and the *Greedy* algorithms. While in the original MLP problem all the nodes should be visited, in a real-life situation, usually some of the nodes are pre-identified to be visited. Our algorithms work both when all or some of the nodes should be visited.

4.1. IP-based algorithm

We develop an algorithm that is based on solving an optimization model iteratively and then finding the routes for each of the RDCs. For that, we first need to introduce an IP model that solves the offline version of the MOMLP called the Multiple Minimum Latency Problem (MMLP). In order to solve the MMLP, we apply the IP model given in Angel-Bello *et al.* (2019). This model is given in Appendix A. From this point forward, we refer to this model as the Offline Integer Programming Model (OIPM).

The pseudocode of the IP-based algorithm is given in Algorithm 2. The input of this algorithm is the initial graph given as $G = (V, E)$, the set of nodes that should be visited given by $\mathcal{V}$, the associated traversal cost or time of traveling edge $(i, j) \in E$ which is given by c_{ij}. The depot node given as v_0 and the number of RDCs denoted by L. In the IP-based algorithm, we first solve the OIPM on the initial graph (G) assuming that there is no blocked edge in G. In this graph, the blocked edges are not known and hence in the solution found by the OIPM, some of the blocked edges might have been traversed by RDCs. However, we are not utilizing the obtained routes from the applied OIPM on G and instead, we extract the set of nodes that each RDC visits and the order in which they are visited. This is given as O_l for RDC $l \in \{1, ..., L\}$. Once the nodes and the order in which they should be visited by each RDC is obtained, we set up a number of variables to simulate the traversal of the RDCs. This traversal should be in a way that none of the RDCs traverses a blocked edge and once the blockage information on an edge is revealed to one of them, all the other RDCs should receive the same information. For that, we define a graph denoted by G' that is equal to G before the RDCs start traversing the edges. Once some information on the blocked edges is revealed to one of the RDCs, we then update graph G' and eliminate the revealed blocked edge from it.

We also keep track of the dynamic position of each RDC $l \in \{1, ..., L\}$ by P_l. We then enter a traversal stage which ends only after all the nodes in the set $\mathcal{V}$ are visited. For the traversal step of the IP-based algorithm, given the orders obtained from the mathematical model, we first extract the next node that each RDC should visit. For each RDC l, the next node that should be visited is the first un-visited node in O_l. We

denote this by u_l in the pseudocode. We then identify the first node node on the shortest path route from the current position of each RDC l, P_l, to the node u_l and denote it by v_l. In the next step of the algorithm given in step 10, we search among the RDCs and find the index of the RDC that has the shortest distance to the next node on its path. This is to simulate the state of the system and find the next time that an event changes its status. We denote the index of this RDC by $\boldsymbol{\ell}$. If $(P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$ is a blocked edge, it will be added to the list of known blocked edges and consequently G' will be updated as well. We then recalculate the next nodes that each of the RDCs should visit and find the index of the RDC that has the shortest path to the next node on its route. But if $(P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$ is not a blocked edge, RDC $\boldsymbol{\ell}$ traverses to $v_{\boldsymbol{\ell}}$ and the current position of the RDC $\boldsymbol{\ell}$ will be updated. Moreover, if this node is the next node that was assigned to RDC $\boldsymbol{\ell}$ ($v_{\boldsymbol{\ell}} = u_{\boldsymbol{\ell}}$), we add $u_{\boldsymbol{\ell}}$ to the set of visited nodes. This procedure continues until all the nodes are visited by the RDCs.

Algorithm 2: IP-based Algorithm

Input: $G = (V, E)$, $\mathcal{V}$, c_{ij}, v_0, L.

1: $O_l = OIPM(G), \quad \forall l \in \{1, ..., L\}$
▷ $OIPM(G)$: the offline IP model solved on the initial graph with no blocked edges.
▷ O_l : the order of visiting nodes for RDC $l \in \{1, ..., L\}$.
2: $G' = G$ ▷ G' : the graph that will get updated with blockage information once they are revealed
3: $F = \emptyset$ ▷ F : set of revealed blocked edges.
4: $N = \emptyset$ ▷ N : set of visited nodes.
5: $P_l = v_0, \quad l \in \{1, ..., L\}$ ▷ P_l : position of RDC $l \in \{1, ..., L\}$.
6: $T_l = 0, \quad l \in \{1, ..., L\}$ ▷ T_l : time of the last event for RDC $l \in \{1, ..., L\}$.
7: **While** $N \neq \mathcal{V}$: **do:**
8: u_l : first node in $O_l \notin N \quad l \in \{1, ..., L\}$ ▷ u_l : first node in O_l that has not been visited
9: $v_l = SP(P_l, u_l, G') \quad l \in \{1, ..., L\}$
▷ v_l : first node on the shortest path from current position of RDC l to u_l on G'.
10: $\boldsymbol{\ell} = \underbrace{\arg\min}_{1}\{T_l + c_{P_l v_l} \quad l \in \{1, ..., L\}\}$
▷ $\boldsymbol{\ell}$: the RDC that has the shortest distance to the next node on his path
11: **If** $(P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$ is blocked **then:**
a: $F = F \cup (P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$
b: $G' = (V, E \setminus F)$
c: go to 9
12: **Else:**
a: $T_{\boldsymbol{\ell}} = T_{\boldsymbol{\ell}} + c_{P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}}}$
b: $P_{\boldsymbol{\ell}} = v_{\boldsymbol{\ell}}$
c: **If** $P_{\boldsymbol{\ell}} = u_{\boldsymbol{\ell}}$ **then:**
d: $N = N \cup u_{\boldsymbol{\ell}}$
e: go to 7
f: **Else:**
g: go to 9
h: **end if**
13: **end if**
14: **end while**

4.2. The CFRS algorithm

In order to obtain the routes of the RDCs using the IP-based algorithm, we need to solve an optimization model which is very time consuming to solve when the size of the problem increases. Moreover, in case the decision makers do not have access to optimization solvers, they cannot use the IP-based algorithm. In order to remedy these issues, we developed two more algorithms. The first one is based on a cluster-first-route-second procedure. We refer to this algorithm as the CFRS algorithm. The pseudocode of this algorithm is given in Algorithm 3.

Here we describe how to generate the assignment of nodes to the RDCs using the K-means clustering algorithm. In an optimal solution of the MOMLP, at least one node must be assigned to each RDC. In the CFRS algorithm, we keep the same rule meaning that we do not allow any idle RDCs in our solutions. We apply the K-means clustering algorithm in order to find an assignment of the nodes to each RDC. We note that (1) the number of generated clusters equals the number of RDCs and (2) the RDCs are homogeneous and are all initially positioned in the same depot node. 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 L groups. The K-means algorithm starts off by choosing L initial cluster centers (centroids which are chosen randomly) and then iteratively improves them as follows. Each node is assigned to its nearest centroid. Then, each centroid is updated to be 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 coordinates of the nodes that should be visited ($\mathcal{V}$) and the corresponding distances in our K-means clustering algorithm.

As it is stated above, the number of clusters is the same as the number of RDCs denoted by L. Given the coordinates of each node that should be visited by (X_v, Y_v), we implement the K-means clustering algorithm to obtain the best possible clustering associated with each RDC. The coordinates of each node are treated as its features for the K-means clustering algorithm. Since this algorithm cluster the nodes based on their coordinates and proximity to each other, the total latency of the nodes which are visited by each RDC could be reduced and as a result, the total latency of all the nodes may be decreased. Some authors (Ajam *et al.* (2021), Cinar *et al.* (2016), Reed *et al.* (2014) and Geetha *et al.* (2012)) took advantage of the K-means clustering algorithm to obtain fairly good solutions to different routing problems.

By applying the K-means clustering algorithm with the above description, we can find L clusters of nodes and then associate the nodes in the l^{th} cluster to RDC $l = \{1, ..., L\}$. The set of nodes that are assigned to each RDC is denoted by A_l in the pseudocode. We can see that the collection of the nodes in these sets presents the set of nodes that should be visited ($A_1 \cup A_2 \cup ... \cup A_L = \mathcal{V}$). Once the assigned nodes to each RDC are determined, the RDCs can start their traversal procedure. Different from the IP-based algorithm where both the nodes and the order in which they should be visited are determined by the optimization model, in

the CFRS algorithm, A_l only gives the nodes assigned to RDC l and the order of their visit should be decided in the traversal step of the CFRS algorithm. Similar to the traversal step of the IP-based algorithm, we define a graph G' that will be updated once blockage information of a blocked edge is revealed. We also denote the dynamic position of RDC l by P_l. Again we denote the next node that RDC l should visit by u_l and use the formula described in step 8 to find it. Using this formula, u_l is the closest node from the set of assigned nodes to RDC l, A_l, that has not been visited yet. Similar to the IP-based algorithm, for each RDC l, we find the node that should be visited first on the shortest path route from P_l to u_l. We denote this node by v_l. The remaining steps of the CFRS algorithm are similar to those of the IP-based algorithm. In the computational section we will show that the CPU run time of this algorithm is significantly lower than those of the IP-based algorithm.

Algorithm 3: CFRS Algorithm

Input: $G = (V, E), \mathcal{V}, c_{ij}, v_0, L, (X_v, Y_v) : \quad v \in V.$

1: $A_l = KMC(L, (X_v, Y_v) : v \in V_0), \quad \forall l \in \{1, ..., L\}$
▹ $KMC(L, (X_v, Y_v) : v \in V_0)$: the K means clustering algorithm to generate L clusters with coordinates of nodes as their features.
▹ A_l : the nodes that are in the l^{th} cluster of the applied K means clustering procedure.

2: $G' = G$ ▹ G' : the graph that will get updated with blockage information once they are revealed

3: $F = \emptyset$ ▹ F : set of revealed blocked edges.

4: $N = \emptyset$ ▹ N : set of visited nodes.

5: $P_l = v_0, \quad l \in \{1, ..., L\}$ ▹ P_l : position of RDC $l \in \{1, ..., L\}$.

6: $T_l = 0, \quad l \in \{1, ..., L\}$ ▹ T_l : time of the last event for RDC $l \in \{1, ..., L\}$.

7: **While** $N \neq \mathcal{V}$: **do:**

8: $u_l = \underbrace{\arg\min}_{u}\{SP(P_l, u, G') \;\; u \in A_l : u \notin N\} \;\; l \in \{1, ..., L\}$
▹ u_l : the closest node from the set of assigned nodes to RDC l that has not been visited yet.

9: $v_l = SP(P_l, u_l, G') \;\; l \in \{1, ..., L\}$
▹ v_l : first node on the shortest path from current position of RDC l to u_l on G'.

10: $\boldsymbol{\ell} = \underbrace{\arg\min}_{l}\{T_l + c_{P_l v_l} \;\; l \in \{1, ..., L\}\}$
▹ $\boldsymbol{\ell}$: the RDC that has the shortest distance to the next node on his path

11: **If** $(P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$ is blocked **then:**

a: $F = F \cup (P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$

b: $G' = (V, E \setminus F)$

c: go to 9

12: **Else:**

a: $T_{\boldsymbol{\ell}} = T_{\boldsymbol{\ell}} + c_{P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}}}$

b: $P_{\boldsymbol{\ell}} = v_{\boldsymbol{\ell}}$

c: **If** $P_{\boldsymbol{\ell}} = u_{\boldsymbol{\ell}}$ **then:**

d: $N = N \cup u_{\boldsymbol{\ell}}$

e: go to 7

f: **Else:**

g: go to 9

h: **end if**

13: **end if**

14: **end while**

4.3. Greedy algorithm

In order to further explore the performance of our algorithms and to provide an alternative algorithm to the CFRS algorithm that does not rely on the clustering procedure, in this section we develop a Greedy algorithm to solve the MOMLP. This algorithm is given as Algorithm 4.

In the Greedy algorithm, different from the CFRS and IP-based algorithms, we do not have initial assignment of the nodes to the RDCs for their traversals. Instead, we decide which node to be visited next by each RDC dynamically. This is done by considering the nodes that have been already visited and the positioning of each RDC. The inputs of this algorithm are similar to those of the IP-based algorithm, including the input intact graph, the traversal cost or time for each edge given as c_{ij}, the depot node for the RDCs, the number of RDCs given by L and the set of nodes that should be visited ($\mathcal{V}$). In this algorithm, we define a set A, that keeps track of the nodes that has been assigned to an RDC. The main difference of this algorithm and the CFRS is in selecting the next node that each RDC should visit. Similar to the IP-based and the CFRS algorithms, we denote the next node that RDC $l = \{1, ..., L\}$ should visit by u_l and calculate this node using steps 8 to 11. Given these formulas, we first assign the closest node that has not yet been visited to each RDC and then update the set of nodes that have been visited. Once the next assigned node to each RDC is determined, the remaining traversal steps are similar to the CFRS algorithm.

5. Computational analysis of the algorithms

We present computational analysis for the algorithms developed for the MOMLP in this section. We first compare our algorithms with algorithms from the literature that are generated for a variation of the MOMLP with only one agent (i.e., RDC). This is because the MOMLP is introduced in our study for the first time. We then analyze the performance of our algorithms on real-life graphs that are generated from the road network of Istanbul, Turkey. All our experiments are conducted on a computer with Intel Xeon W-2123 3.60 GHz 3.60 GHz processor, 32 GB RAM and 64-bit Windows 10 Professional operating system. The algorithms are coded in Python. We solve the exact models using Gurobi Optimizer 9.0 in the Python environment.

5.1. Comparison of the IP-based, the CFRS and Greedy algorithms with the literature

In order to provide a comparison of our algorithms with the literature, we compared our algorithms with the proposed algorithms in Zhang *et al.* (2019) the instances that were used by this paper. Since Zhang *et al.* (2019) studied the problem with one agent (i.e., RDC), we also applied our algorithms to the same instances using only one RDC. We note that their algorithms are not for the MOMLP and since we are introducing the MOMLP for the first time, there are no algorithms in the literature that can be applied to the MOMLP directly. Zhang *et al.* (2019) introduced two algorithms to tackle the OMLP which are called the *GoodTreeTraversal* and the *Detour* algorithms. The GoodTreeTraversal algorithm is designed based on

Algorithm 4: Greedy Algorithm

Input: $G = (V, E)$, $\mathcal{V}$,c_{ij}, v_0, L.

1: $G' = G$ ▷ G' : the graph that will get updated with blockage information once they are revealed
2: $A = \emptyset$ ▷ A : set of nodes that have been assigned to an RDC.
3: $F = \emptyset$ ▷ F : set of revealed blocked edges.
4: $N = \emptyset$ ▷ N : set of visited nodes.
5: $P_l = v_0, \quad l \in \{1, ..., L\}$ ▷ P_l : position of RDC $l \in \{1, ..., L\}$.
6: $T_l = 0, \quad l \in \{1, ..., L\}$ ▷ T_l : time of the last event for RDC $l \in \{1, ..., L\}$.
7: **While** $N \neq \mathcal{V}$: **do:**
8: **for** $l \in \{1, ..., L\}$:
9: $u_l = \underbrace{\arg\min}_{u}\{SP(P_l, u, G') \quad u \in V_0 \setminus A\}$
▷ u_l : the closest node to RDC l that has not been assigned to other RDCs yet.
10: $A = A \cup u_l$
11: **End for**
12: $v_l = SP(P_l, u_l, G') \quad l \in \{1, ..., L\}$
▷ v_l : first node on the shortest path from current position of RDC l to u_l on G'.
13: $\boldsymbol{\ell} = \underbrace{\arg\min}_{l}\{T_l + c_{P_l v_l} \quad l \in \{1, ..., L\}\}$
▷ $\boldsymbol{\ell}$: the RDC that has the shortest distance to the next node on his path
14: **If** $(P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$ is blocked **then:**
a: $F = F \cup (P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}})$
b: $G' = (V, E \setminus F)$
c: go to 12
15: **Else:**
a: $T_{\boldsymbol{\ell}} = T_{\boldsymbol{\ell}} + c_{P_{\boldsymbol{\ell}}, v_{\boldsymbol{\ell}}}$
b: $P_{\boldsymbol{\ell}} = v_{\boldsymbol{\ell}}$
c: $P_{\boldsymbol{\ell}} = v_{\boldsymbol{\ell}}$
d: **If** $P_{\boldsymbol{\ell}} = u_{\boldsymbol{\ell}}$ **then:**
e: $N = N \cup u_{\boldsymbol{\ell}}$
f: go to 7
g: **Else:**
h: go to 12
i: **end if**
16: **end if**
17: **end while**

traversing "good" K-trees. In order to describe a "good" K-tree, we need to first define a K-stroll. A K-stroll on graph $G = (V, E)$ is the minimum weight path beginning at node v_0 which contains at least K different nodes. A good K-tree is then defined as a tree spanning at least K distinct nodes of the graph G including the node v_0 in a way that the total weight of the spanning tree is less than or equal to the weight of a K-stroll. In the detour algorithm, whenever the RDC reaches to a blocked edge (u, v) on the direction from u to v, the RDC should find the alternative shortest path to go from node u to v such that the alternative path does not contain any blocked edges.

These algorithms are compared on 120 instances that are provided in Zhang *et al.* (2019) and the results are summarized in Table 1 and Figure 2. In these instances, since only one RDC is considered, the CFRS

and Greedy algorithms generate the same solutions. This is because in the clustering phase of the CFRS algorithm, all the nodes are assigned to the same RDC and the routing phase of these two algorithms are exactly the same. Zhang *et al.* (2019) generated 30 instances under four scenarios and evaluated their algorithms on these instances. The number of nodes is set to 40 in all of these four scenarios. Scenarios 1 and 2 have 80 edges and scenarios 3 and 4 have 160 edges. There are 5 blocked edges ($k = 5$) in scenario 1 and 3 and 10 blocked edges ($k = 10$) in scenarios 2 and 4. Here, we mention that the number of blocked edges in the instances in Zhang *et al.* (2019) is very low and we use them only to verify that the IP-based, CFRS and Greedy algorithms outperform the GoodTreeTraversal and Detour algorithms. We note that, within a one hour time limit, the offline MLP model (OIPM) can only find the optimal solution of instances with up to 30 nodes. However, the instances in Zhang *et al.* (2019) have 40 nodes. Therefore, we have relaxed the binary variables to continuous variables between 0 and 1 and used the obtained lower bounds from this relaxation to calculate the competitive ratio values. Hence, if the exact optimal values were used, the reported competitive ratio values could be slightly better.

Hereafter, we denote the competitive ratio by CR in the article and tables. Also, we refer to the CPU run time of our device as run time in the paper and denote it by RT in the tables. Table 1 presents the average of the obtained results for the 30 instances for each of the four scenarios. For example, the average CR and run time of the Detour algorithm over the 30 instances of the the first scenario are 3.02 and 1.24, respectively. Using either the CFRS or the Greedy algorithms, the average CR over the same instances (scenario 1) is 1.39 with an average run time of 1.39 seconds. Also, the average CR of the IP-Based algorithm is 1.67 with an average run time of 3125.17 seconds in the same instances (scenario 1). For the IP-based algorithm, we had to set a time limit of one hour for the IP model to extract the order of visiting the nodes for the RDC. Since this model could not be solved optimally in most of the instances, we had to terminate the model and extract the order of visiting the nodes for the RDC once the time limit was reached. Even though this issue prevented the IP-based algorithm to show its best performance, we can see that all of our algorithms found better solutions compared to both the GoodTreeTraversal and Detour algorithms in all the scenarios. It can be observed that as the number of blocked edges and the intact edges increases, the Detour algorithm performance becomes worse. However, even in the largest instances with 10 blocked edges and 160 edges, while the average CR of the Detour algorithm goes as high as 3.72, the average CR of the IP-based algorithm is 1.76 and the average CR of the CFRS or Greedy algorithms remains under 1.35. The GoodTreeTraversal algorithm is also performing worse than either of these algorithms in all of these instances. A summary of our comparison in terms of the CR values of the algorithms for all the 120 instances is provided in Figure 2. It can be observed that in all the tested instances, on average, the IP-based, CFRS and Greedy algorithms outperformed the Detour and GoodTreeTraversal algorithms. Another observation is that the CFRS or Greedy algorithms outperform the IP-based algorithm in almost all of these instances as well.

Table 1: Results of algorithms tested on Zhang *et al.* (2019) instances

				GoodTreeTraversal		Detour		IP-Based		CFRS-Greedy	
Scenario	n	E	B	CR	RT	CR	RT	CR	RT	CR	RT
1	40	80	5	4.91	62.74	3.02	1.24	1.67	3125.17	1.39	0.06
2	40	80	10	6.24	83.78	3.22	1.78	2.44	3110.66	1.40	0.06
3	40	160	5	4.89	106.83	3.67	1.98	1.56	3156.26	1.36	0.09
4	40	160	10	6.22	136.93	3.72	2.59	1.76	3152.14	1.35	0.09

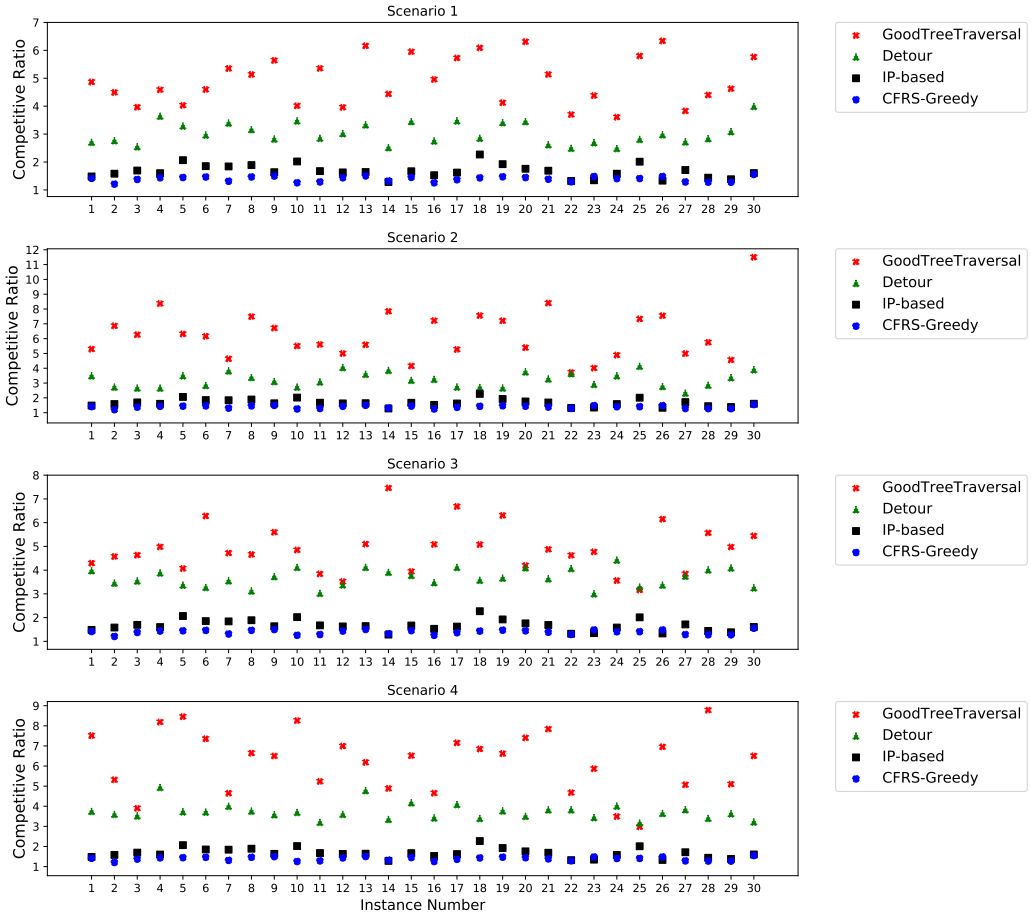

Figure 2: Comparison of algorithms

5.2. *Evaluation of the algorithms on real-life instances*

We also evaluated our algorithms on real city instances which are generated according to the road network of Istanbul city. Akbari & Salman (2017) presented three disaster affected road networks by considering four disaster scenarios and their corresponding blocked edges. We note that the blockage of the edges represent inaccessibility of them in the aftermath of a disaster. Akbari & Salman (2017) named these networks as Simplified, Detailed, and Southwest. The Simplified Istanbul network includes 74 nodes and 179 edges, which represents the main highways and their connections to major locations (Figure 3). For this road network, we give an instance of the problem in which the location of the depot, the set of critical locations together with the intact edges and the online blocked edges are given. Since for each of the road networks, we have tested 48 different instances with their specified online blocked edges and critical nodes, we cannot

show all of them. The Detailed Istanbul network contains 349 nodes and 689 edges, which also includes street road segments and various locations in the city (Figure 4). The Southwest Istanbul network models the road network of the Southwestern region of Istanbul with 250 nodes and 539 edges (Figure 5). In these road networks, the distances are calculated by ArcGIS application using the actual road distances. Since these road networks are larger than our intended size, we chose 20, 25, 30 and 35 nodes as the critical locations that should be visited. These nodes represent critical locations such as healthcare centers or hospitals, schools and emergency assembly shelters. In the emergency post-disaster response phase, the RDCs must visit these critical locations and supply their demand for relief items. Here we emphasize that in the online problem with online blocked edges, since the blocked edges are not known from the beginning, it is not possible to calculate the accurate shortest path distances between critical nodes and pre-process the problem and form a complete graph in which all the nodes should be visited. Therefore, when the number of intact and blocked edges as well as nodes increases, the problem becomes more challenging. All of our algorithms are designed in a way that they can solve both instances where all or some of the nodes should be visited. In our study, we have used four disaster scenarios with Moderate, Considerable, High, and Extreme disaster magnitudes. Table 2 represents these instances in detail. The column denoted by "Network" shows which network the instance belongs to. The columns denoted by "n" and "E" denote the number of nodes and edges in that instance, respectively. The column denoted by "Instance Name" is the name of that particular instance with given topology and blocked edges. The column denoted by "Category", identifies whether that instance belongs to the Moderate, Considerable, High or Extreme category. The last two columns denoted by "PoB" and "k" denote the percentage of blocked edges and the number of blocked edges in that instance. Since the blocked edges are non-recoverable, we simulated them in a way that the road networks are not disconnected after the blocked edges are eliminated from them. This is to ensure that the RDCs can access all the critical nodes. For different road networks, we had to adjust the percentage of the blocked edges for different scenarios to assure that they remain connected after removal of the blocked edges.

Table 2: Characteristics of the used data sets

Network	n	E	Instance Name	Category	PoB	k
Simplified	74	179	SimMod	Moderate	25%	44
			SimCon	Considerable	30%	53
			SimHg	High	35%	62
			SimExt	Extreme	40%	71
Southwest	250	539	SWMod	Moderate	20%	107
			SWCon	Considerable	25%	134
			SWHg	High	30%	161
			SWExt	Extreme	35%	188
Detailed	349	689	DetMod	Moderate	10%	68
			DetCon	Considerable	15%	103
			DetHg	High	20%	137
			DetExt	Extreme	25%	172

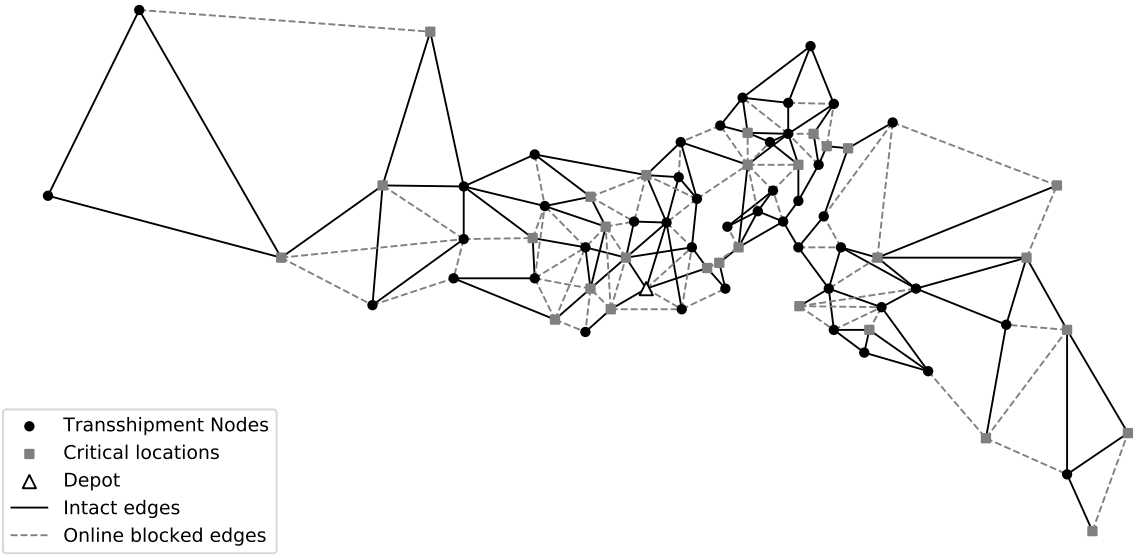

Figure 3: Representation of the Simplified road network for the SimpExt case and 30 critical nodes

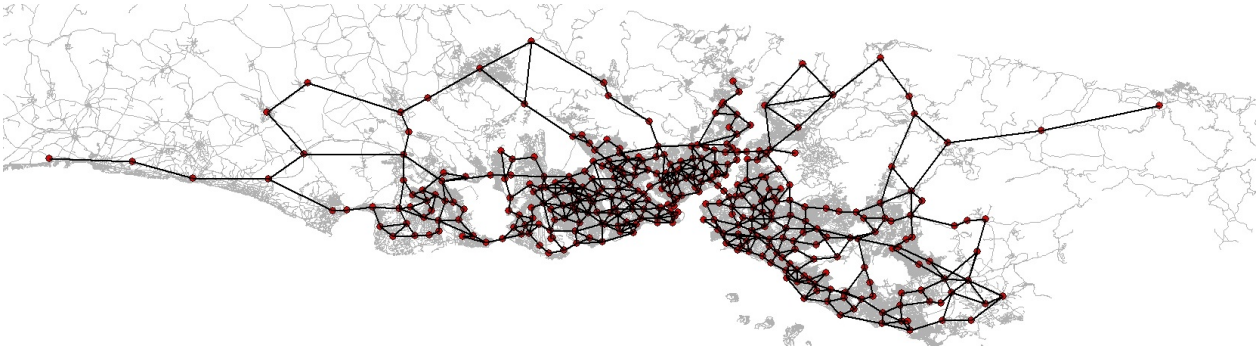

Figure 4: Representation of the Detailed road network (ArcGIS view)

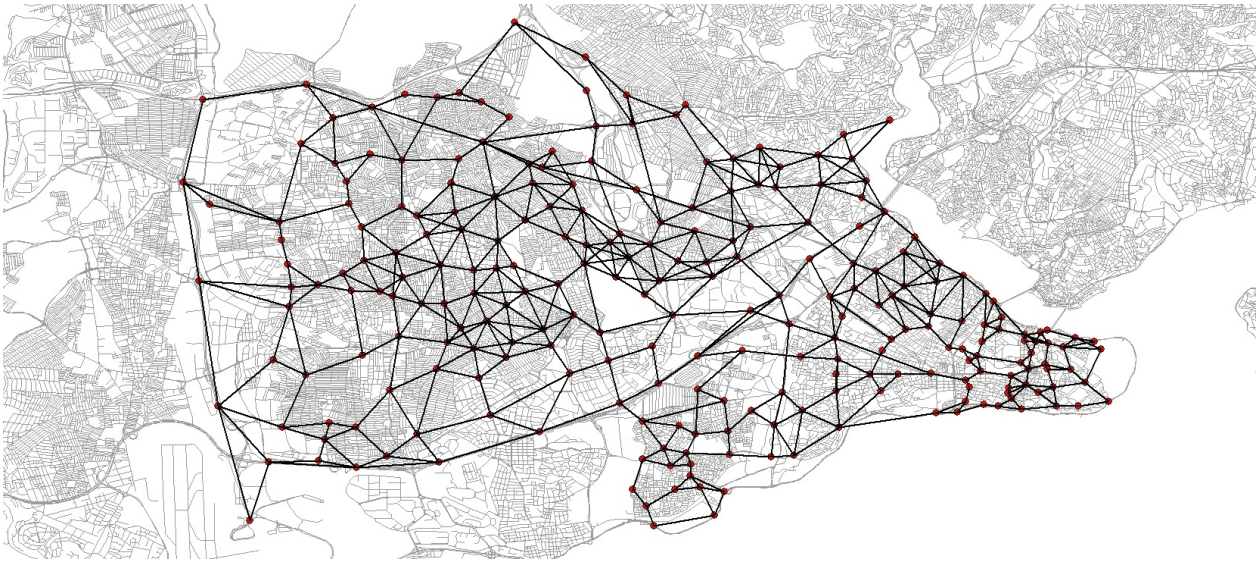

Figure 5: Representation of the Southwest road network (ArcGIS view)

5.3. Results of the Simplified network

For the simplified network, we used the given instances in Table 2 with 25, 30, 35 and 40 percent of blocked edges. These instances are referred to as SimMod, SimCon, SimHg and SimExt in the order of the percentage of blocked edges in them. For each of these categories of blocked edges, we then tested our algorithms with 20, 25, 30 and 35 nodes that should be visited. With each of these cases, we tested each instance with 2, 3, 4 and 5 RDCs. The results of our experiments are given in Table 3.

Table 3 is divided into 4 sections and the corresponding results to each of the networks SimMod, SimCon, SimHg and SimExt are given in their devoted sections. In the first column denoted by $|C|$, number of chosen critical locations of that particular instance is given. The column given by L represents the number of RDCs in that instance. The column denoted by Offline RT gives the CPU run-time of the offline model in seconds. For each of the algorithms, two columns labeled as RT and CR give the run time (in seconds) and the obtained competitive ratio for that algorithm. As can be observed, the offline model becomes more difficult to solve once the number of nodes that should be visited increases and particularly in the case where we have fewer RDCs. For example, the average run time of the offline model over the 4 networks with 2 RDCs and 35 nodes is 113.12 seconds while it is only 1.77 seconds with 5 RDCs over the same instances. The main point of this computational experiment is however to evaluate the performance of the algorithms.

From the competitive ratio (CR) perspective, the IP-based algorithm, had the best performance among the three algorithms with an average of 1.23 over all the tested instances of the Simplified instances. This is the average of the CR value obtained from testing the IP-based algorithm on the four network scenarios, with varying numbers of RDCs (from 2 to 5) and nodes that should be visited (20, 25, 30 and 35). This average CR value, over all the Simplified instances, is 1.40 for the CFRS algorithm and 1.42 for the Greedy algorithm. The amount of damage does not appear to impact the performance of the IP-based algorithm. The average calculated CR over different categories remains under 1.28 for this algorithm. The maximum reported CR for the IP-based algorithm is 1.51 on the SimHg instance with 5 RDCs and 35 nodes to be visited. For the CFRS algorithm, the average CR over the SimExt instances increases to up to 1.57. The maximum CR using the CFRS algorithm occurred in the SimHg instance with 2 RDCs and 20 nodes. The maximum average CR of the Greedy algorithm over the four different networks, is on the SimHg instances which is 1.56. The maximum calculated CR using the Greedy algorithm is on the SimHg instance with 2 RDCs and 20 nodes.

Table 3: Results of the Simplified instances

		SimMod							SimHg						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	1.32	0.44	1.13	0.05	1.65	0.06	1.20	0.34	0.38	1.19	0.04	1.29	0.06	2.05
	3	0.07	0.19	1.11	0.03	1.13	0.13	1.58	0.16	0.19	1.18	0.04	1.18	0.08	1.80
	4	0.13	0.41	1.24	0.03	1.15	0.06	1.24	0.05	0.33	1.24	0.03	1.26	0.09	1.75
	5	0.09	0.15	1.22	0.04	1.29	0.07	1.49	0.08	0.31	1.24	0.06	1.29	0.07	1.42
25	2	2.70	1.99	1.16	0.04	1.42	0.07	1.95	0.50	2.02	1.23	0.04	1.33	0.09	1.45
	3	0.38	0.53	1.10	0.04	1.65	0.08	1.42	0.54	0.57	1.26	0.05	1.26	0.10	1.72
	4	0.33	0.37	1.13	0.03	1.18	0.07	1.18	0.29	0.38	1.18	0.04	1.23	0.12	1.82
	5	0.14	0.54	1.24	0.04	1.33	0.10	1.32	0.12	0.50	1.14	0.04	1.31	0.10	1.29
30	2	44.86	13.25	1.19	0.15	1.54	0.19	1.19	5.00	5.42	1.31	0.06	1.59	0.10	1.50
	3	7.41	2.19	1.16	0.04	1.14	0.13	1.38	2.24	1.92	1.29	0.05	1.37	0.11	1.53
	4	0.86	1.58	1.15	0.06	1.21	0.12	1.28	1.10	1.32	1.29	0.04	1.37	0.10	1.24
	5	0.56	1.05	1.13	0.05	1.50	0.15	1.42	0.27	1.08	1.33	0.05	1.36	0.10	1.40
35	2	341.38	17.67	1.34	0.07	1.29	0.13	1.16	7.62	16.70	1.36	0.08	1.48	0.12	1.72
	3	6.74	15.04	1.29	0.06	1.19	0.13	1.32	32.11	14.69	1.46	0.05	1.45	0.13	1.54
	4	4.47	1.93	1.23	0.05	1.25	0.16	1.31	3.21	1.94	1.35	0.07	1.50	0.12	1.30
	5	2.72	2.30	1.35	0.06	1.43	0.13	1.21	1.28	2.10	1.51	0.05	1.72	0.16	1.41

		SimCon							SimExt						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	0.96	0.40	1.35	0.04	1.56	0.07	1.67	0.91	0.39	1.13	0.04	1.86	0.05	1.25
	3	0.23	0.18	1.18	0.04	1.36	0.07	1.59	0.18	0.17	1.13	0.04	1.29	0.08	1.42
	4	0.13	0.33	1.21	0.04	1.42	0.07	1.42	0.06	0.34	1.26	0.05	1.39	0.06	1.48
	5	0.05	0.20	1.24	0.04	1.21	0.06	1.38	0.06	0.15	1.27	0.04	1.38	0.07	1.46
25	2	5.54	2.39	1.24	0.06	1.26	0.08	1.22	2.74	2.19	1.16	0.04	1.47	0.06	1.29
	3	1.14	0.58	1.18	0.04	1.42	0.09	1.19	0.77	0.59	1.18	0.06	1.83	0.10	1.21
	4	0.34	0.38	1.20	0.04	1.20	0.08	1.39	0.47	0.38	1.12	0.05	1.43	0.07	1.42
	5	0.29	0.50	1.18	0.05	1.19	0.12	1.35	0.17	0.49	1.18	0.07	1.49	0.12	1.32
30	2	3.10	5.49	1.23	0.10	1.42	0.13	1.36	65.99	5.40	1.17	0.06	1.71	0.09	1.44
	3	2.21	1.91	1.15	0.05	1.22	0.10	1.25	1.81	1.89	1.23	0.05	1.55	0.15	1.44
	4	0.78	1.34	1.19	0.04	1.26	0.14	1.33	1.21	1.33	1.15	0.05	1.45	0.10	1.40
	5	0.53	1.13	1.18	0.06	1.30	0.12	1.46	0.75	1.05	1.20	0.05	1.64	0.11	1.43
35	2	17.87	17.25	1.29	0.08	1.28	0.14	1.52	85.58	17.10	1.41	0.10	1.58	0.19	1.40
	3	24.02	14.80	1.24	0.06	1.28	0.14	1.17	29.32	14.81	1.42	0.07	1.62	0.14	1.46
	4	1.73	2.01	1.17	0.05	1.33	0.16	1.23	3.58	2.00	1.32	0.05	1.64	0.14	1.34
	5	1.49	2.17	1.30	0.05	1.42	0.15	1.33	1.58	2.15	1.33	0.07	1.74	0.17	1.45

From a computational time perspective, the average run time of the IP-based algorithm over all the tested instances is 3.3 seconds while it is just 0.5 seconds for the CFRS algorithm and 0.11 seconds for the Greedy algorithm. The maximum run time of the IP-based algorithm is in the SimCon instance with 2 RDCs and 35 nodes which is 17.25 seconds. Overall, the run time of all the algorithms remained under 17.25 seconds for each of the instances.

Figure 6 depicts the summary of the calculated CR values from Table 3. As can be observed, the IP-based algorithm has consistently shown a better performance among these algorithms. While for the SimMod, SimCon and SimHg networks, the CFRS shows a better performance compared to the Greedy algorithm, for the SimExt category, the Greedy algorithm had a better performance compared to the CFRS algorithm.

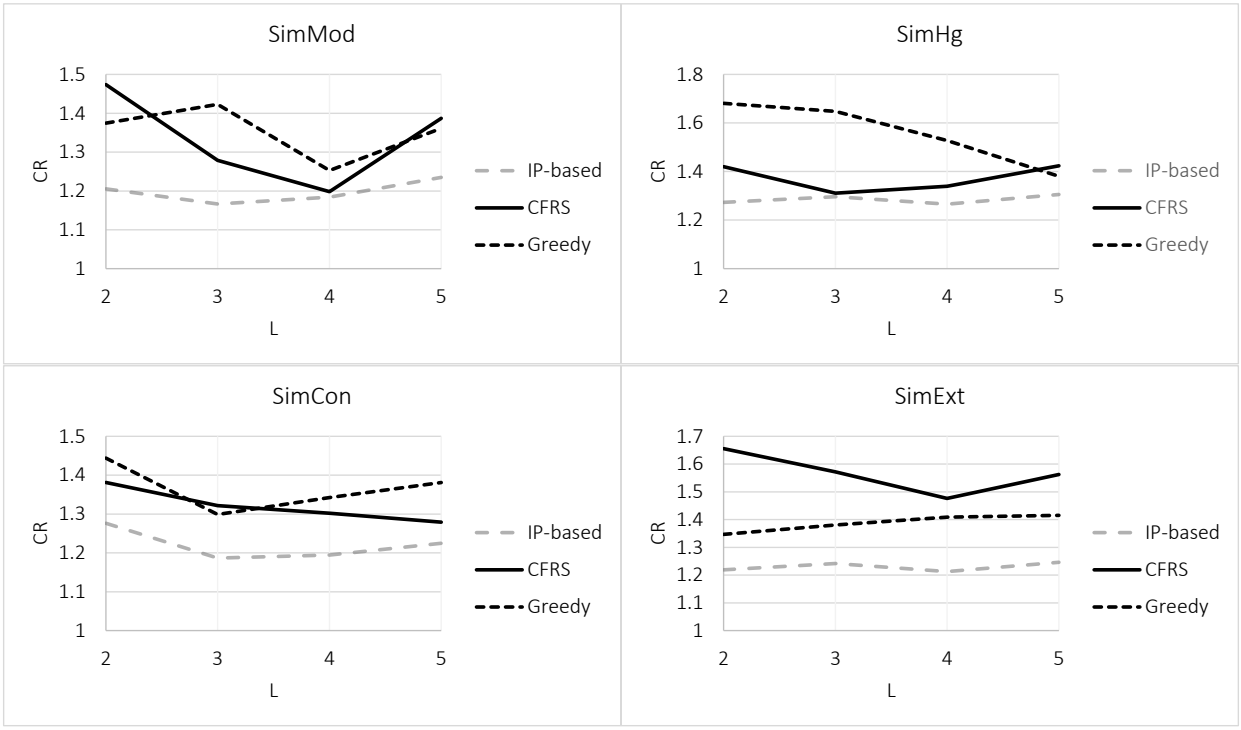

Figure 6: summary of the Simplified instances

5.4. Results of the Southwest network

For the Southwest network, we also used the SWMod, SWCon, SWHg and SWExt instances with 20, 25, 30 and 35 percent of blocked edges, respectively. More details from these instances is given in Table 2. Similar to the Simplified instances, for each of these categories of blocked edges, we tested our algorithms with 20, 25, 30 and 35 nodes that should be visited. We also tested each instance with 2, 3, 4 and 5 RDCs. The results of our experiments are given in Table 4 and Figure 7.

The columns of Table 4 are the same as Table 3 and are explained above. As can be observed, since the size of the network in the SouhWest instances is larger than the Simplified instances, the CPU run time of the offline model and the IP-based algorithm increases. The maximum CPU run time of the offline model increases to up to 2142.62 seconds for the SWMod instance with 2 RDCs and 35 nodes. The maximum reported CPU run time of the IP-based algorithm is 3101.42 seconds for the same instance with 2 RDCs and 35 nodes. Meanwhile, the maximum CPU run time of the CFRS and Greedy algorithms are only 0.42 and 1.16 seconds respectively. The average CPU run time of the IP-based algorithm over the Southwest instances is 175.5 seconds, while this average is just 0.15 and 0.36 seconds for the CFRS and Greedy algorithms, respectively.

Table 4: Results of the Southwest instances

		SWMod							SWHg						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	1.90	0.45	1.18	0.08	1.22	0.13	1.25	1.21	0.51	1.13	0.10	1.11	0.22	1.70
	3	0.41	0.45	1.34	0.12	1.47	0.29	1.18	0.41	0.45	1.38	0.11	1.34	0.19	1.62
	4	0.12	0.50	1.29	0.13	1.55	0.21	1.18	0.13	0.34	1.35	0.11	1.40	0.23	1.60
	5	0.10	0.28	1.29	0.11	1.52	0.22	1.18	0.13	0.31	1.29	0.12	1.43	0.26	1.35
25	2	1.17	2.13	1.20	0.17	1.56	0.23	1.29	2.05	2.59	1.17	0.13	1.24	0.30	1.63
	3	1.24	0.97	1.30	0.11	1.54	0.25	1.20	0.56	1.06	1.28	0.14	1.39	0.31	1.70
	4	0.54	0.53	1.30	0.13	1.54	0.31	1.22	0.45	0.60	1.30	0.11	1.38	0.29	1.23
	5	0.45	0.92	1.31	0.10	1.42	0.29	1.21	0.28	1.06	1.27	0.13	1.36	0.33	1.32
30	2	113.8	116.6	1.25	0.15	1.46	0.43	1.22	16.31	131.3	1.16	0.15	1.22	0.31	1.51
	3	1.86	3.71	1.34	0.12	1.45	0.38	1.19	2.04	4.38	1.26	0.15	1.40	0.35	1.71
	4	1.87	1.32	1.26	0.14	1.52	0.34	1.16	1.60	1.56	1.20	0.11	1.33	0.39	1.21
	5	1.41	1.46	1.29	0.14	1.38	0.39	1.22	0.64	1.53	1.21	0.14	1.35	0.43	1.30
35	2	2142	3101	1.38	0.22	1.32	0.49	1.23	390.5	2637	1.14	0.32	1.38	0.52	1.64
	3	6.41	33.05	1.42	0.20	1.53	0.58	1.40	21.30	27.54	1.15	0.34	1.36	1.16	1.54
	4	8.15	5.21	1.26	0.17	1.49	0.63	1.27	2.69	3.33	1.20	0.12	1.28	0.46	1.74
	5	2.98	2.53	1.25	0.21	1.41	0.62	1.33	2.57	1.96	1.19	0.12	1.37	0.42	1.26

		SWCon							SWExt						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	1.08	0.53	1.44	0.13	1.51	0.25	1.29	0.31	0.43	1.25	0.10	1.26	0.20	1.80
	3	0.24	0.51	1.51	0.13	1.47	0.23	1.28	0.24	0.66	1.45	0.10	1.34	0.22	2.15
	4	0.34	0.37	1.48	0.14	1.55	0.28	1.27	0.15	0.30	1.25	0.09	1.41	0.21	1.87
	5	0.13	0.50	1.45	0.23	1.55	0.38	1.27	0.13	0.24	1.27	0.12	1.44	0.21	1.46
25	2	3.73	3.02	1.39	0.15	1.45	0.27	1.33	3.57	2.15	1.45	0.12	1.71	0.19	1.90
	3	1.02	1.26	1.36	0.18	1.55	0.38	1.24	1.59	0.94	1.53	0.11	1.61	0.20	1.85
	4	0.75	0.56	1.41	0.14	1.55	0.28	1.37	0.66	0.70	1.51	0.16	1.54	0.29	1.64
	5	0.41	1.00	1.42	0.16	1.48	0.36	1.40	0.23	0.93	1.41	0.10	1.52	0.30	1.78
30	2	47.11	128.6	1.34	0.14	1.48	0.34	1.23	20.45	114.4	1.43	0.17	1.72	0.31	1.97
	3	20.33	3.62	1.23	0.14	1.52	0.36	1.35	7.76	4.07	1.52	0.25	1.72	0.34	1.87
	4	2.96	2.60	1.36	0.15	1.59	0.34	1.36	1.63	1.44	1.43	0.15	1.94	0.33	1.61
	5	1.20	1.64	1.38	0.13	1.49	0.35	1.33	1.31	1.24	1.41	0.11	1.63	0.34	1.68
35	2	922.5	2303	1.41	0.18	1.42	0.50	1.23	652.9	2494	1.43	0.21	1.86	0.44	1.84
	3	44.18	29.15	1.24	0.18	1.49	0.46	1.34	20.54	30.25	1.40	0.20	2.20	0.45	1.79
	4	18.57	4.42	1.35	0.16	1.49	0.49	1.35	9.46	3.85	1.42	0.17	1.88	0.48	2.03
	5	2.62	2.40	1.40	0.19	1.55	0.51	1.36	3.14	4.01	1.51	0.42	1.91	1.10	1.93

From the CR perspective, the IP-based had a better performance by an average of 1.33. The average CR value of the CFRS and Greedy algorithms are 1.49 and 1.46 over all the Southwest instances, respectively. The variance of the performance of the IP-based algorithm is lower and equal to 0.01 with a maximum CR of 1.53 for the SWExt instance with 25 nodes and 3 RDCs. The variance of the CFRS and Greedy algorithms are higher and the maximum calculated CR value of the CFRS algorithm is 2.20 for the SWExt instance with 35 nodes and 3 RDCs. The maximum CR of the Greedy algorithm is 2.15 for the SWExt instance with 20 nodes and 3 RDCs. In general, as expected, once the percentage of the blocked edges increases, the performance of the algorithms will be negatively impacted. For instance, for the Greedy algorithm, the average calculated CR is 1.23, 1.31, 1.50 and 1.82 for the SWMod, SWCon, SWHg and SWExt instances,

respectively.

Figure 7 further illustrates and compares the performance of the three algorithms over different instances of the Southwest network with different categories. A different trend from the Simplified instances can be observed in here. The Greedy algorithm, outperforms the IP-based algorithm for the Moderate and Considerable categories. In these instances, the Greedy algorithm outperforms both the IP-based and the CFRS algorithms. However, the number of blocked edges and the severity of the scenarios seem to have a negative impact on the performance of the Greedy algorithm. In particular, for the SWHg and SWExt instances, the calculated CR values from the Greedy algorithm are higher than those of the CFRS and IP-based algorithms. Moreover, in both SWHg and SWExt instances, the IP-based algorithm has consistently performed better than the CFRS algorithm.

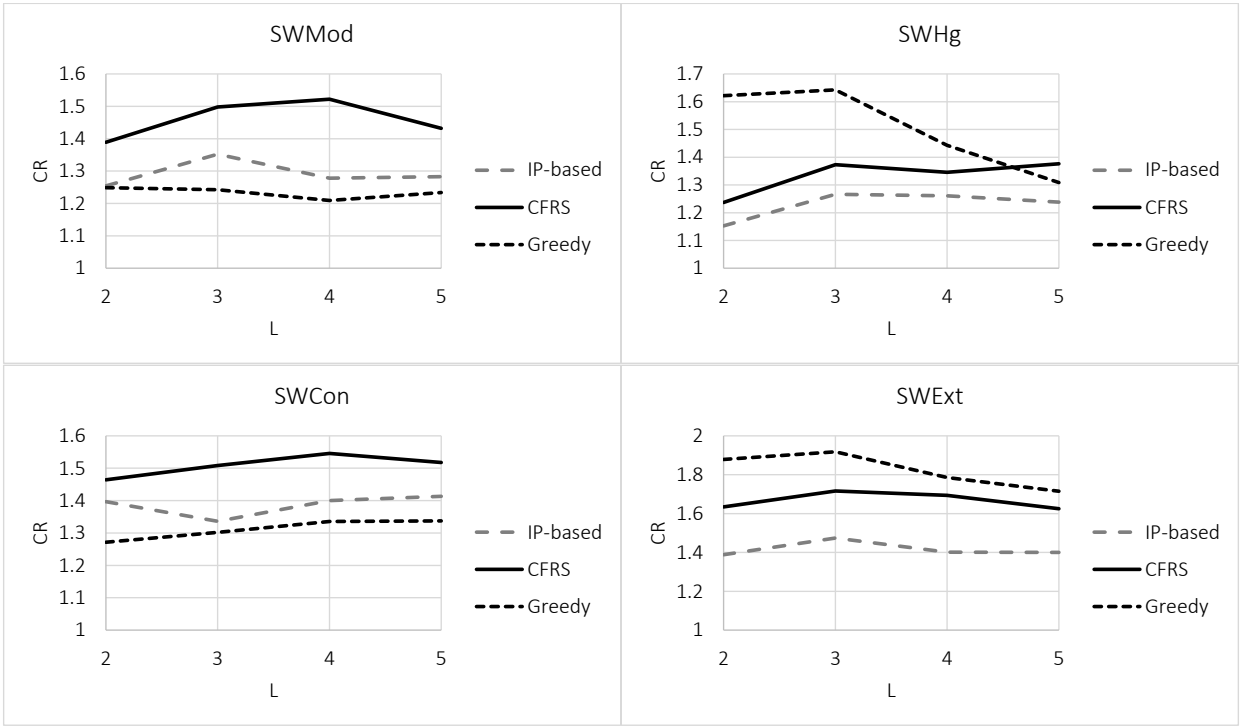

Figure 7: summary of the Southwest instances

5.5. Results of the Detailed network

For the Detailed network, we used the DetMod, DetCor, DetHg and DetExt instances given in Table 2. For each of these instances, we first selected 20, 25, 30 and 35 nodes to be visited and then tested our algorithms with 2, 3, 4 and 5 RDCs. The results of these experiments are given in Table 5.

For the Detailed network, similar to the Southwest instances, the run times of the offline model and the IP-based algorithm are larger than those of the Simplified network. The maximum CPU run times of the offline model and the IP-based algorithm are 2406.72 and 429.16 seconds, respectively. However, similar to other networks, the maximum CPU run times of the CFRS and Greedy algorithms are significantly smaller and are equal to 0.68 and 2.43 seconds, respectively.

From the CR perspective, the IP-based algorithm has consistently performed better and had an average of 1.28 over all the Detailed instances with a maximum CR of 1.50 for the DetCon instance with 30 nodes and 2 RDCs. The CFRS algorithm has also performed consistently better than the Greedy algorithm. While the average calculated CR values for the CFRS algorithm are 1.40, 1.64, 1.40 and 1.47 for the DetMod, DetCon, DetHg and DetExt instances, these average CR values are 1.72, 2.04, 1.81 and 1.94 for the Greedy algorithm over the same instances, respectively.

Table 5: Results of the Detailed instances

		DetMod							DetHg						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	0.33	0.60	1.24	0.14	1.30	0.27	1.49	0.26	0.66	1.20	0.19	1.34	0.30	1.64
	3	0.30	0.48	1.12	0.12	1.25	0.34	1.47	0.16	0.37	1.11	0.21	1.38	0.36	1.65
	4	0.09	0.24	1.25	0.14	1.30	0.29	1.55	0.07	0.34	1.17	0.17	1.45	0.66	1.86
	5	0.12	0.24	1.25	0.12	1.15	0.29	1.62	0.06	0.34	1.19	0.22	1.23	0.66	1.88
25	2	2.17	3.14	1.06	0.17	1.39	0.32	1.44	1.64	3.43	1.23	0.22	1.25	0.47	1.69
	3	0.85	0.77	1.09	0.16	1.49	0.38	1.64	0.60	0.88	1.15	0.22	1.49	0.52	1.59
	4	0.26	0.59	1.20	0.13	1.21	0.46	1.83	0.37	0.69	1.20	0.17	1.31	0.62	1.78
	5	0.34	0.68	1.20	0.12	1.19	0.41	1.49	0.28	0.65	1.14	0.15	1.22	0.66	1.62
30	2	3.09	2.75	1.07	0.19	1.40	0.44	1.97	10.28	3.34	1.46	0.33	1.43	0.53	1.51
	3	1.10	1.71	1.23	0.20	2.10	0.50	2.14	1.90	2.12	1.33	0.25	1.85	0.65	1.86
	4	0.68	0.98	1.22	0.16	1.29	0.56	1.84	0.90	1.20	1.30	0.20	1.53	0.64	2.00
	5	0.73	0.99	1.22	0.15	1.66	0.59	1.68	1.14	1.23	1.31	0.21	1.49	0.81	1.72
35	2	113.3	318.3	1.19	0.30	1.29	0.61	1.74	2406	355.1	1.21	0.27	1.31	0.64	1.70
	3	13.41	10.71	1.14	0.25	1.47	0.69	2.02	39.17	11.79	1.16	0.47	1.39	0.82	2.06
	4	2.45	10.81	1.09	0.32	1.41	0.89	1.92	6.42	12.26	1.27	0.22	1.43	0.73	2.20
	5	1.71	2.34	1.05	0.21	1.44	0.76	1.69	3.54	2.38	1.23	0.18	1.40	0.82	2.13

		DetCon							DetExt						
		Offline	IP-based		CFRS		Greedy		Offline	IP-based		CFRS		Greedy	
$\|C\|$	L	RT	RT	CR	RT	CR	RT	CR	RT	RT	CR	RT	CR	RT	CR
20	2	0.43	0.57	1.38	0.19	1.55	0.24	1.79	1.35	0.56	1.39	0.18	1.34	0.24	1.67
	3	0.48	0.34	1.25	0.16	1.64	0.28	1.84	0.66	0.50	1.21	0.24	1.38	0.40	1.64
	4	0.22	0.27	1.34	0.16	1.79	0.40	2.02	0.17	0.31	1.35	0.19	1.42	0.35	1.75
	5	0.11	0.25	1.34	0.18	1.61	0.42	1.97	0.19	0.29	1.38	0.17	1.34	0.54	1.80
25	2	1.11	2.83	1.42	0.21	1.77	0.44	2.12	1.94	3.29	1.29	0.36	1.53	0.45	2.01
	3	0.99	0.78	1.46	0.22	1.91	0.52	2.31	0.62	0.81	1.43	0.21	1.75	0.51	1.99
	4	0.31	0.66	1.48	0.18	1.79	0.52	2.06	0.72	0.83	1.45	0.17	1.45	0.52	1.73
	5	0.30	0.60	1.32	0.20	1.63	0.64	2.05	0.27	0.73	1.43	0.17	1.47	0.58	1.74
30	2	3.47	3.32	1.50	0.34	1.60	0.50	1.70	51.98	3.11	1.38	0.28	1.37	0.57	1.60
	3	1.98	1.80	1.41	0.27	1.84	0.56	2.05	3.65	1.79	1.38	0.27	1.72	0.49	1.75
	4	0.74	1.03	1.45	0.19	1.79	0.62	2.14	3.21	1.03	1.46	0.18	1.67	0.57	1.92
	5	1.02	1.09	1.46	0.20	1.69	0.70	2.10	0.75	1.16	1.49	0.19	1.55	0.58	1.93
35	2	257.6	335.7	1.39	0.36	1.28	0.88	1.89	911.2	429.6	1.29	0.41	1.28	0.86	2.43
	3	51.25	12.11	1.38	0.33	1.53	0.89	2.18	13.19	12.31	1.24	0.68	1.47	1.60	2.57
	4	6.64	13.23	1.20	0.25	1.46	0.94	2.28	4.79	23.62	1.40	0.68	1.48	2.42	2.07
	5	3.65	2.77	1.14	0.32	1.49	1.08	2.07	4.09	4.62	1.41	0.37	1.41	1.30	2.26

Figure 8 illustrates the average performance of the algorithms on the Detailed instances further. As can be observed, over the Detailed instances, the IP-based algorithm has consistently performed better and achieves lower average CR values over all the networks with different number of RDCs. Comparison of the

CFRS and Greedy algorithms confirms that the CFRS algorithm is outperforming the Greedy algorithm in all the different categories with different number of RDCs.

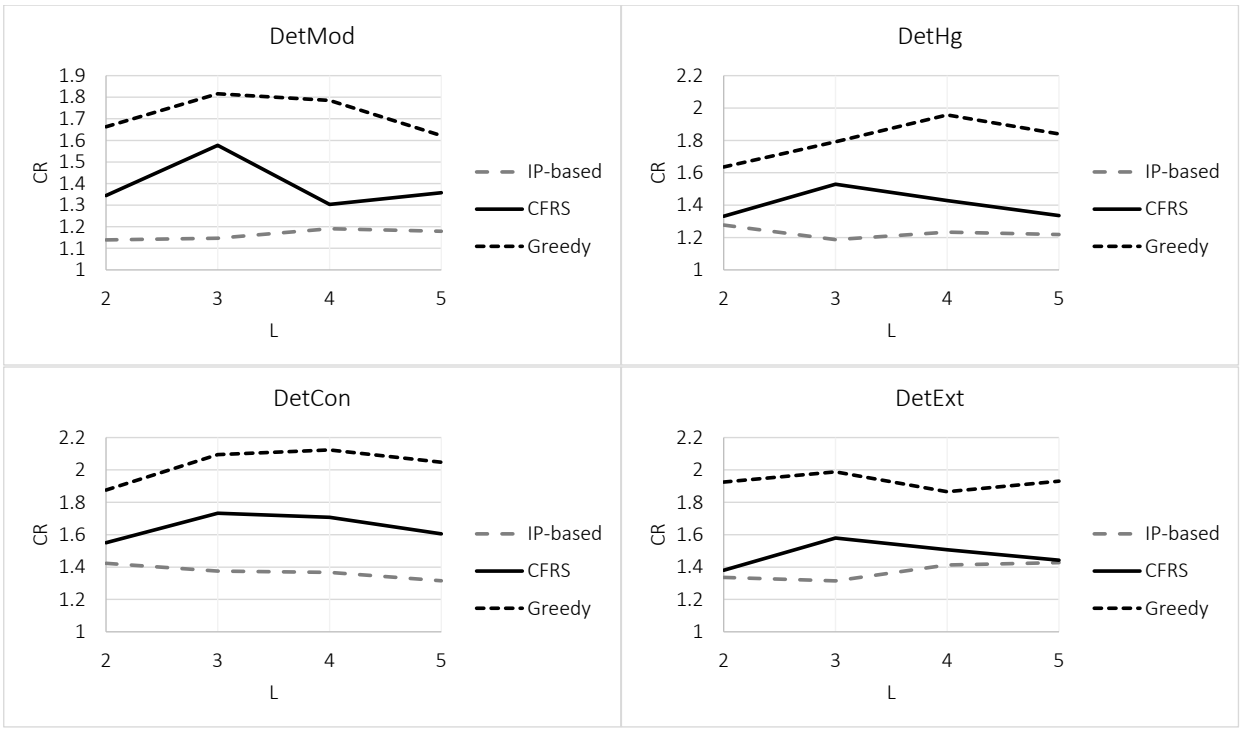

Figure 8: summary of the Detailed instances

5.6. Summary of the computational results

In our computational study, we first demonstrated that the IP-based, CFRS and Greedy algorithms outperform algorithms from the literature in the OMLP instances. We then tested our algorithms on real-life instances from the road network of Istanbul city in Turkey. The summary of our computational results are given in Tables 6 and 7.

Table 6 gives the average results of testing our algorithms with respect to the number of RDCs. These results are the average results over three road networks from Istanbul city that were experimented under four blockage scenarios as; moderate, considerable, high and extreme. The RT column gives the average CPU run time of each of the algorithms in seconds and the CR column gives the average CR value obtained from each of the algorithms. The CPU run time of the IP-based algorithm is more than the other two algorithms but it decreases once the number of RDCs increases. The average CPU run time of the CFRS and Greedy algorithms remains under 0.4 seconds. The obtained average CR of the IP-based algorithm is the lowest among these three algorithms. It remained under 1.30 even for the largest instances with 5 RDCs. It can be observed that, in the IP-based algorithm, the average CR increases once the number of RDCs increases. For the CFRS and Greedy algorithms however, the maximum CR occurred in the instances with 3 RDCs. Moreover, the CFRS algorithm, showed a better performance compared to the Greedy algorithm in average.

Table 6: The average relation between competitive ratio and number of RDCs

	IP-based		CFRS		Greedy	
L	RT	CR	RT	CR	RT	CR
2	262.809	1.278	0.160	1.438	0.302	1.581
3	5.658	1.279	0.159	1.500	0.363	1.629
4	2.334	1.283	0.133	1.447	0.383	1.586
5	1.214	1.290	0.136	1.445	0.399	1.548

Table 7 gives the summary of the results with respect to the number of nodes that were visited in each instance. As can be observed, the average CPU run time of all the algorithms increases as the number of visited nodes increases. This increase for the IP-based algorithm is considerable and once the number of nodes that should be visited goes beyond 50, we might not be able to extract a solution from this algorithm in a logical time limit. The CFRS and Greedy algorithms, remain tractable even for the largest instances confirming their capacities in terms of solving larger instances. In general, it can be observed that once the number of nodes increases, the reported CR from the IP-based algorithm also increases, but it still remains lower than those of the CFRS and Greedy algorithms.

Table 7: The average relation between competitive ratio and the visited nodes

	IP-based		CFRS		Greedy	
$\|C\|$	RT	CR	RT	CR	RT	CR
20	0.369	1.268	0.111	1.390	0.227	1.568
25	1.170	1.272	0.124	1.452	0.292	1.562
30	12.407	1.293	0.144	1.517	0.354	1.560
35	258.069	1.298	0.209	1.473	0.573	1.653

Overall, we can conclude that the IP-based algorithm finds better solutions if its CPU run time is not a constraint. Another constraint with the IP-based algorithm is that it requires an optimization solver which might restrict its applications. Among the CFRS and Greedy algorithms, the CFRS appeared to show a better average performance as well as lower CPU running times.

6. Conclusions and future research directions

In this study, we addressed the online multiple minimum latency problem with edge uncertainty (MOMLP) for the first time. In the MOMLP, a set of L RDCs should visit a number of nodes while minimizing the overall waiting time of the first visits to each node. Furthermore, k ($k > L$) edges are blocked but their blockages are not known in advance. Moreover, in our study, the blocked edges are non-recoverable and the RDCs cannot use them in their routes. The blockage of an edge is learned online when one of the RDCs visits an end-node of that blocked edge. Once this information is revealed to one of the RDCs, it is then immediately communicated between the RDCs and all of them will know about that blockage.

We prove a lower bound of $2\lfloor \frac{k}{L} \rfloor + 1$ on the competitive ratio of deterministic online algorithms for this problem. Moreover, we show that the competitive ratio of the optimal deterministic online algorithm for this problem lies in the interval between $2\lfloor \frac{k}{L} \rfloor + 1$ and $2k + 1$. This is by showing that $2k + 1$ is an upper bound on the competitive ratio of the optimal deterministic online algorithm for the MOMLP. With the aim of providing fast and efficient algorithms for realistic-sized problems defined on real-life road networks, we proposed three heuristic algorithms for the MOMLP. These algorithms are called, the IP-based algorithm, the Cluster First Route Second (CFRS) algorithm and the Greedy algorithm. We first showed how our algorithms outperform those from the literature on a special case of our problem with only one RDC and then conducted computational experiments on numerous cases of three real-life instances adopted from the road network of Istanbul city in Turkey. Our computational experiments verify the performance of the developed algorithms as we compared the results of our algorithms with those of the offline optimum. We also present a summary of our computational experiments in which we compare the developed algorithms in terms of their performance and draw conclusions on which one might perform better under what circumstances.

Online optimization enables us to give decisions under lack of some information and to represent situations where some data are revealed dynamically. Such situations arise in many practical applications, in addition to disaster management. For instance, in providing on-site services as in the traveling repairman problem, the technician routing and scheduling problems, and home healthcare routing and scheduling, the service times at different customer locations are actually revealed online since the service provider can assess the required service time after close observation at the customer location (Jaillet & Wagner 2008a). In city logistics and last mile delivery, often traffic conditions cause significant delays in some roads that can only be observed online (Zhang *et al.* 2019). Taking an online optimization approach to address such problems may lead to interesting new studies.

One immediate future research possibility would be to consider the weight of each of the critical nodes while minimizing their total latency. This weight could for example represent the population or the importance of that critical node. In this case, the problem changes to the weighted MOMLP and the offline problem becomes more complex as well. Another possible future research direction would be to consider different communication levels among the RDCs. This means that once a blockage is revealed to one of the RDCs, it might not be revealed to all of them immediately for various reasons such as jamming or lack of equipment. The case where the blocked edges are recoverable can also be addressed in the future studies. This means that the nature of the problem changes to a road restoration or road clearance problem in a post-disaster situation. The offline variations of this road clearance problem considering a single road restoration team (Kasaei & Salman (2016) and Moreno *et al.* (2019)) or multiple road restoration teams (Morshedlou *et al.* (2021) and Akbari *et al.* (2021a)) have been addressed in the literature extensively. While in these offline problems, the location of the blocked edges and the amount of required equipment/time to unblock

them is known in advance, in their online versions, both the location of the blocked edges and the amount of required time to unblock them (Akbari *et al.* 2021b), can be unknown initially and have an online nature in the sense that the unknown information is revealed incrementally over time.

References

Afrati, Foto N., Cosmadakis, Stavros S., Papadimitriou, Christos H., Papageorgiou, George, & Papaconstantinou, Nadia. 1986. The complexity of the Traveling Repairman Problem. *RAIRO-Theoretical Informatics and Applications*, **20**, 79—87.

Ajam, Meraj, Akbari, Vahid, & Salman, F. Sibel. 2019. Minimizing latency in post-disaster road clearance operations. *European Journal of Operational Research*, **277**(3), 1098 – 1112.

Ajam, Meraj, Akbari, Vahid, & Salman, F. Sibel. 2021. Routing multiple work teams to minimize latency in post-disaster road network restoration. *European Journal of Operational Research*.

Akbari, Vahid, & Salman, F. Sibel. 2017. Multi-vehicle prize collecting arc routing for connectivity problem. *Computers & Operations Research*, **82**, 52–68.

Akbari, Vahid, & Shiri, Davood. 2021. Weighted online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, **295**(1), 51–65.

Akbari, Vahid, Sadati, Mir Ehsan Hesam, & Kian, Ramez. 2021a. A decomposition-based heuristic for a multicrew coordinated road restoration problem. *Transportation Research Part D: Transport and Environment*, **95**, 102854.

Akbari, Vahid, Shiri, Davood, & Sibel Salman, F. 2021b. An online optimization approach to post-disaster road restoration. *Transportation Research Part B: Methodological*, **150**, 1–25.

Albers, Susanne. 2003. Online algorithms: A survey. *Mathematical Programming*, **97**, 3–26.

Angel-Bello, F., Cardona-Valdes, Y., & Alvarez, A. 2019. Mixed integer formulations for the multiple minimum latency problem. *Operational Research*, **19**, 369–398.

Angel-Bello, Francisco, Alvarez, Ada, & García, Irma. 2013. Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, **37**(4), 2257–2266.

Avci, Mualla Gonca, & Avci, Mustafa. 2019. An adaptive large neighborhood search approach for multiple traveling repairman problem with profits. *Computers & Operations Research*, **111**, 367 – 385.

Bender, Marko, & Westphal, Stephan. 2015. An optimal randomized online algorithm for the k-Canadian Traveller Problem on node-disjoint paths. *Journal of Combinatorial Optimization*, **30**, 87—96.

Bianco, Lucio, Mingozzi, Aristide, & Ricciardelli, Salvatore. 1993. The traveling salesman problem with cumulative costs. *Networks*, **23**(2), 81–91.

Bock, Stefan, & Klamroth, Kathrin. 2019. Combining Traveling Salesman and Traveling Repairman Problems: A multi-objective approach based on multiple scenarios. *Computers & Operations Research*, **112**, 104766.

Bruni, M.E., Khodaparasti, S., & Beraldi, P. 2020. The selective minimum latency problem under travel time variability: An application to post-disaster assessment operations. *Omega*, **92**, 102154.

Bulhões, Teobaldo, Sadykov, Ruslan, & Uchoa, Eduardo. 2018. A branch-and-price algorithm for the Minimum Latency Problem. *Computers & Operations Research*, **93**, 66–78.

Cinar, Didem, Gakis, Konstantinos, & Pardalos, Panos M. 2016. A 2-phase constructive algorithm for cumulative vehicle routing problems with limited duration. *Expert Systems with applications*, **56**, 48–58.

Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C.R., & Vansteenwegen, P. 2013. Heuristics for the traveling repairman problem with profits. *Computers & Operations Research*, **40**(7), 1700–1707.

Fischetti, Matteo, Laporte, Gilbert, & Martello, Silvano. 1993. The Delivery Man Problem and Cumulative Matroids. *Operations Research*, **41**, 1055—1064.

Geetha, Shanmugam, Vanathi, PT, & Poonthalir, Ganesan. 2012. Metaheuristic approach for the multi-depot vehicle routing problem. *Applied Artificial Intelligence*, **26**(9), 878–901.

Hu, Shaolong, Han, Chuanfeng, Dong, Zhijie Sasha, & Meng, Lingpeng. 2019. A multi-stage stochastic programming model for relief distribution considering the state of road network. *Transportation Research Part B: Methodological*, **123**, 64–87.

Hu, Zhi-Hua, Sheu, Jiuh-Biing, & Xiao, Ling. 2014. Post-disaster evacuation and temporary resettlement considering panic and panic spread. *Transportation Research Part B: Methodological*, **69**, 112 – 132.

Jaillet, P., & Wagner, M.R. 2008a. Online Vehicle Routing Problems: A Survey. *Chap. 10, pages 221–237 of:* Golden B., Raghavan S., Wasil E. (ed), *The Vehicle Routing Problem: Latest Advances and New Challenges*. Boston, MA: Springer.

Jaillet, Patrick, & Wagner, Michael R. 2006. Online Routing Problems: Value of Advanced Information as Improved Competitive Ratios. *Transportation Science*, **40**(2), 200–210.

Jaillet, Patrick, & Wagner, Michael R. 2008b. Generalized Online Routing: New Competitive Ratios, Resource Augmentation, and Asymptotic Analyses. *Operations research*, **56**, 745–757.

Kasaei, Maziar, & Salman, F. Sibel. 2016. Arc routing problems to restore connectivity of a road network. *Transportation Research Part E: Logistics and Transportation Review*, **95**, 177–206.

Liao, Chung-Shou, & Huang, Yamming. 2014. The Covering Canadian Traveler Problem. *Theoretical Computer Science*, **530**, 80—88.

Liu, Bingsheng, Sheu, Jiuh-Biing, Zhao, Xue, Chen, Yuan, & Zhang, Wei. 2020. Decision making on post-disaster rescue routing problems from the rescue efficiency perspective. *European Journal of Operational Research*, **286**(1), 321 – 335.

Liu, Yajie, Lei, Hongtao, Wu, Zhiyong, & Zhang, Dezhi. 2019. A robust model predictive control approach for post-disaster relief distribution. *Computers & Industrial Engineering*, **135**, 1253 – 1270.

Luo, Zhixing, Qin, Hu, & Lim, Andrew. 2014. Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. *European Journal of Operational Research*, **234**(1), 49–60.

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

Martínez-Salazar, Iris, Angel-Bello, Francisco, & Alvarez, Ada. 2015. A customer-centric routing problem with multiple trips of a single vehicle. *Journal of the Operational Research Society*, **66**(8), 1312–1323.

Mladenović, Nenad, Urošević, Dragan, & Hanafi, Saïd. 2013. Variable neighborhood search for the travelling deliveryman problem. *4OR*, **11**(1), 57–73.

Moreno, Alfredo, Alem, Douglas, Ferreira, Deisemara, & Clark, Alistair. 2018. An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research*, **269**(3), 1050–1071.

Moreno, Alfredo, Munari, Pedro, & Alem, Douglas. 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, Nazanin, Barker, Kash, González, Andrés D., & Ermagun, Alireza. 2021. A Heuristic Approach to an Interdependent Restoration Planning and Crew Routing Problem. *Computers & Industrial Engineering*, 107626.

Muritiba, Albert Einstein Fernandes, Bonates, Tibérius O., Da Silva, Stênio Oliveira, & Iori, Manuel. 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.

Najafi, Mehdi, Eshghi, Kourosh, & Dullaert, Wout. 2013. A multi-objective robust optimization model for logistics planning in the earthquake response phase. *Transportation Research Part E: Logistics and Transportation Review*, **49**(1), 217 – 249.

Ngueveu, Sandra Ulrich, Prins, Christian, & Wolfler Calvo, Roberto. 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, **37**(11), 1877 – 1885. Metaheuristics for Logistics and Vehicle Routing.

Ozdamar, Linet, Ekinci, Ediz, & Küçükyazici, Beste. 2004. Emergency Logistics Planning in Natural Disasters. *Annals of Operations Research*, **129**(1), 217 – 245.

Papadimitriou, Christos, & Yannakakis, Mihalis. 1991. Shortest paths without a map. *Theoretical Computer Science*, **84**, 127—150.

Picard, Jean-Claude, & Queyranne, Maurice. 1978. The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling. *Operations Research*, **26**(1), 86–110.

Pérez-Rodríguez, Noel, & Holguín-Veras, José. 2016. Inventory-Allocation Distribution Models for Postdisaster Humanitarian Logistics with Explicit Consideration of Deprivation Costs. *Transportation Science*, **50**(4), 1261–1285.

Rawls, Carmen G., & Turnquist, Mark A. 2010. Pre-positioning of emergency supplies for disaster response. *Transportation Research Part B: Methodological*, **44**(4), 521–534.

Reed, Martin, Yiannakou, Aliki, & Evering, Roxanne. 2014. An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, **15**, 169–176.

Saharidis, George K. D., Dimitropoulos, Charalampos, & Skordilis, Erotokritos. 2014. Minimizing waiting times at transitional nodes for public bus transportation in Greece. *Operational Research*, **14**, 341–359.

Sheu, Jiuh-Biing. 2007. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, **43**(6), 687 – 709. Challenges of Emergency Logistics Management.

Shiri, Davood, & Salman, F. Sibel. 2017. On the online multi-agent o-d k-Canadian Traveler Problem. *Journal of Combinatorial Optimization*, **34**, 453–461.

Shiri, Davood, & Salman, F. Sibel. 2019a. Competitive analysis of randomized online strategies for the online multi-agent *k*-Canadian Traveler Problem. *Journal of Combinatorial Optimization*, **37**, 848–865.

Shiri, Davood, & Salman, F. Sibel. 2019b. On the randomized online strategies for the *k*-Canadian traveler problem. *Journal of Combinatorial Optimization*, **38**, 254–267.

Shiri, Davood, & Salman, F. Sibel. 2020. Online Optimization of First-responder Routes in Disaster Response Logistics. *IBM Journal of Research and Development*, **64**, 1–9.

Shiri, Davood, Akbari, Vahid, & Salman, F. Sibel. 2020. Online routing and scheduling of search-and-rescue teams. *OR Spectrum*, **42**(3), 755–784.

Sleator, Daniel, & Tarjan, Robert. 1985. Amortized efficiency of list update and paging rules. *Communications of the ACM*, **28**, 202—208.

Tzeng, Gwo-Hshiung, Cheng, Hsin-Jung, & Huang, Tsung Dow. 2007. Multi-objective optimal planning for designing relief delivery systems. *Transportation Research Part E: Logistics and Transportation Review*, **43**(6), 673 – 686. Challenges of Emergency Logistics Management.

Wang, Haijun, Du, Lijing, & Ma, Shihua. 2014. Multi-objective open location-routing model with split delivery for optimized relief distribution in post-earthquake. *Transportation Research Part E: Logistics and Transportation Review*, **69**, 160 – 179.

Wei, Xiaowen, Qiu, Huaxin, Wang, Dujuan, Duan, Jiahui, Wang, Yanzhang, & Cheng, T.C.E. 2020. An integrated location-routing problem with post-disaster relief distribution. *Computers & Industrial Engineering*, **147**, 106632.

Westphal, Stephan. 2008. A note on the k-Canadian Traveller Problem. *Information Processing Letters*, **106**(3), 87 – 89.

Zhang, Huili, & Xu, Yinfeng. 2018. Online covering salesman problem. *Journal of Combinatorial Optimization*, **35**, 941—954.

Zhang, Huili, Xu, Yinfeng, & Qin, Lan. 2013. The k-Canadian Travelers Problem with communication. *Journal of Combinatorial Optimization*, **26**, 251—265.

Zhang, Huili, Tong, Weitian, Xu, Yinfeng, & Lin, Guohui. 2015. The Steiner Traveling Salesman Problem with online edge blockages. *European Journal of Operational Research*, **243**, 30—40.

Zhang, Huili, Tong, Weitian, Xu, Yinfeng, & Lin, Guohui. 2016. The Steiner Traveling Salesman Problem with online advanced edge blockages. *Computers & Operations Research*, **70**, 26—38.

Zhang, Huili, Tong, Weitian, Lin, Guohui, & Xu, Yinfeng. 2019. Online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, **273**, 418–429.

Appendix A. The IP formulation to solve the offline MOMLP

As it is stated in the description of the MBR algorithm in Section 3 and in the description of the IP-Based algorithm in Section 4.1, we have used the IP formulation proposed in Angel-Bello *et al.* (2019) to solve

the offline MOMLP, i.e., the multiple minimum latency problem (MMLP). We also referred to this model as Offline Integer Programming Model (OIPM) in Section 4.1. In the MMLP, all of the L RDCs are initially positioned in the depot node denoted by v_0. They all dispatch at time zero and the collection of their routes should visit all the nodes from v_1 to v_n while minimizing the total latency of the nodes. This formulation is an offline optimization model which assumes that there is no blocked edge in the graph. Furthermore, we point out that Angel-Bello *et al.* (2019) proposed 5 mathematical models to solve the MMLP, and among them, we have selected the last one as it gave better results based on their computational experiments. We note that since this model is designed to solve the multiple MLP, it can also solve the single MLP by setting the number of RDCs equal to one ($L = 1$).

Given that the level based model is designed in a way that calculation of the arrival times to the nodes is not necessary, this model has shown better computational performance compared to alternative models. In this model, only a set of binary variables given as x_{ij}^r are defined. x_{ij}^r equals to 1 if and only if the arc (v_i, v_j) is used to link node v_i at level $r + 1$ with node v_j at level r. A representation of this level-based structure is given in Figure A.9. Since in the optimal solution, each RDC visits at least one node, the number of levels at which a node is visited is limited to $M = n - L + 1$, where n shows the number of nodes to be visited and L gives the number of RDCs. Then the objective function of the latency minimization objective can be given as A.1. Using this level based model, the objective calculates the accumulation of the times in which each of the nodes are visited which is equivalent to the minimum latency objective. The constraints are presented in Equations from A.2 to A.7.

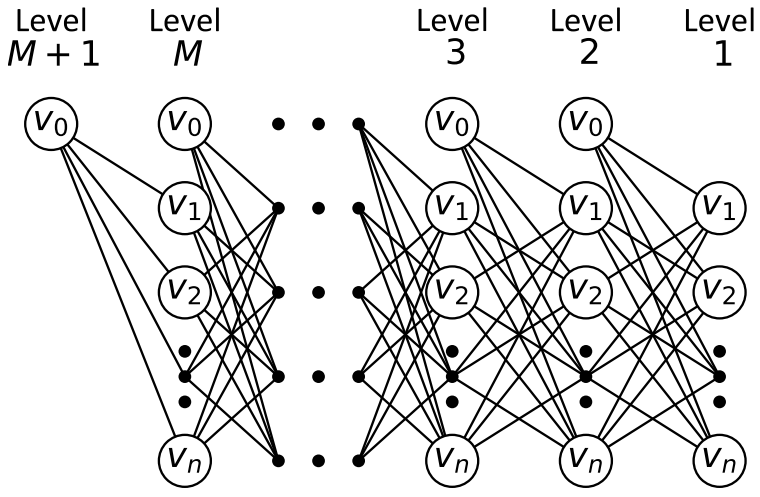

Figure A.9: Level based structure

$$\textbf{Min} \quad \sum_{j=1}^{n} c_{0j} \sum_{r=1}^{M} r x_{0j}^r + \sum_{i=1}^{n} \sum_{j=1 \neq i}^{n} c_{ij} \sum_{r=1}^{M-1} r x_{ij}^r \tag{A.1}$$

$$\sum_{j=1}^{n} x_{0j}^{1} + \sum_{i=1}^{n} \sum_{j=1 \neq i}^{n} x_{ij}^{1} = L \tag{A.2}$$

$$\sum_{r=1}^{M} \sum_{j=1}^{n} x_{0j}^{r} = L \tag{A.3}$$

$$x_{0j}^{r+1} + \sum_{i=1 \neq j}^{n} x_{ij}^{r+1} = \sum_{i=1 \neq j}^{n} x_{ji}^{r}, \quad j \in 1, ..., n, r \in 1, ..., M-2 \tag{A.4}$$

$$x_{0j}^{M} = \sum_{i=1 \neq j}^{n} x_{ji}^{M-1}, \quad j \in 1, ..., n \tag{A.5}$$

$$\sum_{r=1}^{M} x_{0j}^{r} + \sum_{r=1}^{M-1} \sum_{i=1 \neq j}^{n} x_{ij}^{r} = 1, \quad j \in 1, ..., n \tag{A.6}$$

$$x_{ij}^{r} \in \{0, 1\}, \quad i \in 0, 1, ..., n, j \in 1, ..., n, i \neq j, r \in 1, ..., M \tag{A.7}$$

(A.1) gives the level-based objective function for the MMLP. By constraint (A.2) we guarantee that exactly L RDCs finish their routes at level 1. Constraint (A.3) sets the number of RDCs that leave the depot node equal to L. Constraint sets (A.4) and (A.5) are the flow conservation constraints to ensure continuity of the paths. Constraints set (A.6) is to make certain that each node in $\{v_1, ..., v_n\}$ is active at a single level in any feasible solution. Finally, (A.7) is the variable description constraints.