An online optimization approach to post-disaster road restoration

Vahid Akbari[a,*], Davood Shiri[b], F. Sibel Salman[c]

[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*
[c]*College of Engineering, Koç University, Sariyer, Istanbul 34450, Turkey*

Abstract

Natural disasters impact transportation networks adversely and cause road sections to be damaged or blocked. The road network may even become disconnected, impeding accessibility between disaster-stricken areas and critical locations such as hospitals, relief aid depots and transportation hubs. In the immediate response phase, a set of blocked edges should be selected and restored to reconnect the transportation network. While locations of the disrupted roads can be identified using drone or satellite images, an accurate estimation of time to restore a road segment can be carried out only after expert observations on the field. In this article, we study a post-disaster road restoration problem modeled on an undirected edge-weighted graph with k blocked edges, where the unblocking time of a blocked edge is revealed online once the road restoration team visits an end-node of that blocked edge. The objective is to minimize the time at which the road network is reconnected. We first investigate the worst-case performance of online algorithms against offline optimal solutions by means of the competitive ratio. We prove that any online deterministic algorithm cannot achieve a competitive ratio better than $2k-1$. We also provide an optimal online algorithm that is proven to achieve this lower bound. In addition, to achieve good performance on realistic instances, we implement an algorithm that solves a mixed integer programming model each time new information is revealed. Since model solution is prohibitively time-consuming, we also propose a novel polynomial time online algorithm. We compare these two algorithms with two other benchmark online algorithms on both Istanbul road network instances and several other city instances from the literature. Our experiments show that the proposed polynomial time online algorithm performs superior to the benchmark ones and obtains solutions close to the offline optimum on all the tested instances.

Keywords: Disaster response logistics, online optimization, road clearance, network restoration, competitive ratio

*Corresponding author
Email addresses: vahid.akbari@nottingham.ac.uk (Vahid Akbari), davood.shiri@medipol.edu.tr (Davood Shiri), ssalman@ku.edu.tr (F. Sibel Salman)

1. Introduction

Disasters threat the well-being of people worldwide increasingly. According to the emergency events database (EM-DAT) (CRED 2013), past two decades have seen a significant increase in the number and impact of natural and man-made disasters worldwide. A recent report by CRED & UNDRR (2020) states that in the period 2000 to 2019, there were 7,348 major recorded disaster events claiming 1.23 million lives, affecting 4.2 billion people (many on more than one occasion) resulting in approximately 2.97 trillion dollars in global economic losses, indicating a sharp increase over the previous twenty years. While effective disaster preparedness, mitigation and response are the most important means to reduce the losses, a commonly experienced complication in the aftermath of a disaster is the blockage of the roads due to damage on road segments, bridges and viaducts, together with accumulation of debris (Shiri & Salman 2020).

These incidents cause the road networks to be disconnected and consecutively impede first response efforts including search-and-rescue operations and relief distribution. The urgent need to access casualties obligates timely debris clearance, road repair and restoration. Examples of post-disaster road damages and inaccessibility are abundant. For instance, in the 1999 Kocaeli Earthquake in Northwest Turkey, some highway sections were severely damaged and many roads in the urban areas were blocked by building debris and displaced cars (EEFIT 2003). After Japan's devastating 2011 earthquake and tsunami, the debris accumulated in the downtown of Kamaishi City caused the community to be isolated from search-and-rescue efforts (Tokuyama 2012). In the 2015 Nepal earthquake, some road segments near the epicenter of the earthquake became impassable, and a significant number of villages were isolated for days (Kasaei & Salman 2016). As a result of the 2018 Hurricane Michael in Florida, both road and power networks were damaged by broken and uprooted trees, twisted road signs laying on the ground and downed power lines (Zhai & Peng 2020). After the 2019 Golestan province flood in North of Iran, delivering aid to several urban areas and villages was not possible due to the disruptions in the transportation network (Abazari *et al.* 2020).

To facilitate emergency transportation in the immediate disaster response phase, blocked roads should be rapidly cleared or repaired; and if that is not possible, bypassed by alternative modes such as helicopters. Road clearing teams should be dispatched to the affected areas to restore network components and regain the connectivity of the road network in the shortest time. These teams are equipped with heavy machinery, bulldozers, lighting vehicles, drainage pump vehicles, satellite communication vehicles, etc., and consist of their operators and other required personnel (Kasaei & Salman 2016). Once the damaged road segments have been identified after examining satellite or drone images, the next step is to decide which roads to restore and in which order to achieve connectivity. This is to ensure that in the first critical hours and the following time frame, search-and-rescue (SAR) teams can reach the affected regions, casualties can be transported to emergency care units, victims can evacuate the region, and emergency relief supplies can be distributed. Several optimization approaches have been proposed to tackle these time-critical decisions in recent years.

Most of these approaches assume that it is known both which edges are blocked and how much time it would take to unblock/restore them (e.g., Akbari *et al.* (2021), Akbari & Salman (2017b), Kasaei & Salman (2016), Aksu & Ozdamar (2014), Yan & Shih (2009)). Alternatively, a few studies have assumed uncertain road restoration times (e.g., Caunhye *et al.* (2020), Sayarshad *et al.* (2020), Çelik *et al.* (2015)). Caunhye *et al.* (2020) propose a robust optimization approach, assuming a polyhedral uncertainty set for restoration times. Both Çelik *et al.* (2015) and Sayarshad *et al.* (2020) assume a probability distribution regarding the debris amount on each blocked edge, that in turn affects the amount of resources required to clear that edge. In a Partially Observable Markov Decision Process (POMDP) framework, information is updated regionally, when some edges become reachable and/or observable. That is, debris information obtained for roads in a certain region leads to updated belief probabilities for the clearance resource requirements in the same region. In Sayarshad *et al.* (2020), limited observable information about the disaster-relief resources located on the road network is also taken into consideration, along with a queuing model for clearance equipment requests and response times under uncertainty.

In general, the robust optimization approach has the disadvantage of making decisions under the most pessimistic scenario, whereas in approaches that assume a probability distribution for the uncertain edge restoration times, obtaining a probability distribution in the immediate response stage is difficult, as data acquisition is challenging in post-disaster environments (Ransikarbum & Mason 2016). However, since response time is crucial, relief operations may start before complete or probabilistic information about the blocked roads is available. Although locations of the damaged roads and the damage level can be identified from imagery, we believe that the road clearance/restoration times can only be estimated accurately when the situation is observed by experts at the field. Since deterministic approaches undermine the inherent uncertainties and stochastic approaches are not practical given the pressure of providing a fast response, we propose an online optimization approach as it can operate under incomplete information and construct a solution dynamically while obtaining and utilizing new information.

In this paper, we extend the deterministic problem studied in Kasaei & Salman (2016), that has the objective of reconnecting the network in shortest time, to an online variation, where the restoration times of the blocked edges are not known in advance. Instead, the time to restore a blocked edge is learned online when the road clearance team visits an end-node of that blocked edge for the first time. We propose an online optimization approach for minimizing the total operational time for the road clearing team to regain the connectivity of the disrupted network. In our solution, we select both the blocked edges that should be restored and the route of the road clearance team which covers all the selected blocked edges which in turn recover the connectivity of the network. The connectivity of the road network plays a key role in effective disaster management, in addition to preventing disruptions in vital services and industry.

The network connectivity objective has not been considered before in network restoration or debris clear-

ance studies that address uncertainty in different ways. Moreover, to the best of our knowledge, post-disaster network restoration or debris clearance problems have not been investigated from an online optimization point of view in the literature before. We introduce this new problem to the online optimization literature with the aim of optimizing post-disaster road clearance operations involving uncertainty. We call this problem the *Online Road Clearance for Connectivity Problem* and refer to it as ORCCP.

Online algorithms receive their input piece by piece and have to act upon the receipt of each piece of input. Their goal is usually to guarantee a performance which is as close as possible to the optimal performance achievable if the entire input is known in advance. How close they get to this optimal performance is usually analyzed by means of *competitive analysis* in the literature. In the competitive analysis approach, the performance of the solution obtained under incomplete information by the *online algorithm* is compared with the performance of the optimal solution obtained in presence of complete information, i.e., the *offline optimum*. This approach requires no probabilistic knowledge about the incomplete information and is therefore a worst-case measure of performance (Jaillet & Stafford 2001). This type of analysis was first suggested in Sleator & Tarjan (1985) and later called *competitive analysis* in Karlin *et al.* (1988). To evaluate the performance of an online algorithm, the notion of *competitive ratio* has been introduced in Sleator & Tarjan (1985).

A deterministic algorithm is one whose actions are completely determined by the given input (Komm 2016). In contrast, a randomized algorithm is allowed to make random choices to determine its actions. In this study, we focus on deterministic online algorithms. For a deterministic online algorithm applied to an online minimization problem, the competitive ratio is the maximum ratio of the cost of the online algorithm to the cost of the offline optimum over all instances of the problem. We first prove a lower bound on the competitive ratio of all deterministic online algorithms for the ORCCP and then prove that this lower bound is tight by introducing an optimal deterministic online algorithm called the *Backtrack* algorithm. This tight lower bound on the competitive ratio suggests that no deterministic online algorithm can output better solutions in the worst case compared to our optimal algorithm from the competitive analysis viewpoint.

Since competitive analysis is a worst-case measure of performance, the Backtrack algorithm is designed to be efficient on the worst-case instance so that the maximum ratio of the cost of the online algorithm to the cost of the offline optimum is minimized. Consequently, in some real-world instances, other online algorithms may achieve smaller competitive ratios albeit they are not theoretically optimal. Moreover, the Backtrack algorithm has exponential running time and cannot be applied to large problem instances arising from urban networks. To remedy these limitations, we propose a polynomial time heuristic online algorithm called the *Thresholding* algorithm, which is shown to run in very short times on realistic-sized instances. We also develop and implement an online algorithm that is based on solving a mixed integer programming (MIP) model each time new information is revealed. With extensive computational experiments, we show

that although this method can find solutions with low competitive ratios, as expected, it is prohibitively time-consuming as it has to solve optimization models repetitively. Nevertheless, it yields very good benchmark solutions. Moreover, we develop two fast benchmark online algorithms to underline the importance of utilizing the revealed information. We evaluate all of our proposed algorithms on instances from the literature and observe the good performance of the Thresholding algorithm compared to the offline optimum values and the other algorithms.

The remainder of the article is organized as follows. In the next section, we provide a review of the relevant existing research. In Section 3, we give the detailed description of the ORCCP and give a MIP model for the offline version of this problem. In Section 4, we derive a tight lower bound on the competitive ratio of deterministic online algorithms and propose an optimal deterministic algorithm for the ORCCP. In Section 5, we first describe the online algorithm that solves MIP models repetitively, and then introduce our deterministic online algorithm (the Thresholding algorithm) and develop two other benchmark deterministic online algorithms, one based on solving a single mathematical model and the other having greedy choices. In Section 6, we provide a computational study to investigate the performance of the Thresholding algorithm in comparison to the offline optimum and the three alternative algorithms. Finally, we give concluding remarks and discuss future research directions in Section 7.

2. Literature review

Since our study addresses a network restoration problem with edge blockages which is defined as an online routing problem, our primary focus in the literature review section will be on studies in two areas. Firstly, we review studies related to road clearance and restoration problems with different objectives and then discuss articles on online routing problems with edge blockages.

2.1. Network restoration problems

In network restoration problems, recovery crews make damaged edges available by repairing components in a disrupted network (Iloglu & Albert 2020). The objectives that are addressed in these studies include maximizing accessibility defined by its different measures, maximizing total reward, minimizing total penalty, or minimizing the maximum time necessary to restore a particular subset of the network, etc. (Baxter *et al.* 2020). Since the comprehensive review of the literature on post-disaster network restoration and recovery provided by Çelik (2016), the area has expanded with recent studies.

Many studies focus on a single team, i.e. a single work crew for the road restoration problem. In order to ensure accessibility, Kasaei & Salman (2016) studied two variants of a single crew problem. The first problem minimizes the time in which a single work crew reconnects a disconnected road network (called the *Arc Routing for Connectivity Problem (ARCP)*). The second problem maximizes the total prize obtained from connecting the components (isolated regions) within a time limit (called the *Prize Collection Arc Routing*

for Connectivity problem (PC-ARCP)). In the PC-ARCP, a prize is assigned to each component according to its priority, and the prize is received if the component is reconnected to the depot node. These two problems were solved using both variable neighborhood descent and variable neighborhood search algorithms and an exact MIP model was developed for benchmarking purposes. Vodák *et al.* (2018) developed a metaheuristic using Ant Colony Optimization that, similar to the ARCP, reconnects all the isolated components by dispatching a single team.

Duque *et al.* (2016) developed an exact Dynamic Programming algorithm, for small scale instances and an iterated greedy-randomized constructive procedure to solve large scale instances of the so called *Network Repair Crew Scheduling and Routing Problem*. This problem addresses the scheduling and routing of a single crew, while maximizing accessibility to the towns and villages that demand humanitarian relief. Moreno *et al.* (2019) introduced the *Crew Scheduling and Routing Problem* that targets minimizing the time during which affected areas remain inaccessible. It is assumed that blockage occurs on the nodes instead of the edges and a branch-and-Benders-cut algorithm is developed to tackle this problem. Moreno *et al.* (2020a) studied the same problem and developed two new metaheuristics for its solution. Using a new hybrid branch-and-Benders-cut approach, they found new optimal solutions and improved bounds for benchmark instances. Ajam *et al.* (2019) addressed another single team road clearance problem that minimizes the total latency of reaching the critical nodes that need to be visited. They proposed both a matheuristic and a metaheuristic, along with a lower bounding approach.

While the majority of the studies in this area consider a single work crew, several studies that address the problem with multiple teams have also appeared. Aksu & Ozdamar (2014) proposed a dynamic path based mathematical model to clear blockages with multiple work crews. Özdamar *et al.* (2014) aimed to reduce post-disaster travel times between all pairs of locations throughout the cleanup operation and to minimize the makespan of the operations by means of a rule-based constructive heuristic. Li *et al.* (2020) addressed the scheduling and routing of repair crews considering how to provide required materials as well and proposed both a nonlinear programming model and a two-stage heuristic approach. Moreno *et al.* (2020b) proposed a heterogeneous multi-crew scheduling and routing problem that minimizes the accessibility time defined as the time that the demand nodes remain unconnected from the source node. They developed alternative mathematical programming formulations and valid inequalities that are tested through a case study.

When multiple work crews are considered, in routing of restoration crews, allocation of the tasks to the crews should also be taken into account. Once a blocked road is opened/restored by a team, other teams can traverse the same road without the need for restoring it again. Then, the question of how to coordinate the teams arises. This aspect complicates the problem further. Only few studies have addressed the coordinated routing of repair crews. Akbari & Salman (2017b) developed a mathematical model for ARCP studied in Kasaei & Salman (2016), extending it with multiple teams. The authors developed a

matheuristic approach for larger instances, which is based on relaxing the exact mathematical model and repairing the obtained asynchronized solution into a synchronized feasible one, followed by a local search procedure for improvement. Akbari *et al.* (2021) developed a decomposition-based heuristic algorithm to solve larger instances of this problem. In another study, Akbari & Salman (2017a) formulated a mathematical model with an efficient heuristic (based on Lagrangian relaxation) for the PC-ARCP with multiple teams. All of these studied resolved the complication related to the coordination of the teams.

Morshedlou *et al.* (2018) proposed two mathematical models to solve a synchronized routing problem for planning and scheduling restorative efforts for damaged networks. In this problem, the number of crews designated for each disrupted component has been considered as a decision variable to increase the flexibility of the models and the models were tested on infrastructure network instances from Shelby County in Tennessee, USA. All the studies mentioned above neglect the uncertainties in repair times and assume that the repair times of the edges or the nodes are known in advance. Different from these studies, in our online optimization approach, we assume that the blockage times are not known in advance and are only realized after a close expert observation at the blocked edges.

While most of the studies addressing the post-disaster road restoration problem are of a deterministic nature, few studies assume that stochastic information on the clearing times of blocked edges is available. In order to provide connectivity between supply and demand points, Çelik *et al.* (2015) developed a multi-period *Stochastic Debris Clearance Problem* that captures the dynamic post-disaster status in which road clearing times are assumed uncertain and probability information on the debris amounts along the roads is updated as clearance activities proceed. The solution gives the sequence of the blocked roads that should be cleared in each period. The authors first developed a POMDP model and then suggested a search tree with heuristic pruning, which was tested on both random and real data. With the same goal of establishing connectivity between supply and demand points, Sayarshad *et al.* (2020) extended the dynamic debris clearance problem proposed in Çelik *et al.* (2015) by including the re-positioning decisions for clearance equipment items in anticipation of the resource requirements. While the connectivity between supply and demand nodes is addressed in both of these studies, different from our study, the connectivity of the entire road network is not of interest.

In our study, we bring novelty to the road restoration literature by handling the uncertain information on road unblocking/restoration times via online optimization. In addition to showing optimality in terms of worst-case analysis, by computational tests it is shown that our proposed online algorithm runs very fast and returns solutions close to the offline optimal (that is generated under complete a priori information). Furthermore, we have highlighted the significance of our proposed online algorithm by comparing its results with a model-based algorithm that runs a MIP every time the repair time of a blocked edge is observed.

2.2. *Routing problems with online blocked edges*

Within a competitive analysis framework, the performance of an online algorithm is assessed by comparing its results with the offline optimal solution using the competitive ratio measure. Online algorithms are distinguished according to whether they are deterministic or randomized. In a deterministic algorithm, actions of the decision maker are not based on probabilistic outcomes, whereas in a randomized algorithm the actions of the decision maker are taken with respect to some probability distribution (Albers 2003). The competitive ratio of a deterministic online algorithm on an online minimization problem is the maximum ratio of the objective function value of that online algorithm to the optimal objective value of the offline version over all the instances of that problem. For a randomized algorithm however, the expected competitive ratio is measured. The expected competitive ratio of a randomized online algorithm on an online minimization problem is equivalent to the maximum ratio of the expected objective function value of the online algorithm to the optimal objective value of the offline version over all the instances of that problem.

Routing problems in which road networks are modeled by undirected edge-weighted graphs in presence of a bounded number of online blocked edges have been investigated widely in recent years. Work on these problems are divided into two streams according to the two cases of 1) recoverable and 2) non-recoverable online blocked edges. If an online blocked edge is non-recoverable, the location of the blocked edge is unknown. If an online blocked edge is recoverable, the unblocking cost, or both location and unblocking cost of the blocked edge are unknown. Incomplete information associated to an online blocked edge is revealed whenever one of its end-nodes is visited by a traveling agent.

Shiri & Salman (2019b) studied a variant of the shortest path problem that has been analyzed in presence of online recoverable blocked edges whose locations are known but unblocking costs are not known. A tight lower bound on the competitive ratio of deterministic online algorithms together with an optimal deterministic online algorithm is presented for the single-agent version of the problem. Also, a lower bound on the competitive ratio of deterministic online algorithms is provided for the multi-agent version of the problem. Here we note that the term *agent* used in online optimization literature corresponds to the work crew/team in the road restoration literature. Liao & Huang (2015) investigated a single-agent variant of the shortest path problem with online recoverable blocked edges, but in their work locations of the blocked edges and their unblocking costs are not known. A tight lower bound on the competitive ratio of deterministic online algorithms and an optimal deterministic online algorithm are presented. Also, a lower bound on the expected competitive ratio of randomized online algorithms is proven. None of the aforementioned studies which involve recoverable blockages is defined in the context of post-disaster management. Therefore, these problems differ from the ORCCP, especially in terms of their objectives.

To the best of our knowledge, in other studies which investigate routing problems in presence of online blocked edges, the blocked edges are assumed to be non-recoverable. In these problems, the locations of

blocked edges are not given. Zhang *et al.* (2015) studied the Steiner TSP in which a traveling agent should visit a certain subset of the vertices where there are online non-recoverable blocked edges in the input graph. A tight lower bound on the competitive ratio of deterministic online algorithms together with an exponential time deterministic algorithm which meets the given lower bound are presented. Also, a polynomial-time asymptotically optimal deterministic online algorithm is provided and its performance is tested on sparse randomly generated networks. Zhang *et al.* (2016) studied a variant of the Steiner TSP with online advanced non-recoverable blocked edges, where the agent receives blockage messages at a certain distance to the respective blocked edges. A lower bound on the competitive ratio of deterministic online algorithms is proven. In addition, a polynomial-time deterministic online algorithm is proposed and tested on sparse randomly generated networks. The minimum latency problem with online non-recoverable blocked edges has been investigated in Zhang *et al.* (2019) and Akbari & Shiri (2021). While Zhang *et al.* (2019) proposed a lower bound on the competitive ratio of the deterministic online algorithms, Akbari & Shiri (2021) provided further insights by developing an optimal deterministic online algorithm which meets this lower bound.

Routing problems involving online non-recoverable blocked edges and multiple traveling agents have been also studied in the literature. Shiri & Salman (2017) and Shiri & Salman (2019a) investigated deterministic and randomized online algorithms for this problem from a theoretical competitive analysis perspective, respectively. This problem has also been investigated from a computational point of view in Shiri & Salman (2020), where an efficient heuristic deterministic online algorithm is proposed and tested on real city, as well as randomly generated road networks. Recently, Shiri *et al.* (2020) studied a post-disaster heterogeneous multi-team search-and-rescue problem modeled on an undirected graph involving non-recoverable online blocked edges. In this study, the search-and-rescue time for each critical node is also online. They proposed two lower bounds on the competitive ratio of deterministic online algorithms together with two efficient heuristic deterministic algorithms.

To the best of our knowledge, the road restoration problem with online blocked edges associated with unknown unblocking times has not been studied from competitive analysis and online optimization viewpoints in the literature. In this article, we define a road restoration problem on an undirected graph in presence of online blocked edges for the first time. We present a comprehensive theoretical competitive analysis and four online algorithms that are developed to solve this problem together with their extensive computational study.

3. Problem Description

Let G be an undirected connected graph, where V is the vertex set and E is the edge set. A positive cost/time (we interchangeably use cost and time) c_{ij} is associated with traversing each edge $e = (i, j) \in E$. Each edge (i, j) can be traversed in the direction from i to j and from j to i with the same cost c_{ij}. Here we

note that all the aforementioned problems which consider online blocked edges are defined on undirected road networks. This is a standard assumption since in an emergency situation roads can be used in both directions to facilitate the urgent traffic flow (Morshedlou *et al.* 2018).

After a natural disaster damages the road network, a subset of the edges, $B \subset E$ will be blocked. The intact edges together with the node set V form the graph $G_B = (V, E \setminus B)$. We assume that G_B is a disconnected subgraph including Q ($Q \geq 2$) connected components where C_q ($q = 1, 2, ..., Q$) is the q^{th} connected component with node set V_q and edge set E_q. A team equipped with needed machinery and consisting of their operators and other required personnel are initially positioned at a node called the depot. Furthermore, without loss of generality, we assume that the depot node denoted by v_s is in the first connected component ($v_s \in V_1$).

Traversing a blocked edge is not possible unless it is opened. The opening operation may involve clearing of the debris or repairing damaged segments. We refer to this task as *unblocking* throughout the paper. When a blocked edge is traversed for the first time, it is unblocked with an additional time b_{ij} and becomes open for later traversals with time c_{ij}, i.e., a total time of $b_{ij} + c_{ij}$ is required for unblocking a blocked edge together with its traversal.

Components C_m and C_n where $m, n \in \{1, ..., Q\}$ get connected if the team unblocks an edge $(i, j) \in B$ such that $i \in V_m$ and $j \in V_n$ or vice versa. We are looking for a subset of blocked edges, $R \subset B$, such that $G_R = (V, (E \setminus B) \cup R)$ is connected. The objective is to identify R and find a walk for the team such that it starts from the depot and covers edges in R to regain the connectivity of the network in shortest time. We not only consider the selection of blocked edges to be opened, R, but also construct the route for the team simultaneously.

In the offline version of the ORCCP, all the blocked edges, B, and their corresponding unblocking times, b_{ij} for $(i, j) \in B$ are known in advance. In this study, we consider the online version of ORCCP, where the blocked edges are known but the required unblocking time of a blocked edge $(i, j) \in B$, b_{ij}, is revealed upon a close assessment of the damage when the team is at one of the nodes adjacent to that edge. The notation used throughout the paper is given in Table 1.

We illustrate the ORCCP using a simple example presented in Figure 1. In Figure 1, part (a) shows the initial network having no damage, i.e., graph G. As it can be observed, the graph is connected since there is a path between each pair of nodes. Part (b) shows the disconnected network G_B having three components, C_1, C_2 and C_3. In the ORCCP, the opening times of the blocked edges are not known a priori and hence, information about the unblocking times should be collected during the course of opening the blocked edges. In Figure 1 part (c), according to an ORCCP solution, edges (10, 11) and (8, 11) are chosen to be unblocked; thus, they form the set R. In this figure, we can see the road clearing route that includes nodes 1, 3, 2, 5, 6, 10, 11 and 8 in order of traversal. Figure 1 part (d) shows the connected network consisting of a single

Notation	Description
$G = (V, E)$	The initial undirected graph with node set V and edge set E.
v_s	Depot node.
$n + 1$	Index of a dummy sink node.
DE	Set of all dummy edges between each node in $V \setminus v_s$ and the sink node.
$E' = E \cup DE$	The union of the edges in E and the dummy edges in DE.
B	Set of blocked edges.
$k = \|B\|$	Number of blocked edges.
R	Set of blocked edges selected to be opened.
$G_B = (V, E \setminus B)$	The graph formed by the intact edges and the node set V.
$G_R = (V, (E \setminus B) \cup R)$	The graph after road restoration operations.
Q	Number of components in G_B.
$C_q, q \in \{1, \ldots, Q\}$	The qth component.
$V_q, q \in \{1, \ldots, Q\}$	Node set of the qth component.
$E_q, q \in \{1, \ldots, Q\}$	Edge set of the qth component.
$c_{ij}, (i, j) \in E$	Traversal cost/time of edge $(i, j) \in E$.
$b_{ij}, (i, j) \in B$	Unblocking cost/time of blocked edge $(i, j) \in B$ (unknown in advance).

Table 1: Table of notation

component, i.e., graph G_R, which results after the selected opening tasks are completed.

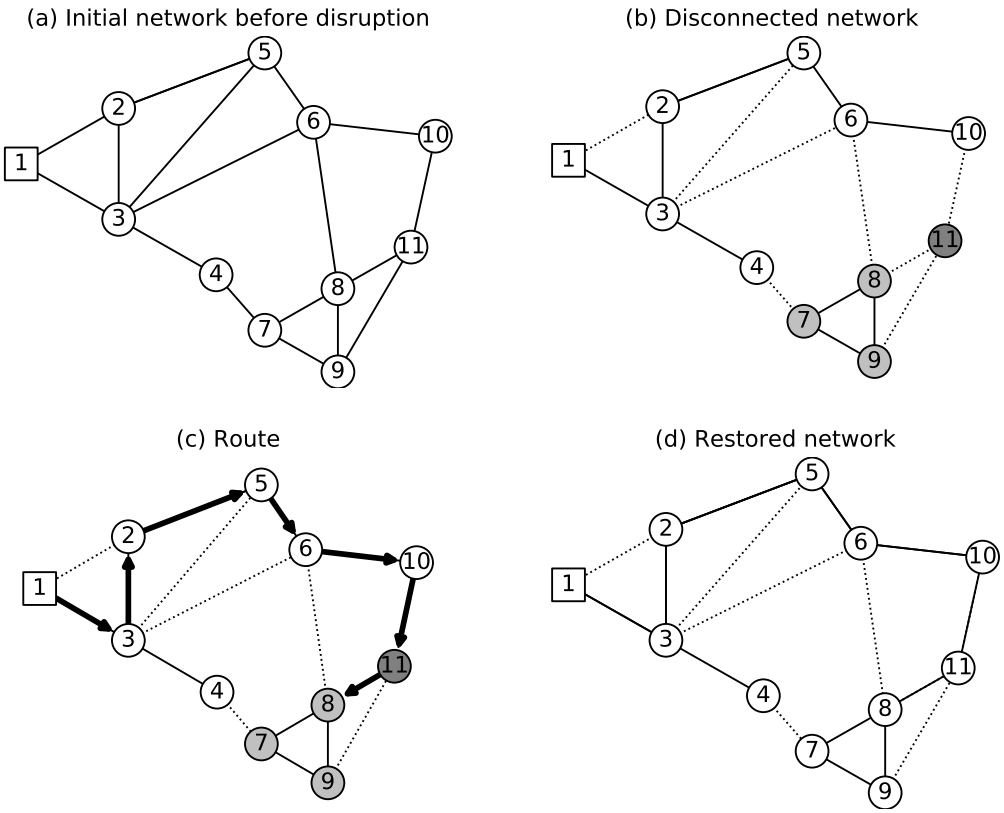


Figure 1: An example of the problem

3.1. The Offline Model

In this section we present an exact mathematical model for the offline ORCCP in which the blocked edges and their unblocking times are known in advance. We emphasize that this model's solution is used for benchmarking purposes in calculating the competitive ratio.

The offline problem was first introduced and studied in Kasaei & Salman (2016). In this problem, the objective is to minimize the length of the route for the team to traverse some of the intact edges as necessary and unblock some of the blocked roads as selected, in order to collectively form a connected graph. It is already shown in Kasaei & Salman (2016) that the offline problem is NP-hard. To formulate this problem, we define a dummy sink node indexed by $n+1$. The route of the clearing team starts from the depot and ends in the dummy sink node, thereby constructing an open route. We furthermore add a link from all the nodes except for the depot to the dummy sink node and set its cost equal to 0, i.e, $c_{i(n+1)} = 0, \forall i \in V \setminus v_s$. We also define a set $E' = E \cup DE$, where DE is the set of all dummy edges between each node in $V \setminus v_s$ and the dummy sink node. In the following, we first describe the decision variables and then present the model.

Decision Variables

$$x_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \in E' \text{ is traversed in the direction from } i \text{ to } j. \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$z_{ij} = \begin{cases} 1, & \text{if blocked edge } (i,j) \in B \text{ is unblocked.} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$f_{ij} \geq 0 : \text{flow amount on edge } (i,j) \in E' \text{ from node } i \text{ to } j \tag{3}$$

$$y_i \geq 0 : \text{number of times node } i \in V \text{ is visited} \tag{4}$$

MIP Model

$$Min \quad \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{(i,j)\in B} b_{ij}z_{ij} \tag{5}$$

$$S.t.$$

$$\sum_{j:(i,j)\in E'} x_{ij} - \sum_{l:(l,i)\in E} x_{li} = 0 \quad \forall i \in V \setminus v_s \tag{6}$$

$$\sum_{j:(v_s,j)\in E} x_{v_s j} - \sum_{l:(l,v_s)\in E} x_{l v_s} = 1 \tag{7}$$

$$\sum_{i\in V\setminus V_s} x_{i(n+1)} = 1 \tag{8}$$

$$\sum_{i\in V_q} y_i \geq 1 \quad \forall q \in Q \tag{9}$$

$$\sum_{j:(j,i)\in E} x_{ji} = y_i \quad \forall i \in V \tag{10}$$

$$x_{ij} + x_{ji} \geq z_{ij} + z_{ji} \quad \forall (i,j) \in B \tag{11}$$

$$x_{ij} \leq z_{ij} + z_{ji} \quad \forall (i,j) \in B \tag{12}$$

$$z_{ij} + z_{ji} \leq 1 \quad \forall (i,j) \in B \tag{13}$$

$$\sum_{l:(l,i)\in E} f_{li} - \sum_{j:(i,j)\in E'} f_{ij} = y_i \quad \forall i \in V \setminus v_s \tag{14}$$

$$\sum_{j:(v_s,j)\in E} f_{v_s j} - \sum_{l:(l,v_s)\in E} f_{l v_s} = \sum_{i\in V} y_i \tag{15}$$

$$\sum_{i\in V\setminus V_s} f_{i(n+1)} = 1 \tag{16}$$

$$f_{ij} \leq |V| x_{ij} \quad \forall (i,j) \in E' \tag{17}$$

$$x_{ij} \leq f_{ij} \quad \forall (i,j) \in E' \tag{18}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E' \tag{19}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in B \tag{20}$$

$$f_{ij} \geq 0 \quad \forall (i,j) \in E' \tag{21}$$

$$y_i \geq 0 \quad \forall i \in V \tag{22}$$

The objective function, given in (5), minimizes the summation of the total traveled distance and the total unblocking times. Constraints (6), (7) and (8) are to set the flow balance of the team in the intermediate nodes, the depot and the sink node, respectively. Constraint (9) is to ensure that at least one node in each component is visited and hence after opening the selected blocked edges, the graph is reconnected. Constraint (10) is to count the number of times a node is visited. Constraints (11), (12) and (13) are to make sure that blocked edges are not traversed unless they are opened and a blocked edge cannot be opened unless it is traversed. Moreover, opening a blocked edge in either direction makes it possible to traverse it on the opposite direction without a need for unblocking. Constraints (14) to (18) are to eliminate sub-tours using flow equations. Figure 2 adopted from Akbari & Salman (2017a) gives an example of how these constraints prevent sub-tours. In the given walk in this figure, total number of visits is 12. A flow of 12 leaves the depot node and at each visit to any node, one unit of flow is left. Finally one unit of flow enters the sink node. Constraints (14) and (15) make a work team leave one unit of flow whenever it visits a node. Constraints (16) ensure that walks end in the sink node. Constraint (17) does not allow flow on an edge if it is not traversed. Constraint (18) forces a positive amount of flow through an edge whenever that edge is traversed. Constraints from (19) to (22) are to set the variable restrictions. Constraint (19) sets binary x variables that denote the steps of the walk for the road clearance team. (20) is another binary variable defined on blocked edges and shows whether they are opened or not. (21) are continuous flow variables defined for sub-tour

elimination. The integrality of the flow variables is relaxed as they can only take integer values given constraints (14) and (15). (22) are continuous variables that show the number of times a node is visited. The integrality of these variables is also relaxed because of constraint (10).

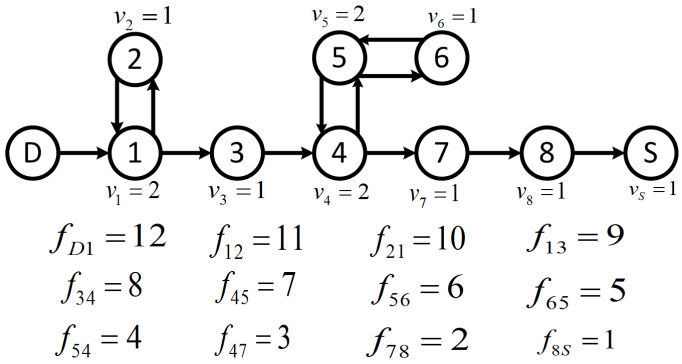


Figure 2: Demonstration of the flow formulation Akbari & Salman (2017a)

The solution to this offline model specifies the minimized walk for the road clearance team that satisfies the following: 1) This walk should contain a set of blocked edges such that opening them guarantees the connectivity of the network. This is achieved by constraint 9; 2) A blocked edge is not traversed unless it is opened, and opening a blocked edge is not possible unless it is traversed; 3) The walk starts in the depot and ends in the dummy sink node. All of these requirements are achieved by the given constraints and due to the objective function. The walk can be extracted using the x_{ij} variables whose values are equal to 1. In order to do this, we can use the walk extraction algorithm given in Akbari & Salman (2014).

4. Worst-case Competitive Analysis

In this section, we analyze the ORCCP by investigating the competitive ratio. First, we derive a lower bound on the competitive ratio of deterministic online algorithms for the ORCCP. Next, we prove the tightness of our proposed lower bound by introducing an optimal deterministic online algorithm. Competitive analysis is a standard approach found in the literature for theoretical analysis of online algorithms. In this approach, the performance of an online algorithm is analyzed by comparing its solutions with the best solution which can be obtained in presence of complete information (offline optimum). This comparison both specifies the value of having complete information at the moment the problem is solved and gives an indicator on the performance of the online algorithm.

A lower bound on the competitive ratio of the online algorithms for the ORCCP measures the worst-case performance ratio of online algorithms against the offline optimal solution. To prove a lower bound on the competitive ratio of deterministic online algorithms for an online minimization problem (e.g., the ORCCP), it is sufficient to consider an instance of that problem and show that the ratio of the cost of any deterministic online algorithm to the cost of the offline optimum is not less than the proposed lower bound on that instance.

(We refer the reader to Shiri *et al.* (2020), Zhang *et al.* (2019) and Zhang *et al.* (2015) for proofs driving lower bounds for related online optimization problems.) In Lemma 4.1, we prove a lower bound of $2k-1$ on the competitive ratio of deterministic online algorithms of the ORCCP, where k denotes the number of blocked edges in problem input.

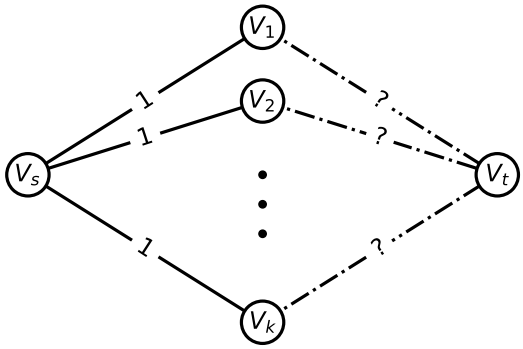


Figure 3: An instance used in the proof of Lemma 4.1

Lemma 4.1. *No deterministic online algorithm achieves a competitive ratio less than* $2k-1$ *for the ORCCP.*

Proof. We consider the instance shown in Figure 3, where nodes v_s and v_t are disconnected from each other. The road clearance team is initially positioned at v_s. We let the traversal costs of the edges $\{(v_s, v_i) \mid i = 1, 2, ..., k\}$ equal one and the traversal costs of the edges $\{(v_i, v_t) \mid i = 1, 2, ..., k\}$ equal to a sufficiently small positive value ϵ. There are k blocked edges in the graph, i.e., $\{(v_i, v_t) \mid i = 1, 2, ..., k\}$, whose unblocking costs are not known in advance.

In an arbitrary deterministic online algorithm referred to as *ALG*, the team traverses one of the edges (v_s, v_i) $(i \in \{1, 2, ..., k\})$, arrives at node v_i, and observes the unblocking time of edge (v_i, v_t). If the team unblocks (v_i, v_t), then *ALG* terminates. Otherwise, the team backtracks to v_s with the gained information (the unblocking time of the observed blocked edge). Therefore, *ALG* applied to the instance described above corresponds to a permutation which specifies in which order the unblocking times of the edges $\{(v_i, v_t) \mid i = 1, 2, ..., k\}$ are observed. For the worst-case analysis of *ALG*, we consider the instance in which the first $k-1$ observed unblocking times are M and the k^{th} observed unblocking time is ϵ, where M and ϵ are sufficiently large and small positive values, respectively.

Whenever the road clearance team learns that the unblocking time of an edge is M, it backtracks to v_s. Since in this worst-case analysis, the first $k-1$ observed blocked edges have an unblocking time of M, the road clearing team backtracks to the depot $k-1$ times. When the k^{th} blocked edge with an unblocking time of ϵ is observed, then the road clearing team unblocks it and the graph becomes connected. As a result, the cost of *ALG* is at least $2k-1+2\epsilon$. In order to find the competitive ratio, we note that the objective value of the offline optimum for this instance is $1+2\epsilon$. This is because in the offline problem, the information of the

blockage is known and the road clearing team directly goes to the blocked edge with an unblocking time of ϵ and opens it. The lemma follows when ϵ approaches zero. □

4.1. The Backtrack algorithm

Next, we introduce a deterministic online algorithm whose competitive ratio meets the proposed lower bound of $2k-1$ in Lemma 4.1. We call this algorithm the *Backtrack algorithm*. The Backtrack algorithm is an iterative algorithm with at most $k+1$ iterations. At each iteration of the Backtrack algorithm, the team travels on the graph to either connect the graph or observe the unblocking cost of a new blocked edge. If the graph becomes connected, the algorithm terminates. Otherwise, the team backtracks to v_s and the next iteration starts.

The backtrack algorithm has three main steps including; Initiation, Step 1 and Step 2. In the Initiation step of the backtrack algorithm, the sets that are used in the algorithm are defined and their initial values are assigned to them. In the first step of the backtrack algorithm, we utilize the MIP model in Section 3.1 and compute the cost of a walk for the team which connects graph G when the set of non-observed blocked edges are removed from G. If such a walk does not exist on G, we set its cost to a sufficiently large positive value. We also apply the MIP model in Section 3.1 to compute the cost of a walk for the team on a graph G^{ϵ} where the unblocking costs for non-observed blocked edges is set to a small positive value (ϵ). Based on the obtained values, we check if it is feasible to connect the graph such that the competitive ratio remains under the provided lower bound of $2k-1$. Then, we either terminate the algorithm by connecting G or move to Step 2. Finally, Step 2 of the backtrack algorithm is for observation of blocked edges and updating the set of observed blocked edges. After observing the unblocking cost of a new blocked edge, we again check if it is feasible to connect the graph such that the competitive ratio remains under the provided lower bound of $2k-1$. Next, we either terminate the algorithm by connecting the graph or move back to Step 1. In the following, we first give the pseudo-code of the Backtrack algorithm in Algorithm 1, and then give an example of this algorithm and finally prove its optimality for the ORCCP in Theorem 4.2.

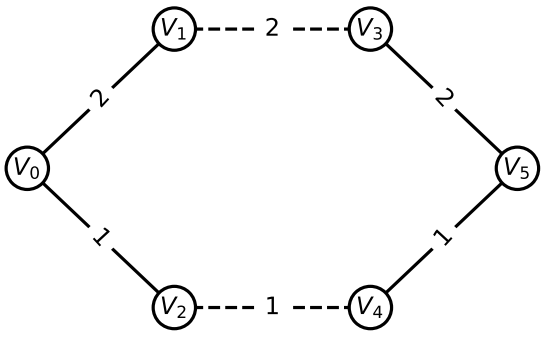


Figure 4: Example to illustrate the Backtrack algorithm

To clarify the description of the Backtrack algorithm, we explain its steps on the instance shown in

Backtrack algorithm

Input:

1: $G = (V, E)$, B, c_{ij}, v_s, k, C_q.

2: **Initiation:**

a: $\gamma = 1$ ▷ γ : iteration counter variable.

b: $\Delta_\gamma = 0$ ▷ Δ_γ : incurred cost from the start of the algorithm, until the beginning of the γ^{th} iteration.

c: $B_1 = B$ ▷ B_γ : set of non-observed blocked edges at the beginning of γ^{th} iteration.

3: **Step 1:**

a: $G_\gamma = (V, E \setminus B_\gamma)$

b: α_γ, $W_\gamma \leftarrow MIP(G_\gamma)$ ▷ α_γ : objective value of $MIP(G_\gamma)$ ▷ W_γ : extracted walk from $MIP(G_\gamma)$ ▷ $MIP(G)$: the MIP model from Section 3.1 on graph G.

c: **If** $MIP(G_\gamma)$ is infeasible **then:** $\alpha_\gamma = \infty$ **end if**

d: $G^\epsilon_\gamma = (V, E)$ ▷ G^ϵ_γ : the graph in which the unblocking cost of the edges in B_γ is set to ϵ.

e: α^ϵ_γ, $W^\epsilon_\gamma \leftarrow MIP(G^\epsilon_\gamma)$ ▷ α^ϵ_γ : objective value of $MIP(G^\epsilon_\gamma)$. ▷ W^ϵ_γ : extracted walk from $MIP(G^\epsilon_\gamma)$.

f: **If** $\dfrac{\Delta_\gamma + \alpha_\gamma}{\alpha^\epsilon_\gamma} \leq 2k - 1$ **then:** ▷ Stopping condition.

g: Take W_γ and complete the traversal of the walk. ▷ Possible Termination Point.

h: **Else:**

i: Go to **step 2**.

4: **Step 2:**

a: Take W^ϵ_γ to an end-node of $e_\gamma \in B_\gamma$. ▷ e_γ : the first non-observed blocked edge in W_γ.

b: b_{e_γ} : observe and record the unblocking cost of e_γ.

c: $B_{\gamma+1} = B_\gamma \setminus \{e_\gamma\}$

d: $\alpha^\epsilon_{\gamma+1}$, $W^\epsilon_{\gamma+1} \leftarrow MIP(G^\epsilon_{\gamma+1}) \leftarrow G^\epsilon_{\gamma+1}$

e: **If** e_γ is the only edge in W^ϵ_γ that belongs to B_γ and $\dfrac{\Delta_\gamma + b_{e_\gamma} + \alpha^\epsilon_\gamma}{\alpha^\epsilon_{\gamma+1}} \leq 2k - 1$ **then:** ▷ Stopping condition.

f: Unblock e_γ, and complete the traversal of W^ϵ_γ. ▷ Possible Termination Point

g: **Else:**

h: Backtrack to the depot (v_s).

i: $\gamma = \gamma + 1$

j: $\Delta_\gamma = \Delta_{\gamma-1}$+ cost of traversing to the $e_{\gamma-1}$ + backtracking cost from $e_{\gamma-1}$ to the depot.

k: Go to **Step 1**.

Figure 4, where edges (V_1, V_3) and (V_2, V_4) are blocked. In this instance, the graph is disconnected because of the blockage on (V_1, V_3) and (V_2, V_4). The number on each of the edges represents the traversal time. The unblocking times of (V_1, V_3) and (V_2, V_4) are assumed to be 100 and 20, respectively. However, the unblocking times are unknown to the restoration team and can be learned only if the team visits one of the end-nodes of the blocked edges. Note that the number of blocked edges is 2 (i.e., $k = 2$) and the restoration team is initially at V_0. At the beginning of the **first iteration** no cost is incurred, i.e., $\Delta_1 = 0$. The set of non-observed blocked edges $B_1 = \{(V_1, V_3), (V_2, V_4)\}$. There is no walk which connects the graph when (V_1, V_3) and (V_2, V_4) are removed from the graph, i.e., W_1 does not exists and $\alpha_1 = M$ such that M approaches $+\infty$. Also, when the unblocking costs of (V_1, V_3) and (V_2, V_4) are set to a sufficiently small value ϵ which approaches to 0, the walk traversing $V_0 \rightarrow V_2 \rightarrow V_4$ connects the graph with a minimum cost of $1 + 1 + \epsilon$. That is, $W^\epsilon_1 = V_0 \rightarrow V_2 \rightarrow V_4$ and $\alpha^\epsilon_1 = 2 + \epsilon$. According to Line 3:f of the pseudo-code, since

$\frac{\Delta_1 + \alpha_1}{\alpha_1^\epsilon} = \frac{0 + M}{2 + \epsilon} > 2k - 1 = 3$, the team does not traverse W_1 (note that W_1 does not exist and $\alpha_1 = M$). Instead, the road restoration team traverses W_1^ϵ to arrive at V_2 and observes the unblocking time of (V_2, V_4) which is 20.

At this point, since the unblocking time of a new blocked edge is learned (and at each iteration of the Backtrack algorithm at most one new blocked edge can be observed), B_2 is updated to $\{(V_1, V_3)\}$ and hence W_2^ϵ as well as α_2^ϵ can be calculated. Thus, W_2^ϵ is updated to $V_0 \rightarrow V_1 \rightarrow V_3$ with cost of $\alpha_2^\epsilon = 2+2+\epsilon = 4+\epsilon$. According to Line 4:e of the pseudo-code, since $\frac{\Delta_1 + \alpha_1^\epsilon + b_{(V_2,V_4)}}{\alpha_2^\epsilon} = \frac{0 + 2 + \epsilon + 20}{4 + \epsilon} \approx 5.5 > 2k - 1 = 3$, the team backtracks to V_0 and the second iteration starts. At the beginning of the **second iteration**, $\Delta_2 = 1 + 1 = 2$, $B_2 = \{(V_1, V_3)\}$, W_2 is $V_0 \rightarrow V_2 \rightarrow V_4$, $\alpha_2 = 1 + 1 + 20 = 22$, W_2^ϵ is $V_0 \rightarrow V_1 \rightarrow V_3$, and $\alpha_2^\epsilon = 2 + 2 + \epsilon = 4 + \epsilon$. According to Line 3:f of the pseudo-code, since $\frac{\Delta_2 + \alpha_2}{\alpha_2^\epsilon} = \frac{2 + 22}{4 + \epsilon} \approx 6 > 2k - 1 = 3$, the team does not take W_2. Instead, the team traverses W_2^ϵ to arrive at V_1 and observes the unblocking time of (V_1, V_3) which is 100 units. At this point, since the unblocking time of a new blocked edge is learned, B_3 is updated to $\emptyset$, W_3^ϵ is updated to $V_0 \rightarrow V_2 \rightarrow V_4$, and α_3^ϵ is updated to $1 + 1 + 20 = 22$. According to Line 4:e of the pseudo-code, since $\frac{\Delta_2 + \alpha_2^\epsilon + b_{(V_1,V_3)}}{\alpha_3^\epsilon} = \frac{2 + 4 + \epsilon + 100}{22} > 4.8 > 2k - 1 = 3$, the team backtracks to V_0 and the third iteration starts. At the beginning of the **third iteration**, $\Delta_3 = 1 + 1 + 2 + 2 = 6$, $B_3 = \emptyset$, $W_3 = W_3^\epsilon = V_0 \rightarrow V_2 \rightarrow V_4$, and $\alpha_3 = \alpha_3^\epsilon = 1 + 1 + 20 = 22$. According to Line 3:f of the pseudo-code, since $\frac{\Delta_3 + \alpha_3}{\alpha_3^\epsilon} = \frac{6 + 22}{22} \approx 1.27 < 2k - 1 = 3$, the team takes $W_3 = W_3^\epsilon = V_0 \rightarrow V_2 \rightarrow V_4$ and connects the graph. In this example, the cost of the Backtrack algorithm is $1 + 1 + 2 + 2 + 1 + 1 + 20 = 28$ and the cost of the offline optimum is $\alpha_{off} = \alpha_3 = \alpha_3^\epsilon = 1 + 1 + 20 = 22$. Thus, the competitive ratio is $\frac{28}{22} \approx 1.27$, which is less than the lower bound of $2k - 1 = 3$.

Since the Backtrack algorithm requires solving the offline MIP model, its worst-case complexity is exponential. The Backtrack algorithm ends in at most $k + 1$ iterations. We next prove our proposed lower bound of $2k - 1$ (Lemma 4.1) is *tight* for the ORCCP by showing that the competitive ratio of the Backtrack algorithm matches the lower bound of $2k-1$. In this way, we prove the optimality of the Backtrack algorithm for the ORCCP from the worst-case competitive analysis perspective.

Theorem 4.2. *The Backtrack algorithm meets the competitive ratio of* $2k - 1$ *for the ORCCP.*

Proof. We use the definitions and notation which are presented in the pseudo-code of the Backtrack algorithm in our proof. Let W_{off} be the offline optimum walk and α_{off} be its cost. Note that $\alpha_{k+1}^\epsilon = \alpha_{off}$, it is because all the unblocking costs are known when the algorithm enters the $(k + 1)^{th}$ iteration. Also α_γ^ϵ ($\gamma = 1, 2, ..., k + 1$) is non-decreasing in γ as more information reveals. Hence, $\alpha_\gamma^\epsilon \leq \alpha_{off}$ for $\gamma = 1, 2, ..., k + 1$. Note that if the Backtrack algorithm ends before the $(k+1)^{th}$ iteration, the competitive ratio is less than $2k-1$ according to the stopping conditions in Steps 1 and 2 of the algorithm. Thus, we only analyze the case when the algorithm enters the $(k + 1)^{th}$ iteration. Recall that Δ_γ ($\gamma \in \{1, 2, ..., k + 1\}$) denotes the total cost incurred

by the team from the start until the beginning of iteration γ. Note that

$$\Delta_\gamma \leq \sum_{\lambda=1}^{\gamma-1} 2\alpha_\lambda^\epsilon \leq (2\gamma-2)\alpha_{\gamma-1}^\epsilon \leq (2\gamma-2)\alpha_\gamma^\epsilon,$$

since α_γ^ϵ is non-decreasing in γ for $\gamma = 1, 2, ..., k+1$. If the algorithm ends at iteration $k+1$, the total cost is $\Delta_{k+1} + \alpha_{k+1}^\epsilon \leq \sum_{\lambda=1}^{k} 2\alpha_\lambda^\epsilon + \alpha_{off}$. Hence, the competitive ratio is upper bounded by

$$\frac{\sum_{\lambda=1}^{k} 2\alpha_\lambda^\epsilon}{\alpha_{off}} + 1.$$

The algorithm enters the $(k+1)^{th}$ iteration if W_{off} does not contain e_k, i.e., if $\alpha_{off} = \alpha_{k+1}^\epsilon < b_{e_k} + \alpha_k^\epsilon$. This is because if W_{off} contains e_k (i.e., if $\alpha_{k+1}^\epsilon = \alpha_{off} \geq b_{e_k} + \alpha_k^\epsilon$), the team takes W_k^ϵ, unblocks e_k since e_k is the only blocked edge in $B_k = \{e_k\}$ and

$$\frac{\Delta_k + b_{e_k} + \alpha_k^\epsilon}{\alpha_{k+1}^\epsilon} \leq \frac{(2k-2)\alpha_k^\epsilon + \alpha_{k+1}^\epsilon}{\alpha_{k+1}^\epsilon} \leq 2k-1,$$

and the algorithm ends at the k^{th} iteration according to the stopping condition at Step 2 (Line 4:e of the pseudo-code). Let $e_{\gamma^*} \neq e_k$ ($\gamma^* \in \{1, 2, ..., k-1\}$) be the last blocked edge in W_{off}, i.e., there is no blocked edge after e_{γ^*} in W_{off}. The algorithm enters the $(\gamma^*+1)^{th}$ iteration if $\frac{\Delta_{\gamma^*} + b_{e_{\gamma^*}} + \alpha_{\gamma^*}^\epsilon}{\alpha_{\gamma^*+1}^\epsilon} > 2k-1$ according to Step 2 (Line 4:e of the pseudo-code). Therefore,

$$b_{e_{\gamma^*}} > 2\alpha_{\gamma^*+1}^\epsilon$$

since $\Delta_{\gamma^*} \leq (2\gamma^*-2)\alpha_{\gamma^*}^\epsilon$ and $\gamma^* \leq k-1$. Observe that $\alpha_{off} \geq b_{e_{\gamma^*}}$, since we assumed that e_{γ^*} ($\gamma^* \in \{1, 2, ..., k-1\}$) is the last blocked edge on W_{off}, i.e., W_{off} contains e_{γ^*}. We re-write the upper bound of the competitive ratio as

$$\frac{\sum_{\lambda=1}^{k} 2\alpha_\lambda^\epsilon}{\alpha_{off}} + 1 = \frac{\sum_{\lambda=1}^{\gamma^*+1} 2\alpha_\lambda^\epsilon}{\alpha_{off}} + \frac{\sum_{\lambda=\gamma^*+2}^{k} 2\alpha_\lambda^\epsilon}{\alpha_{off}} + 1 \leq \frac{\sum_{\lambda=1}^{\gamma^*+1} 2\alpha_\lambda^\epsilon}{b_{e_{\gamma^*}}} + \frac{\sum_{\lambda=\gamma^*+2}^{k} 2\alpha_\lambda^\epsilon}{\alpha_{off}} + 1,$$

which is less than or equal to

$$\frac{\sum_{\lambda=1}^{\gamma^*+1} 2\alpha_\lambda^\epsilon}{b_{e_{\gamma^*}}} + 2(k-\gamma^*-1) + 1,$$

since we proved that $\alpha_\gamma^\epsilon \leq \alpha_{off}$ for $\gamma = 1, 2, ..., k+1$. To draw a larger competitive ratio we replace $b_{e_{\gamma^*}}$ with $2_{\alpha_{\gamma^*+1}^\epsilon}$ to obtain

$$\frac{\sum_{\lambda=1}^{\gamma^*+1} 2\alpha_\lambda^\epsilon}{2_{\alpha_{\gamma^*+1}^\epsilon}} + 2(k-\gamma^*-1) + 1,$$

which is less than or equal to $\gamma^* + 1 + 2(k-\gamma^*-1) + 1 \leq 2k-\gamma^*$, since α_γ^ϵ is non-decreasing in γ. Also

since γ^* is at least one, it follows that $2k - \gamma^* \leq 2k - 1$. □

5. Online algorithms

In this section we first propose a modified and improved version of the Backtrack algorithm that relies on solving a MIP model every time new information is revealed to the road clearance team. We refer to this algorithm as the Observe and re-Optimize (OREO) algorithm. Since the OREO algorithm is very time consuming as it has to solve the optimization model repetitively, we give an online algorithm to solve the ORCCP that is not based on solving an optimization model. We call this algorithm the Thresholding algorithm. We then give two benchmark algorithms for comparison. We demonstrate the four algorithms on a small-sized network in Section 5.4 for clarification. To describe the algorithms formally, we need the following definitions.

- We call blocked edges which connect at least two components *effective* blocked edges.
- We call effective blocked edges whose unblocking costs are not observed *effective unobserved* blocked edges and denote the set of such edges by B^e.
- We call blocked edges whose unblocking costs are observed *effective observed* blocked edges and denote the set of such edges by B^o.
- We call edges in B^e such that at least one of their end-nodes is reachable from the current location (after removing the blocked edges from the graph) *accessible* blocked edges.
- We denote the set of *accessible end-nodes* of blocked edges by $\mathcal{A}$.
- We define a *cost counter* variable to calculate the incurred cost from the beginning of the iteration until the current moment and denote it by Δ_γ.
- Finally, we define a *cost threshold* variable and denote it by T^*. This cost threshold value is only used in the Thresholding algorithm. If the cost threshold (T^*) is triggered in an iteration, the team considers unblocking an edge in B^o immediately.

5.1. OREO algorithm

In this section, we introduce an improved version of the Backtrack algorithm that is based on re-optimizing the route of the road clearance team using the MIP model presented in Section 3.1 every time new information is revealed. We note that the Backtrack algorithm was designed for the worst-case analysis and to show that it meets the lower bound of $2k - 1$. We have made some modifications to the Backtrack algorithm to improve its performance (on real-world instances) and designed the OREO algorithm given in

Algorithm 2. One of these modifications is that the road clearance team does not return to the depot when new information is revealed and instead, an updated MIP model with the current location of the team and the newly observed unblocking time is solved to decide the next movements of the team. Another modification is that at each node on the path of the team, the team can observe all the edges that are adjacent to that node and collect their blockage information, if there are any blocked edges that are adjacent to them. However, re-optimization is required only if a non-observed blocked edge is on the path of the team and the algorithm terminates if an extracted path for the road clearance team contains no blocked edges whose blockage information have already been observed.

In the initiation step of this algorithm, we set the set of observed blocked edges as an empty set and define a parameter denoted by $\mathcal{L}$ to keep track of the position of the team. The main procedure of the OREO algorithm contains two nested loops. Within the first loop, a MIP model is solved. The basis of this model is the one given in Section 3.1. There are three differences that should be taken into account. First of all, the network is dynamically updated when blocked edges are opened. Secondly, the current positioning of the team should be considered as its depot at the beginning of each iteration. Lastly, the unblocking times of the blocked edges that are not observed and are opened are set to zero for lack of information, and otherwise they are set to their unblocking times respectively.

Within the second loop, once a walk is obtained by the model solution, the edges of the walk are analyzed one by one. If the corresponding edge is not a blocked one, then it is traversed by the team. If it is a blocked one such that its unblocking time has been observed before, again, it will be traversed. Finally, if the corresponding edge on the walk is a blocked edge such that its unblocking time has not been observed, its unblocking time is observed and the edge is added to the set of observed blocked edges. This procedure ends when the team connects the graph.

5.2. Thresholding Algorithm

In this section, since the OREO algorithm is very time-consuming to be solved and requires a commercial optimization solver, we introduce a polynomial time deterministic online algorithm instead. We call this iterative algorithm that proceeds according to a threshold parameter, the *Thresholding* Algorithm. In each iteration of the Thresholding algorithm, the revealed information is evaluated by the algorithm and the iteration ends when a threshold triggers an action. To clarify the notions of Δ_γ and T^* in the Thresholding algorithm, we describe a generic iteration of the this algorithm in the following.

An iteration of the Thresholding algorithm: At each iteration γ of the Thresholding algorithm, the team unblocks one effective blocked edge and connects at least two components as follows. At the beginning of each iteration, the team sets $T^* = \infty$, and $\Delta_\gamma = 0$. Next, the team chooses a blocked edge $e \in B^e$ based on a *utility function*. The utility function considers: 1) the nearness of the blocked edge to the current location and 2) the nearness of the blocked edge to the other accessible blocked edges (see Step 2 in the pseudo-

Algorithm 2: OREO Algorithm

Input:

1: $G = (V, E)$, B, c_{ij}, v_s, $n + 1$

2: **Initiation:**

a: $\gamma = 1$ ▷ γ : a counter variable.

b: $\mathcal{L} = v_s$ ▷ $\mathcal{L}$: location of the team

c: $B^o = \emptyset$ ▷ B^o : set of effective observed blocked edges

d: **For** $(i, j) \in B$: **If** $i = \mathcal{L} : B^o = B^o \cup (i, j)$

e: $R = \emptyset$ ▷ R : set of opened blocked edges

f: $ObjVal = 0$ ▷ $ObjVal$: corresponding objective value

g: Flag1 = True ▷ Flag1: a flag parameter to control the loop.

3: **Main Procedure:**

a: **While** Flag1:

b: $\Delta_\gamma = 0$ ▷ Δ_γ : cost of each iteration

c: **For** $(i, j) \in B \setminus B^o : b_{ij}^\gamma = 0$

d: **For** $(i, j) \in B^o$:

e: **If** $(i, j) \in R : b_{ij}^\gamma = 0$

f: **Else**: $b_{ij}^\gamma = b_{ij}$

g: $G_R = (V, (E \setminus B) \cup R)$ ▷ G_R : restored graph

h: $W_\gamma = MIP(G_R, \mathcal{L}, b_{ij}^\gamma)$ ▷ W_γ : obtained walk in γ^{th} iteration.

▷ $MIP(G_R, \mathcal{L}, b_{ij}^\gamma)$: model on G_R considering the location of the team and updated blockage times.

i: Flag2 = True ▷ Flag2: a flag parameter to control the loop.

j: **While** Flag2:

k: Extract $(\mathcal{L}, v)$ which is the first edge on W_γ.

l: **If** $(\mathcal{L}, v)$ not in B: $\mathcal{L} = v$ and $\Delta_\gamma = \Delta_\gamma + c_{\mathcal{L}v}$

m: **If** $v = n + 1$: Flag1 = Falg2 = False ▷ Possible Termination Point.

n: **Else if** $(\mathcal{L}, v)$ in B^o:

o: **if** $(\mathcal{L}, v)$ in R: $\mathcal{L} = v$ and $\Delta_\gamma = \Delta_\gamma + c_{\mathcal{L}v}$

p: **Else:** $R = R \cup (\mathcal{L}, v)$ and $\mathcal{L} = v$ and $\Delta_\gamma = \Delta_\gamma + c_{\mathcal{L}v} + b_{\mathcal{L}v}$

q: **Else:** $B^o = B^o \cup (\mathcal{L}, v)$, Flag2 = False

r: **For** $(i, j) \in B \setminus B^o$: **If** $i = \mathcal{L} : B^o = B^o \cup (i, j)$

s: Remove $(\mathcal{L}, v)$ from the beginning of W_γ

t: **EndWhile**

u: $ObjVal = ObjVal + \Delta_\gamma$

v: $\gamma = \gamma + 1$

w: **EndWhile**

code). Then, the team takes the cheapest path from its current location to e, observes its unblocking cost, and updates Δ_γ. Note that the team can connect at least two components by incurring the summation of the unblocking cost of e plus the traversal cost of e. We use twice the summation of the unblocking cost of e plus the traversal cost of e as an updated value for the cost threshold T^*. Next, the team chooses another blocked edge $e^{'} \in B^e$ based on the utility function. If it is feasible to observe the unblocking cost of $e^{'}$ and connect at least two components within T^*, the team takes the cheapest path from the current location to $e^{'}$ and observes its unblocking cost. At this point T^* is updated as follows. If twice the summation of the unblocking cost plus the traversal cost of $e^{'}$ is less than T^*, the value of T^* is updated to the newly observed lower value. If it is not feasible to observe the unblocking cost of $e^{'}$ and connect at least two components within T^*, then the

team unblocks one of the edges in B^o (i.e., an observed blocked edge) which can be reached and unblocked with minimum cost. If the graph is connected, then the algorithm terminates. Otherwise, a new iteration begins.

The Thresholding algorithm consists of an initiation step and three steps. We give the descriptions of each step in the following pseudo-codes.

Initiation

Input:

1: $G = (V, E)$, B, c_{ij}, v_s, C_q.

2: **Initiation:**

a: $B^e = B$ ▷ B^e : set of effective unobserved blocked edges

b: $\mathcal{A} = \emptyset$ ▷ $\mathcal{A}$: set of accessible end-nodes of B^e from current location

c: $T^* = \infty$ ▷ T^* : cost threshold

d: $B^O : \emptyset$ ▷ B^O : set of effective observed blocked edges

e: $\Delta_\gamma = 0$ ▷ Δ_γ : cost of iteration γ

f: $\mathcal{L} = v_s$ ▷ $\mathcal{L}$: location of the team

g: $R = \emptyset$ ▷ R : set of opened blocked edges

h: $G_R = (V, (E \setminus B) \cup R)$ ▷ G_R : restored graph

i: $ObjVal = 0$ ▷ $ObjVal$: corresponding objective value

Description of the Initiation Step. Take an undirected graph $G = (V, E)$ together with edge traversal costs (c_{ij}), depot node $v_s \in V$, set of blocked edges $B \subset E$, and disconnected components C_q $(q \in \{1, 2, ..., Q\})$ as input. Let B^e be the set of effective unobserved blocked edges and set $B^e = B$. Define $\mathcal{A}$ as the set of accessible end-nodes of the blocked edges and set $\mathcal{A} = \emptyset$. Let T^* be a cost threshold and set $T^* = \infty$. Let B^o be the set of effective observed blocked edges and set $B^o = \emptyset$. Define Δ_γ as a cost counter variable and set $\Delta_\gamma = 0$. Similar to the OREO algorithm, $\mathcal{L}$ denotes the current location of the team which is set as $\mathcal{L} = v_s$ in the beginning. Let R be the set of opened blocked edges and set $R = \emptyset$. Define $G_R = (V, (E \setminus B) \cup R)$ as the restored graph. Let $ObjVal$ denote the total cost of the algorithm and set $ObjVal = 0$.

Description of Step 1. Let $B^{'}$ be the set of blocked edges which do not connect any two components. Set $B^e = B^e - B^{'}$. Add the end-nodes of the edges in B^e (which are accessible from $\mathcal{L}$) to $\mathcal{A}$. If $\mathcal{A} = \emptyset$, go to Step 3. Otherwise, for $v \in \mathcal{A}$, let t_v be the cost of the cheapest path from $\mathcal{L}$ to v. Also, let $t_v^{'}$ be the average of the costs of the cheapest paths from v to the other nodes in $\mathcal{A}$. Calculate the utility function value $z_v = \frac{1}{2}t_v + \frac{1}{2}t_v^{'}$ for $v \in \mathcal{A}$. Let $v^* \in \mathcal{A}$ be the node whose z_{v^*} is minimum. If $\Delta_\gamma + t_{v^*} < \frac{T^*}{4}$, go to Step 2. Otherwise, go to Step 3. By computing $z_v = \frac{1}{2}t_v + \frac{1}{2}t_v^{'}, v \in \mathcal{A}$, we aim to minimize the distance between the current location and the next node to be visited in set $\mathcal{A}$ while also considering the average distance from the selected node to the remaining nodes in $\mathcal{A}$. This is to ensure that while selecting the next blocked edge for observation, its proximity for other blockage observations or potential unblocking is also considered. In the case of $\Delta_\gamma + t_{v^*} \leq \frac{T^*}{4}$, we set a coefficient of $\frac{1}{4}$ for T* to ensure that a balance between "collecting more unblocking information" and "minimizing the overall time" is achieved. These coefficients are considered

Step 1

$B' = \{(i,j) \in B \text{ if } i,j \in V_q \quad \forall q \in \{1,...,Q\}\}$ ▷ B' : set of non-effective blocked edges
$B^e = B^e \setminus B'$
$\mathcal{A} = \{i : \ \forall (i,j) \in B^e \text{ if node } i \text{ is accessible from } \mathcal{L}\}$
if $\mathcal{A} = \emptyset$
  Go to **Step 3**
Else:
  t_v = shortest path distance from $\mathcal{L}$ to v in G_R, $v \in \mathcal{A}$
  $\mathcal{S}_{ij}$ = Shortest path distance from node i to j in G_R : $\forall i,j \in \mathcal{A}$
  $t'(v) = \dfrac{\sum_{w \in \mathcal{A}, w \neq v} \mathcal{S}_{vw}}{|\mathcal{A}| - 1}, \quad v \in \mathcal{A}$
  $z_v = \frac{1}{2} t_v + \frac{1}{2} t'_v, v \in \mathcal{A}$ ▷ z_v : utility function for node v
  $v^* = \mathcal{V} : z_{\mathcal{V}} < z_w, w \in \mathcal{A}$
  if $\Delta_\gamma + t_{v^*} \leq \frac{T^*}{4}$:
    Go to **Step 2.**
  else:
    Go to **Step 3**

with a number of parameter tuning computational experiments.

Step 2

$ObjVal = ObjVal + \Delta_\gamma$
$\mathcal{L} = v^*$
$\Delta_\gamma = \Delta_\gamma + t_{v^*}$
$S_{v^*} = \{(v^*, i) : (v^*, i) \in B^e\}$
$C_{min} = \min\{c_{ij} + b_{ij} \ \ \forall (i,j) \in s_{v^*}\}$
$B^e = B^e \setminus S_{v^*}$
$\mathcal{A} = \emptyset$
$B^O = B^O \cup S_{v^*}$
$T^* = \min\{2 \times C_{min}, T^*\}$
Go to **Step 1**

Description of Step 2. Update $ObjVal$. Take the cheapest path from $\mathcal{L}$ to v^*. Set $\Delta_\gamma = \Delta_\gamma + t_{v^*}$. Let S_{v^*} be the set of the edges in B^e emanating from v^*. Observe the unblocking costs of the edges in S_{v^*}. Let c_{min} be the minimum of the traversal cost plus the unblocking cost of the edges in S_{v^*}. Set $B^e = B^e - S_{v^*}$, $\mathcal{A} = \emptyset$, $B^o = B^o \cup S_{v^*}$, and $T^* = min\{2 \times c_{min}, T^*\}$. Then, go to Step 1. We compute $T^* = \min\{2 \times c_{min}, T^*\}$ to update the thresholding parameter (T*) based on the observed information to ensure that there is a balance between selecting the next blocked edge to be opened and exploring more blockage information.

Description of Step 3. For each edge $e \in B^o$, calculate q_e as the summation of the traversal cost of e plus the unblocking cost of e plus the cost of the cheapest path from $\mathcal{L}$ to e. Let e^* be the edge in B^o whose q_{e^*} is minimum. Take the cheapest path from $\mathcal{L}$ to e^* and unblock it to arrive at the other end-node. Set $ObjVal = ObjVal + q_{e^*}$, $R = R \cup e^*$, $G_R = (V, (E \setminus B) \cup R)$. If G_R is connected, stop and return $ObjVal$. Otherwise, set $B^o = B^o \setminus B^{'}$, $\Delta_\gamma = 0$, $\mathcal{A} = \emptyset$, and $T^* = \infty$. Then, go to Step 1.

All pairs shortest paths can be calculated at Steps 1 and 3 in $O(|V|^3)$ applying the Floyd-Warshall algo-

Step 3

1: **if** $\mathcal{A} = \emptyset$:
$\mathcal{A}' = \{i : \ \forall (i,j) \in B^o$: if node i is accessible from $\mathcal{L}\} \cup \mathcal{L}$
$\hat{t}_v$ = shortest path distance from $\mathcal{L}$ to v in G_R, $v \in \mathcal{A}'$
$q_{vw} = c_{vw} + b_{vw} + \hat{t}_v, \ \ \forall (v,w) \in B^O : v \in \mathcal{A}'$
$(v^*, w^*) = argmin\{q_{vw} \ \ \forall (v,w) \in B^O : v \in \mathcal{A}'\}$
2: **Else:**
$q_{vw} = c_{vw} + b_{vw} + t_v, \ \ \forall (v,w) \in B^O : v \in \mathcal{A}$
$(v^*, w^*) = argmin\{q_{vw} \ \ \forall (v,w) \in B^O : v \in \mathcal{A}\}$
3: $ObjVal = ObjVal + q_{v^*w^*}$
4: $\mathcal{L} = w^*$
5: $R = R \cup \{(v^*, w^*)\}$
6: Update $G_R = (V, (E \setminus B) \cup R)$
7: **If** G_R is connected.
Terminate **return:** $ObjVal$ ▷ Possible termination point
8: **Else:**
$B^O = B^O \setminus B'$
$\Delta_\gamma = 0$
$\mathcal{A} = \emptyset$
$T^* = \infty$
Go to **Step 1**.

rithm (Chan 2007). Since the team unblocks one blocked edge after visiting all the accessible nodes at the worst-case, the Thresholding algorithm enters Step 1 at most $k * |V|$ times. Hence, the worst-case complexity of Step 1 is $O(k * |V|^4) < O(|E| * |V|^4)$. Also, at most $S_{v^*} < |E|$ comparisons are made each time the algorithm enters Step 2. Since the algorithm enters Step 2 at most $k * |V| < |E| * |V|$ times, the worst-case complexity of Step 2 is $O(|E|^2 * |V|)$. The Thresholding algorithm enters Step 3 at most $k < |E|$ times; therefore, the worst-case complexity of Step 3 is $O(|E| * |V|^3)$. Hence, the Thresholding algorithm is polynomial time.

5.3. Benchmark Algorithms

In this section, we present two deterministic algorithms for the purpose of benchmarking the OREO and the Thresholding algorithms. The benchmark algorithms are based on simpler observations and in both of them the revealed blockage information is neglected on purpose. These algorithms are not adopted from the literature and we have developed, implemented and tested them to show the advantage of having an online algorithm (the Thresholding algorithm) that makes use of the revealed blockage information to obtain better results. By means of these benchmark algorithms, we numerically analyze how important it is to take into account the revealed information.

5.3.1. The MIP-based algorithm

We give a deterministic algorithm where the routing logic is based on the solution of an offline MIP model. Hence, we call this algorithm the *MIP-based* algorithm. The MIP-based algorithm disregards the unblocking times of blocked edges by setting the unblocking times of all blocked edges to zero. Next, it

solves the offline MIP model which is presented in Section 3.1 to determine a walk which connects the graph with minimum total cost. Then, it traverses the walk and unblocks all blocked edges which are on the walk to connect the graph. Since the MIP-based algorithm requires solving the offline MIP model, its worst-case complexity is exponential.

MIP-based algorithm

Input:

1: $G = (V, E)$, B, c_{ij}, v_s, C_q.
2: $b_{ij} = 0 \quad \forall (i, j) \in B$ ▷ set the unblocking times to zero.
3: W^* = optimal offline walk ▷ using the model in Section 3.1.
4: Traverse W^* and unblock all edges on it to connect G
5: $ObjVal$ = cost of W^* **return:** $ObjVal$

5.3.2. The Greedy Algorithm

We also develop a deterministic algorithm based on greedy choices. We call this algorithm the *Greedy* Algorithm (GA). At each iteration, the GA goes to the closest node v^* that is an end-node of a blocked edge that connects two components, i.e., is an effective blocked edge. Among the edges emanating from v^*, the algorithm unblocks the one whose traversal time plus unblocking time is minimum. The procedure is repeated until the graph is connected. The GA can be regarded as a simplified version of the Thresholding algorithm where shortest paths are calculated at most $k < |E|$ times, hence its worst-case complexity is at most $O(|E| * |V|^3)$.

5.4. Demonstration of the algorithms on a sample instance

We select a small-sized instance to show how the four algorithms presented above work. Figure 5 illustrates this instance that has 15 nodes. The intact edges are depicted by solid lines and the blocked edges by dashed lines. The depot is at node 1. As it can be observed, Figure 5 gives the graph G_B which represents the damaged road network. The network is divided into four components with node sets $V_1 = \{1, 2, 3, 4, 5, 6, 10\}$, $V_2 = \{7, 8, 9\}$, $V_3 = \{11, 12, 13\}$, and $V_4 = \{14, 15\}$, where V_q is the node set of component q. The traversal, c_{ij}, and unblocking times, b_{ij}, of the edges are given in Table 2.

Figure 6 shows the optimal solution to the offline model given in Section 3.1. In this solution, the team leaves the depot at node 1 and goes to node 3. It then goes from node 3 to 10 through nodes 2, 5 and 6 without opening a blocked edge. Next the road restoration team unblocks edge (10,14) with a cost of 360 units, and then goes to node 15 to open edge (12,15) with a corresponding cost of 464 units. Finally, the road clearing team goes from node 12 to 11 and then opens (8,11) with 483 units of cost and finishes its path. The overall cost corresponding to traversal of the links and opening the blocked edges is 1,567 units.

Figure 7 illustrates the solution found by the MIP-based algorithm given in Section 5.3.1. In this algorithm, the times/costs to unblock the edges are neglected and a model similar to the one given in Section 3.1

Greedy algorithm

Input:

1: $G = (V, E)$, B, c_{ij}, v_s, C_q.

2: **Initiation:**

a: $B^e = B$ ▷ B^e : set of effective unobserved blocked edges

b: $\mathcal{A} = \emptyset$ ▷ $\mathcal{A}$: set of accessible end-nodes of B^e from current location

c: $\mathcal{L} = v_s$ ▷ $\mathcal{L}$: location of the team

d: $R = \emptyset$ ▷ R : set of opened blocked edges

e: $G_R = (V, (E \setminus B) \cup R)$ ▷ G_R : restored graph

f: $ObjVal = 0$ ▷ $ObjVal$: corresponding objective value

3: $B' = \{(i, j) \in B$ if $i, j \in V_q \quad \forall q \in \{1, ..., Q\}\}$ ▷ B' : set of non-effective blocked edges

4: $B^e = B^e \setminus B'$

5: $\mathcal{A} = \{i : \quad \forall (i, j) \in B^e$ if node i is accessible from $\mathcal{L}\}$
t_v = shortest path distance from $\mathcal{L}$ to v in G_R, $v \in \mathcal{A}$
$v^* = v : t_v < t_w, w \in \mathcal{A}$

6: $\mathcal{L} = v^*$ and update $ObjVal$

7: $S_{v^*} = \{(v^*, w) : (v^*, w) \in B^e\}$

8: $(v^*, w^*) = argmin\{c_{vw} + b_{vw} \quad \forall (v, w) \in s_{v^*}\}$

9: $B^e = B^e \setminus S_{v^*}$

10: $\mathcal{L} = w^*$ and update $ObjVal$

11: $R = R \cup \{(v^*, w^*)\}$

12: Update $G_R = (V, (E \setminus B) \cup R)$

13: **If** G_R is connected.
Terminate **return:** $ObjVal$ ▷ Possible termination point

14: **Else:**
Go to 2.

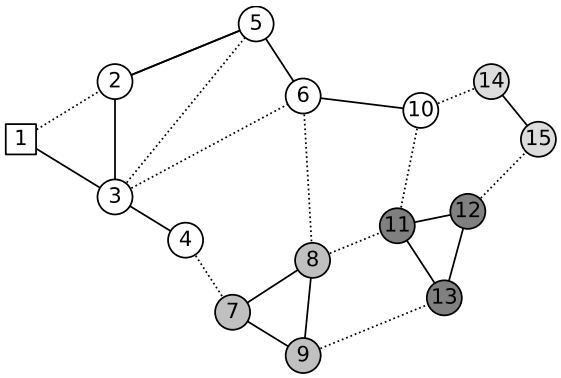


Figure 5: An ORCCP instance

is solved to obtain the route of the team by setting b_{ij} to 0 for each blocked edge (i, j). Since the unblocking times are ignored, it is possible to open blocked edges with very high unblocking times. For example, in this solution the team opens link (4,7) with a cost of 1000 units. The corresponding tour length including road traversal and restoration on blocked edges is 2,111, yielding a competitive ratio of 1.34.

Figure 8 provides the solution found by the greedy algorithm, GA, whose description is given in Section 5.3.2. In this algorithm, the team moves to the closest effective blocked edge and restores it. Similar to the

Table 2: Traversal and unblocking times

Edge	c_{ij}	Edge	c_{ij}	Blocked Edge	b_{ij}
(1,2)	28	(7,9)	21	(1,2)	588
(1,3)	28	(8,9)	33	(3,5)	1206
(2,3)	40	(8,11)	21	(3,6)	689
(2,5)	36	(9,13)	36	(4,7)	1000
(3,4)	21	(10,11)	40	(6,8)	1482
(3,5)	67	(10,14)	18	(8,11)	483
(3,6)	53	(11,12)	15	(9,13)	936
(4,7)	26	(11,13)	26	(10,11)	560
(5,6)	26	(12,13)	30	(10,14)	360
(6,8)	57	(12,15)	29	(12,15)	464
(6,10)	25	(14,15)	22		
(7,8)	24				

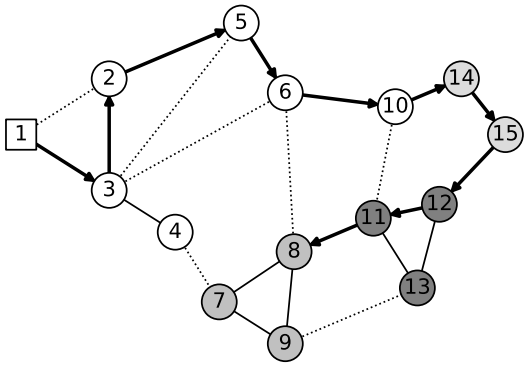


Figure 6: The offline optimal solution

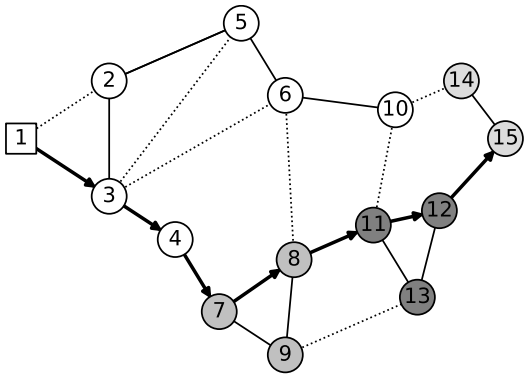


Figure 7: The solution output by the MIP-based algorithm

MIP-based algorithm, the observed blockage information is not utilized in this approach and consequently, in this instance we can observe that edges (4,7) and (9,13) with high blockage costs are opened. The tour length including the road traversal and unblocking times is 2,591, thereby corresponding to a high competitive ratio value of 1.65.

Figure 9 shows the solution found by both the OREO and the Thresholding algorithms for the instance

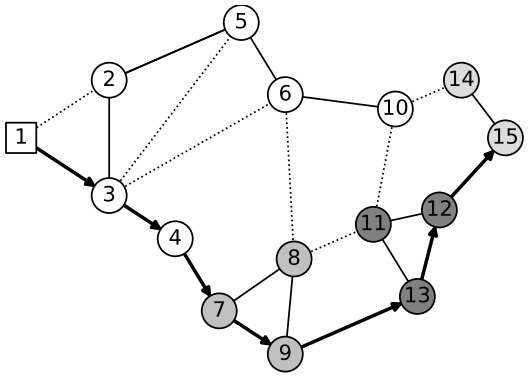


Figure 8: The solution output by the greedy algorithm

given in Figure 5. In these algorithms, the revealed information of the blocked edges is taken into account before opening them and in this particular example, both of these algorithms found the same solution. Since the unblocking times of the blocked edges are not known, the road clearance team first visits a few nodes to collect the blockage information and then decides which blocked links to open. For example, when the team leaves the depot, it goes to node 3 and then to 4 to collect the blockage information on link (4,7). Then, the road restoration team returns to node 3 and from there goes to node 10 through nodes 2, 5 and 6. At node 10, the team collects the blockage information of (10,14) and (10,11) and then decides to open (10,14) incurring 360 units of cost. Once (10,14) is opened, the team goes to node 15 and opens (12,15) with a corresponding cost of 464 units. The team then goes to 11 from node 12. At node 11, the road clearance team first collects the unblocking information of (8,11), which is 483 units, but then decides to collect more information before opening this blocked link and hence goes to node 13. At node 13, the road restoration team observes the unblocking time of (9,13), which is 936. Given that this unblocking time is very large, the road clearance team goes back to node 11 and opens (8,11) instead. The overall length of this route for the road clearance team, including all the traversal and road restorations is 1,661 units, which corresponds to a very low competitive ratio value, 1.06.

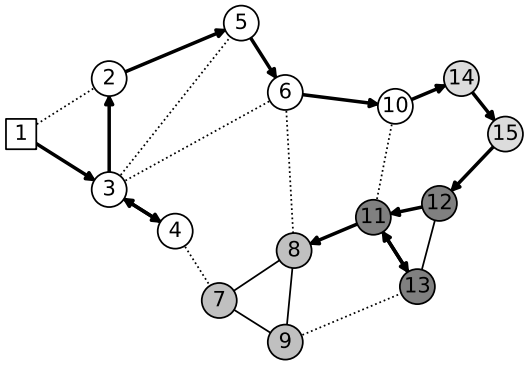


Figure 9: The solution output by the Thresholding and OREO algorithms

6. Computational Study

Since the offline version of the ORCCP was first studied in Kasaei & Salman (2016), we use the same data sets to test our proposed algorithms. These data sets include both the Istanbul city road network (under low, moderate, and high disaster scenarios) as well as some instances obtained from *Rural Postman Problem (RPP)* instances available in the literature. In the following, we give the description of the networks together with the results we obtained from testing the four discussed algorithms. The 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 mathematical models using Gurobi Optimizer 9.0 in the Python environment.

6.1. Istanbul road network

The road network of Istanbul was constructed by considering province centers and actual road distances. In this road network, nodes are identified by considering strategically important locations such as province centers, disaster coordination centers, relief aid depots, harbors, airports, coach bus terminals, fire stations and hospitals. The nodes are depicted in Figure 10. Yellow squares in Figure 10 show the potential supply nodes and the red ones denote the demand points. Although we are not planning the satisfaction of the demand from the supply nodes, we ensure that the network is reconnected so that emergency items can be delivered from the supply nodes to the demand nodes in the next stage. The blue points in Figure 10 show the junction nodes that represent an intersection point where several or different types of routes are linked. The visual representation of the network model generated from the road network of the Istanbul city showing the nodes together with the edges is also given in Figure 11.

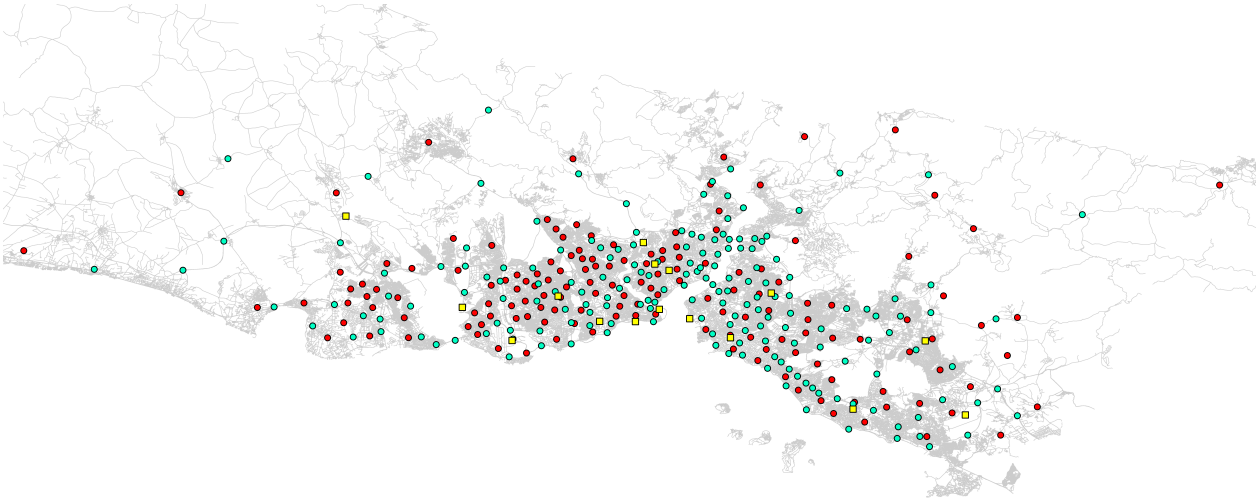


Figure 10: Istanbul network nodes

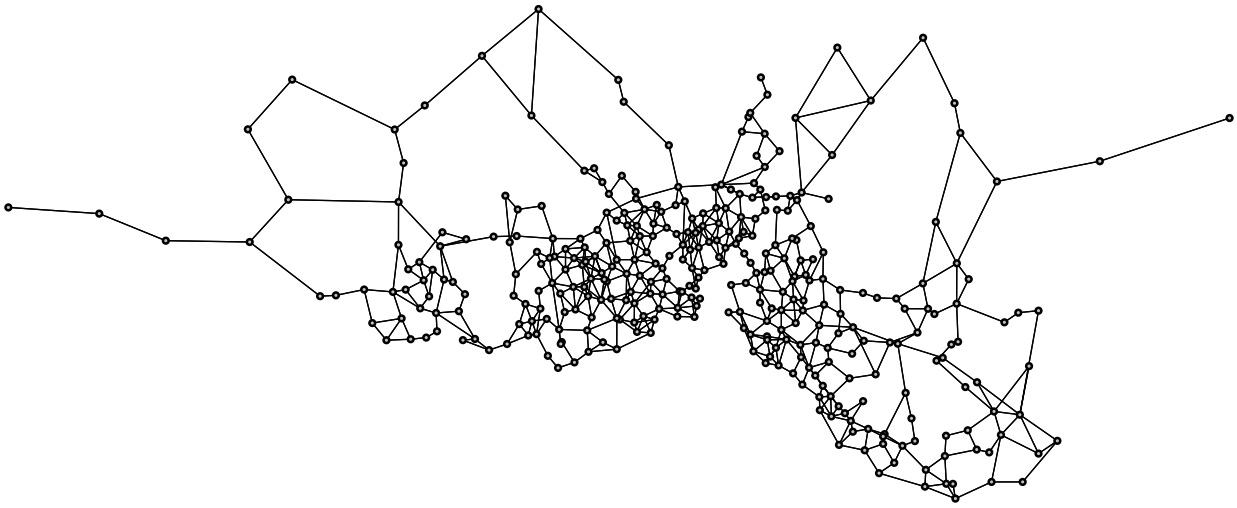


Figure 11: Model road network of Istanbul

The edges represent shortest routes among demand points, potential supply points and road junction points in the network. The distances are considered to be symmetric, i.e, the travel times are the same for both directions for a pair of nodes. Travel times are determined according to road distances considering 40 km/h average speed for the team.

Blocked edges are selected randomly, where blockage probability of each link (i, j) is calculated with respect to a likely earthquake scenario by considering three criteria: 1) the seismic zone factor based on where edge (i, j) is located, 2) its distance from the fault line, and 3) a measure of disaster intensity or magnitude. We note that these probabilities are only used to generate the blocked edges in the instances of our data set and are not related to the developed algorithms.

The unblocking time of a blocked edge is generated based on the damage level on that link, which can be one of the three types: 1) light debris or destruction, 2) bypassable heavy debris or destruction, and 3) non-bypassable heavy debris or destruction. Further explanations on how these blocked edges and their unblocking times are generated can be found in Kasaei & Salman (2016).

Table 3 gives the characteristics of the instances generated from the Istanbul road network. In this table, the first column gives the name of each instance according to its disaster magnitude. Column $|V|$ gives the number of nodes in the network, column $|E|$ gives the number of edges, column k is the number of blocked edges in that instance and column Q is the number of connected components of the particular instance. The last column in Table 3 shows the disaster magnitude. From Table 3, it can be observed that in disaster scenarios with higher number of blocked edges, the number of connected components in the network increases.

We tested the offline model and the four algorithms to solve the ORCCP for all the instances given in Table 3. The results are presented in Table 4. The **Exact MIP RT** gives the CPU run time (RT) of the MIP model to solve the offline problem in seconds. In here, we point out that the focus of this study is not to

Table 3: Characteristics of the Istanbul network instances

Sample name	$\|V\|$	$\|E\|$	k	Q	**Disaster Magnitude**
L1	349	647	172	12	Low
L2	349	647	192	14	Low
L3	349	647	200	15	Low
L4	349	647	204	17	Low
M1	349	647	240	26	Medium
M2	349	647	252	29	Medium
M3	349	647	226	30	Medium
M4	349	647	250	35	Medium
H1	349	647	310	38	High
H2	349	647	284	39	High
H3	349	647	311	44	High
H4	349	647	320	44	High

test the offline model as it has been studied in the literature before. The reason to solve the offline model is to extract the optimal offline values for evaluation of the algorithms and calculation of the competitive ratio values. The columns labeled by **Thresholding Algorithm** give both the run time in seconds and the corresponding competitive ratio value (CR) of this algorithm. The competitive ratio of an online algorithm is the obtained objective function value of that algorithm divided by the optimal objective function value of the offline problem. Similarly, in columns labeled **OREO algorithm**, **MIP-based algorithm** and **Greedy algorithm**, the performance of these algorithms in terms of run time (RT) in seconds and competitive ratio (CR) for the same instances is given.

From Table 4, we can see that the Thresholding and OREO algorithms consistently give better competitive ratio values compared to both MIP-based and Greedy algorithms. While the maximum competitive ratio over these instances remains under 1.47 and 1.29 for the Thresholding and OREO algorithms, respectively, the maximum competitive ratio of the MIP-based algorithm rises up to 3.65 and the maximum competitive ratio of the Greedy algorithm goes as high as 3.10. Similarly, the average of the CR values for the Thresholding algorithm is 1.26 with a variance of 0.01, whereas the average CR values are 2.93 and 2.24 for MIP-based and Greedy algorithms, respectively. In case of the OREO algorithm, the average CR is 1.17 and the variance is 0.005. While the OREO algorithm obtains the best average and maximum CR values, the Thresholding algorithm performs nearly as well.

It can further be observed that the number of blocked edges and consequently the number of connected components do not affect the CR values of the Thresholding and OREO algorithms. For instance, using the Thresholding algorithm, the average of the CR values for the low impact cases is 1.32. This value is 1.26 and 1.21 for moderate impact and high impact Istanbul scenarios, respectively. Since in online optimization, the performance of the algorithms is compared based on the driven CR values, it can be observed that in these instances, in terms of the quality of the found solutions, the Thresholding and OREO algorithms outperform

both benchmark algorithms significantly. We note that in the online optimization literature (see e.g. Zhang *et al.* (2019) and Zhang *et al.* (2015)), algorithms that offer a CR of less than 2 have been interpreted to perform well.

From the comparison between the obtained CR values for each of the algorithms, we confirmed the superiority of OREO and Thresholding algorithms over the benchmark algorithms. However, an analysis of the run times justifies the need to develop the Thresholding algorithm. While the maximum run time of the Thresholding algorithm over the tested instances remains under 30.14 seconds and the average run time of this algorithm over all the tested instances is only 9.05 seconds, the running time of the OREO algorithm goes up to 74,245.50 seconds (which corresponds to more than 20 hours of CPU run time). Moreover, the running time of the OREO algorithm is significantly impacted by the percentage of the blockage in the graph. For instance, the average run time of the OREO algorithm over the low impact instances is 3777.76 seconds, but it increases to 18,561.85 seconds for the medium impact and 44,961.75 seconds for the high impact instances. Although the run time of the Thresholding algorithm also increases slightly with more blocked edges, the average run time of the high impact scenarios remains under only 15.40 seconds. This shows that although the OREO algorithm is capable of finding good solutions, its running time is very high and is not applicable in real-life instances where the post-disaster plans need to be made as soon as possible. In here, we also point out that since the quality of the solutions obtained by both of the benchmark algorithms is poor, they both are impractical even if they have low run times.

Table 4: Results of the Istanbul network instances

Sample	Exact MIP	**Thresholding Algorithm**		**OREO Algorithm**		**MIP-based Algorithm**		**Greedy Algorithm**	
Name	RT	CR	RT	CR	RT	CR	RT	CR	RT
L1	22.46	1.33	2.41	1.25	1024.32	3.37	52.88	1.41	0.15
L2	41.35	1.34	3.88	1.20	2635.39	2.83	147.27	3.10	0.13
L3	33.97	1.47	1.34	1.29	938.67	3.65	670.81	2.63	0.22
L4	401.63	1.12	11.59	1.12	10512.64	1.75	148.93	1.55	0.49
M1	119.48	1.27	9.28	1.15	20984.33	2.09	58.30	1.67	0.60
M2	555.59	1.27	4.96	1.15	20085.69	3.56	583.91	2.36	0.63
M3	753.09	1.33	10.64	1.25	22422.51	2.62	1277.21	2.29	0.87
M4	59.99	1.15	2.98	1.10	10754.87	2.65	61.49	2.47	0.66
H1	106.42	1.23	8.26	1.10	26825.52	3.50	61.32	2.57	1.06
H2	188.46	1.23	18.09	1.19	74245.50	3.54	75.41	2.41	1.33
H3	358.93	1.11	4.83	1.06	41512.01	2.37	898.62	1.70	1.66
H4	45.33	1.28	30.41	1.12	37263.99	3.16	422.15	2.76	3.63

Figure 12 provides further insights to the results presented in Table 4. On the left side of this figure, a comparison between objective function values of the offline model, the OREO algorithm and the Thresholding algorithm is given. As it can be observed, the performance of both Thresholding and OREO algorithms is very good in terms of the found solutions and they both find solutions that are close to the offline opti-

mum. As expected, the objective function values increase when the number of blocked edges increases in the higher impact scenarios. The right side of Figure 12 provides a comparison between the values found using Thresholding and OREO algorithms by reporting $\Phi = \frac{Z_{TA} - Z_{OREO}}{Z_{OREO}}$, where Z_{TA} and Z_{OREO} are the objective function values corresponding to Thresholding and OREO algorithms, respectively. It can be observed that the Thresholding algorithm provides very good solutions, in addition to having short run times.

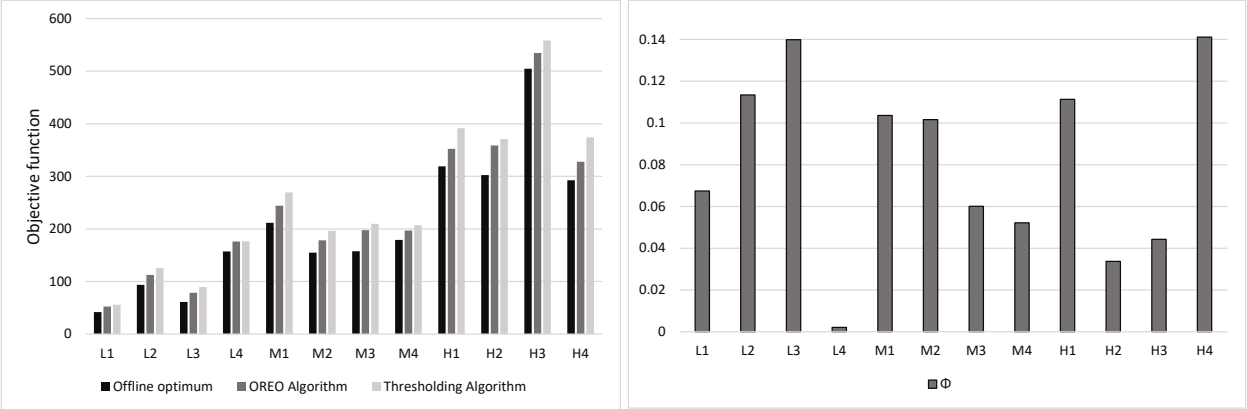


Figure 12: Comparison of the solutions found by the OREO and Thresholding algorithms on Istanbul instances

6.2. *RPP Instances*

In the well-studied Chinese Postman Problem (CPP), given a graph $G = (V, E)$, the problem is to determine a minimum cost closed walk, traversing each edge of G at least once. The RPP is an extension of the CPP in which a subset of the edges should be traversed at a minimal cost. Due to the similarity of the offline version of the problem studied here to the RPP, we used the same instances adopted from the RPP literature in Kasaei & Salman (2016) to further analyze the performance of our algorithms. In these instances, the edges that should be traversed in the RPP are set as blocked edges in our problem. Further details on how the blocked edges and their unblocking times are generated can be found in Kasaei & Salman (2016). The data sets from the RPP literature are derived from the road networks of Albaida and Madriguera cities in Spain. Table 5 summarizes the characteristics of these instances. The instances starting by ALBA and I are from Albaida and the instances starting by MADR are from Madriguera. We have also tested our algorithms on RPP instances denoted by B. These instances are larger than the rest of the instances in terms of the number of nodes and edges and are tested to provide further insights on the performance of our algorithms.

The results of solving the optimal offline model, Thresholding, OREO and the benchmark algorithms for the ORCCP on the adapted RPP instances are given in Table 6. The columns in this table are labeled similar to those in Table 4. We set a time limit of one hour for the solution of the offline model. Except for three instances (B351, B371 and B451), the optimal offline solutions are found within the one hour time limit. For the instances in which the optimal solution could not be verified in one hour, the optimality gaps are 0.6 % for B351, and 3% for both B371 and B451. To calculate the corresponding competitive ratio values for these

Table 5: Characteristics of the RPP instances

Sample Name	$\|V\|$	$\|E\|$	k	Q	Sample Name	$\|V\|$	$\|E\|$	k	Q
I20	50	98	63	32	MADR31	196	316	86	9
I21	49	110	67	26	MADR32	196	316	108	17
I22	50	184	74	20	MADR33	196	316	102	17
I23	50	158	78	23	MADR34	196	316	101	12
I24	41	125	55	20	MADR35	196	316	95	6
ALBA31	116	174	51	12	MADR53	196	316	148	38
ALBA32	116	174	46	6	MADR55	196	316	147	36
ALBA33	116	174	44	12	B351	453	811	357	77
ALBA34	116	174	49	14	B352	453	811	361	69
ALBA35	116	174	57	15	B371	490	873	384	84
ALBA51	116	174	88	37	B372	490	873	387	77
ALBA52	116	174	92	38	B451	465	1055	377	20
ALBA55	116	174	91	35					

instances, we used the best found lower bounds within the one hour time-limit.

We set a half an hour run time limit for the MIP-based algorithm. All the tested instances except for B531, B352, B371 and B372 are solved to optimality within half an hour (1800 seconds) for the generation of initial solutions by the MIP-based algorithm. For the instances that were not solved optimaly, we extracted the best found solution within the time-limit and generated the route of the team. For the OREO algorithm however, we did not set up a time limit to show how the run times increase exponentially when the size of the network increases.

With regards to the performance of the algorithms, similar observations to those made on the Istanbul data sets can be made for the RPP instances. Both Thresholding and OREO algorithms are capable of finding very good solutions and outperform the benchmark algorithms considerably. The Thresholding algorithm provides very good solutions by an average competitive ratio of 1.18 and a variance of 0.03 with the largest and smallest competitive ratio values being 1.79 and 1.00 for B451 and MADR33, respectively. The average competitive ratio of the Thresholding algorithm for the I instances is 1.16; for the ALBA instances, this average is 1.17 and for the MADR and B instances this average is 1.19 and 1.21, respectively. The OREO algorithm also shows a very consistent and good performance in terms of finding high quality solutions. The average competitive ratio of the OREO algorithm over all the RPP instances is only 1.14.

An analysis of the benchmarking algorithms shows that neglecting revealed blockage information can result in generating solutions with very high CR values. For the MIP-based algorithm, the average CR value is 21.63 with a maximum of as high as 136.01. This confirms that the MIP-based algorithm is not reliable, especially in cases with high variability in the unblocking times, for instance when there are some blocked edges with very high unblocking times. Similarly, the Greedy algorithm shows weak performance in terms of the quality of the found solutions. The average CR of this algorithm over the RPP instances is 10.03 with a standard deviations of 16.52 and a maximum CR value of 65.64 for the ALBA31 instance.

Given that the OREO algorithm can find very good solutions that are sometimes better than the ones found by the Thresholding algorithm, comparing their run time can justify the need for developing the Thresholding algorithm. The running time of the OREO algorithm increases considerably when the size of the network and the percentage of the blocked edges increase. The average run time of this algorithm on MADR instances is 982.86 seconds and it is as high as 69,827.43 seconds for the B instances. As it can be observed in Table 6, the run time of the OREO algorithm on B instances is very high and even more than one day for the B371 instance. Although using the OREO algorithm, a very good solution was found for B371 with a competitive ratio of 1.12, the CPU run time of this instance was 134,060.30 seconds (37.23 hours), which confirms impracticality of this algorithm in solving real-life post-disaster planning problems. While the run time of the OREO algorithm increases significantly as the size of the network increases, the maximum run time of the Thresholding algorithm remains under 30 seconds in all of the tested instances. The average run times of the Thresholding algorithm over I and ALBA instances are 0.63 and 0.32 seconds, respectively. This value increases to 4.05 and 9.94 seconds for MADR and B instances but still remains very low even for instances with up to 465 nodes, 1055 edges and 377 blocked edges.

Table 6: Results of the RPP instances

Sample	Exact MIP	**Thresholding Algorithm**		**OREO Algorithm**		**MIP-based Algorithm**		**Greedy Algorithm**	
Name	RT	CR	RT	CR	RT	CR	RT	CR	RT
I20	2.38	1.01	0.42	1.01	8.10	3.04	0.30	1.82	0.08
I21	2.23	1.16	0.62	1.17	7.89	2.43	0.64	2.40	0.06
I22	1.05	1.26	0.62	1.22	6.24	67.67	0.70	44.03	0.06
I23	4.23	1.19	1.07	1.13	10.79	2.50	0.37	2.85	0.16
I24	0.76	1.19	0.43	1.13	4.53	19.10	0.46	3.04	0.06
ALBA31	0.67	1.30	0.36	1.31	13.90	66.12	1.26	65.64	0.10
ALBA32	0.60	1.00	0.10	1.00	15.12	1.29	1.53	1.29	0.10
ALBA33	1.03	1.21	0.13	1.20	45.78	1.79	0.81	1.77	0.07
ALBA34	1.31	1.18	0.40	1.22	19.87	26.99	1.02	26.77	0.06
ALBA35	0.80	1.30	0.29	1.09	16.43	24.45	0.68	1.35	0.04
ALBA51	3.45	1.01	0.29	1.02	156.04	4.09	8.87	2.61	0.12
ALBA52	7.88	1.13	0.43	1.13	167.32	31.30	1.85	31.33	0.14
ALBA55	11.03	1.22	0.59	1.21	201.87	11.58	2.96	1.59	0.10
MADR31	18.72	1.37	1.72	1.15	231.00	1.77	28.87	1.93	0.09
MADR32	19.95	1.36	3.32	1.34	1254.04	22.88	4.21	1.84	0.18
MADR33	20.89	1.00	3.88	1.00	303.63	136.01	2.90	4.52	0.18
MADR34	13.63	1.01	1.21	1.01	390.47	1.07	15.85	2.02	0.13
MADR35	5.12	1.23	0.21	1.32	35.49	1.50	2.51	1.29	0.03
MADR53	839.50	1.29	5.00	1.28	2064.27	2.82	143.57	2.56	0.88
MADR55	175.15	1.03	13.01	1.02	2601.12	1.08	112.87	2.02	0.60
B351	3600	1.05	4.12	1.02	38268.66	6.60	1800	4.78	2.72
B352	1350.99	1.05	9.22	1.04	70101.18	5.20	1800	5.35	2.75
B371	3600	1.16	3.42	1.12	134060.30	54.72	1800	31.00	1.66
B372	852.99	1.03	4.54	1.02	57099.82	8.57	1800	4.51	1.43
B451	3600	1.79	28.38	1.49	49607.20	36.20	120.32	2.33	0.67

Figure 13, provides further insights on the comparison between Thresholding and OREO algorithms. In this figure, similar to Figure 12, we reported $\Phi = \frac{Z_{TA} - Z_{OREO}}{Z_{OREO}}$ over RPP instances to compare these algorithms. Surprisingly, out of the 25 RPP instances, in 10 instances the Thresholding algorithm was able to find better solutions compared to the OREO algorithm. This further confirms the good performance of the Thresholding algorithm and shows that this algorithm can be a very good tool for this online problem.

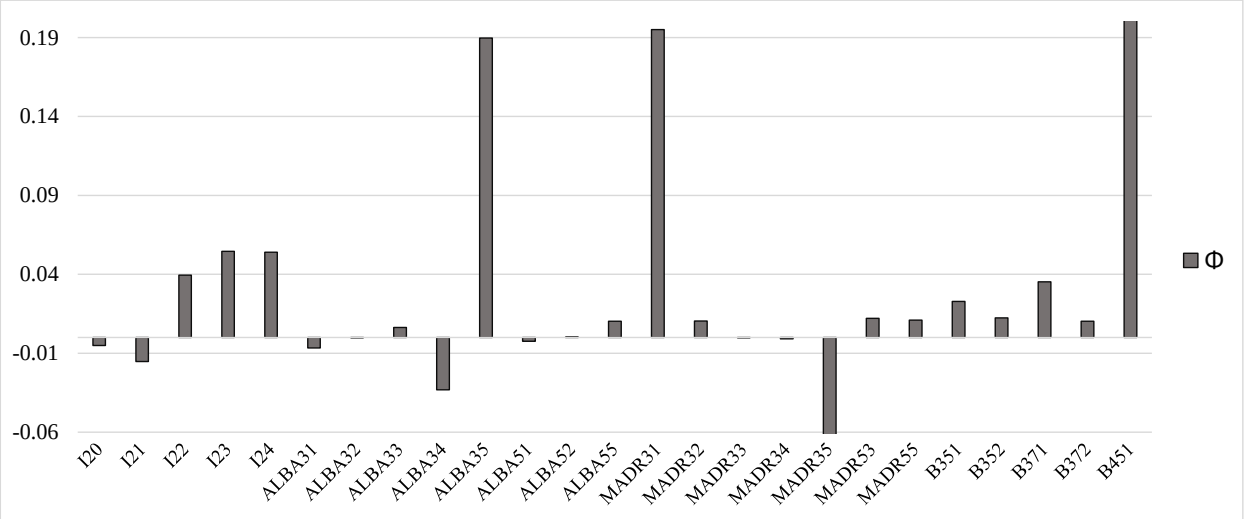


Figure 13: Comparison of the OREO and Thresholding algorithms on RPP instances

7. Conclusion

In this study, we investigated a road restoration problem to achieve network connectivity after a disaster with an online optimization approach for the first time. In this problem, the road network becomes disconnected due to damaged road segments and the times to restore the blocked links are uncertain. A road restoration team is initially positioned at a depot node and aims to regain the connectivity of the network by unblocking a subset of the blocked edges in minimum time. The unblocking time of a blocked edge is not known to the clearing team and is revealed online when the team visits one of the end-nodes of the blocked edge. We analyzed this problem from competitive analysis perspective and provided a tight lower bound on the competitive ratio of deterministic online algorithms. We also developed a deterministic online algorithm and showed its optimality by proving that it meets the obtained tight lower bound over all instances of the problem.

With the aim of providing fast and efficient algorithms for realistic-sized problems defined on real-life road networks, we proposed a novel polynomial time heuristic algorithm that we named the Thresholding algorithm, together with two benchmark deterministic online algorithms. We have also developed an algorithm that is based on observing and re-optimizing (OREO) the route and schedule of the road clearance team when new information is revealed. We measured the performance of our proposed algorithms on real-life

data sets based on the Istanbul city road network, as well as other instances from the literature corresponding to two cities in Spain. The computational experiments reveal that the two benchmark algorithms that do not utilize the observed information adequately fail to yield reasonable competitive ratio values. On the other hand, the OREO algorithm is capable of finding good solutions, but its running time increases enormously when the size of the networks increases and becomes impossible to use this algorithm for real-life instances where a decision should be made in a very short time. However, the Thresholding algorithm yields competitive ratio values nearly as good as the OREO algorithm, while at the same time enjoying very short CPU run times. Our computational results confirmed the efficiency and the good performance of the proposed Thresholding algorithm against the offline optimum, the OREO and the other two benchmark algorithms.

Analyzing an extension of the ORCCP in the presence of multiple teams is a challenging immediate research direction. Two approaches may be taken. First, for simplicity, the teams may be assumed to be uniform, supposing they each have multiple capabilities that are sufficient to handle the required tasks at a blockage. Second, in a more realistic setting, these teams may differ in terms of their capabilities since the machinery, equipment, tools, consumables and operating personnel required for a specific restoration task may show variability. Given a set of limited resources, team forming can be a part of the problem, after an initial assessment of the required tasks. Moreover, some equipment may even be repositioned among the teams working at different locations.

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 locations to be serviced are actually revealed online to the service provider (Jaillet P. 2008). 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.

References

Abazari, Seyed Reza, Aghsami, Amir, & Rabbani, Masoud. 2020. Prepositioning and distributing relief items in humanitarian logistics with uncertain parameters. *Socio-Economic Planning Sciences*, 100933.

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.

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

Akbari, Vahid, & Salman, F. Sibel. 2017b. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. *European Journal of Operational Research*, **257**(2), 625–640.

Akbari, Vahid, & Salman, Sibel. 2014. A Model-Based Heuristic to the Min Max K-Arc Routing for Connectivity Problem. *Pages 76–88 of: 4th Student Conference on Operational Research*. OpenAccess Series in Informatics (OASIcs), vol. 37.

Akbari, Vahid, & Shiri, Davood. 2021. Weighted online minimum latency problem with edge uncertainty. *European Journal of Operational Research*.

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

Aksu, Dilek Tuzun, & Ozdamar, Linet. 2014. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, **61**, 56–67.

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

Baxter, A. E., Wilborn Lagerman, H. E., & Keskinocak, P. 2020. Quantitative modeling in disaster management: A literature review. *IBM Journal of Research and Development*, **64**(1/2), 3:1–3:13.

Caunhye, Aakil M, Aydin, Nazli Yonca, & Duzgun, H Sebnem. 2020. Robust post-disaster route restoration. *OR Spectrum*, **42**(4), 1055–1087.

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

Çelik, Melih, Ergun, Özlem, & Keskinocak, Pınar. 2015. The Post-Disaster Debris Clearance Problem Under Incomplete Information. *Operations Research*, **63**(1), 65–85.

Chan, Timothy M. 2007. More Algorithms for All-Pairs Shortest Paths in Weighted Graphs. *Page 590–598 of: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. STOC '07. New York, NY, USA: Association for Computing Machinery.

CRED. 2013. *People affected by conflict: Humanitarian needs in numbers*. Tech. rept. Centre for Research on the Epidemiology of Disasters, Brussels.

CRED, & UNDRR. 2020. *Human cost of disasters: An overview of the last twenty years 2000-2019*. Tech. rept. Centre for Research on the Epidemiology of Disasters and UN Office for Disaster Risk Reduction.

Duque, Pablo A. Maya, Dolinskaya, Irina S., & Sörensen, Kenneth. 2016. Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, **248**(1), 272 – 285.

EEFIT. 2003. *The Kocaeli, Turkey Earthquake of 17 August 1999*. Tech. rept. A Field Report by the Earthquake Engineering Field Investigation Team (EEFIT) of the Institution of Structural Engineers.

Iloglu, Suzan, & Albert, Laura A. 2020. A maximal multiple coverage and network restoration problem for disaster recovery. *Operations Research Perspectives*, **7**, 100132.

Jaillet, Patrick, & Stafford, Matthew. 2001. Online Searching. *Operations Research*, **49**, 501—515.

Jaillet P., Wagner M.R. 2008. 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.

Karlin, Anna R., Manasse, Mark S., Rudolph, Larry, & Sleator, Daniel D. 1988. Competitive snoopy caching. *Algorithmica*, **3**, 79—119.

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.

Komm, Dennis. 2016. *Introduction to Online Computation*. Springer.

Li, Shuanglin, Ma, Zujun, & Teo, Kok Lay. 2020. A New Model for Road Network Repair after Natural Disasters: Integrating Logistics Support Scheduling with Repair Crew Scheduling and Routing Activities. *Computers & Industrial Engineering*, 106506.

Liao, Chung-Shou, & Huang, Yamming. 2015. Generalized Canadian traveller problems. *Journal of Combinatorial Optimization*, **29**, 701—712.

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.

Moreno, Alfredo, Munari, Pedro, & Alem, Douglas. 2020a. Decomposition-based algorithms for the crew scheduling and routing problem in road restoration. *Computers & Operations Research*, **119**, 104935.

Moreno, Alfredo, Alem, Douglas, Gendreau, Michel, & Munari, Pedro. 2020b. The heterogeneous multicrew scheduling and routing problem in road restoration. *Transportation Research Part B: Methodological*, **141**, 24–58.

Morshedlou, Nazanin, González, Andrés D., & Barker, Kash. 2018. Work crew routing problem for infrastructure network restoration. *Transportation Research Part B: Methodological*, **118**(dec), 66–89.

Özdamar, Linet, Aksu, Dilek Tüzün, & Ergüneş, Biket. 2014. Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, **48**(4), 249–262.

Ransikarbum, Kasin, & Mason, Scott J. 2016. Goal programming-based post-disaster decision making for integrated relief distribution and early-stage network restoration. *International Journal of Production Economics*, **182**, 324–341.

Sayarshad, Hamid R., Du, Xinpi, & Gao, H. Oliver. 2020. Dynamic post-disaster debris clearance problem with repositioning of clearance equipment items under partially observable information. *Transportation Research Part B: Methodological*, **138**, 352 – 372.

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. New Variations of the Online k-Canadian Traveler Problem: Uncertain Costs at Known Locations. *Probability, Combinatorics and Control, IntechOpen*.

Shiri, Davood, & Salman, F. Sibel. 2020. Online Optimization of First-responder Routes in Disaster Response Logistics. *IBM Journal of Research and Development*, **64**, 13:1–13:9.

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.

Tokuyama, Hideo. 2012. Learning From Japan's Ordeal. *Public Roads*, **75**(6).

Vodák, Rostislav, Bíl, Michal, & Křivánková, Zuzana. 2018. A modified ant colony optimization algorithm to increase the speed of the road network recovery process after disasters. *International Journal of Disaster Risk Reduction*, **31**, 1092–1106.

Yan, Shangyao, & Shih, Yu-Lin. 2009. Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers & Operations Research*, **36**(6), 2049–2065.

Zhai, Wei, & Peng, Zhong-Ren. 2020. Damage assessment using Google Street View: Evidence from Hurricane Michael in Mexico Beach, Florida. *Applied Geography*, **123**, 102252.

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 and 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.

View publication stats