Computers and Operations Research 82 (2017) 52–68

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Multi-vehicle prize collecting arc routing for connectivity problem

Vahid Akbari, F. Sibel Salman*

College of Engineering, Koç University, Sariyer, Istanbul 34450, Turkey

ARTICLE INFO

Article history:
Received 27 June 2016
Revised 6 December 2016
Accepted 16 January 2017
Available online 18 January 2017

Keywords:
Arc routing
Prize collecting
Network connectivity
Road clearance
Disaster response
Mixed integer programming
Matheuristic

ABSTRACT

For effective disaster response, roads should be cleared or repaired to provide accessibility and relief services to the affected people in shortest time. We study an arc routing problem that aims to regain the connectivity of the road network components by clearing a subset of the blocked roads. In this problem, we maximize the total prize gained by reconnecting disconnected network components within a specified time limit. The solution should determine the coordinated routes of each work troop starting at a depot node such that none of the closed roads can be traversed unless their unblocking/clearing procedure is finished. We develop an exact Mixed Integer Program (MIP) and a matheuristic method. The matheuristic solves single vehicle problems sequentially with updated prizes. To obtain an upper bound, we first relax the timing elements in the exact formulation and then solve its relaxed MIP, which decomposes into single vehicle problems, by Lagrangian Relaxation. We show the effectiveness of the proposed methods computationally on both random Euclidean and Istanbul road network data generated with respect to predicted earthquake scenarios.

1. Introduction

Millions of people suffer from natural disasters every year. From the destruction of buildings and infrastructure to the spread of diseases, natural disasters can devastate countries overnight. After a disaster, it is critical to reach the affected areas to provide relief operations such as search and rescue, medical response, aid delivery and establishing temporary shelters. Disasters cause the road networks to be disconnected due to having roads blocked by building debris, fallen lampposts, displaced cars, etc. Moreover, structural damage on the roads also causes blockage. These conditions impede accessibility to casualties, hospitals and supply locations, cause severe handicaps on providing essential resources to the affected people and prevent evacuation activities.

In the immediate disaster response phase, in order to restore the connectivity of the isolated areas, a selected subset of roads should be either rapidly cleared or repaired by road clearing teams consisting of machinery and equipment. We refer to each team as a vehicle from now on. Shortly after information on road conditions is gathered, the vehicles are dispatched from a specific node of the network referred to as the depot.

In this study, we aim to provide an efficient solution method that finds a subset of the closed roads to open and the order in which they should be opened for each vehicle. In order to make the most number of people accessible in short time, we model the problem with the objective of maximizing the collected prize within an imposed time constraint. The prize gained from connecting a network component could be set to the number of people in the isolated component or a priority weight factoring in the locations of facilities such as airports, ports, hospitals and relief supply points within the component. A complexity factor arises since the blocked edges are not traversable unless they are completely unblocked and a vehicle has to wait to enter an edge while it is being unblocked. Hence, node arrival times should be part of the solution. Calculating the arrival time of the vehicles to the nodes complicates the problem since an edge can be traversed multiple times by the same or different vehicles. We call the problem that we study the *Multi-vehicle Prize Collecting Arc Routing for Connectivity Problem* (KPC-ARCP) (here K refers to the number of vehicles). We develop an MIP model to solve the addressed problem. However, the timing related conditions prevent us from finding an optimal solution in realistic-sized instances. Considering that the problem should be solved in short time after the incident takes place, we propose a matheuristic to obtain a near-optimal feasible solution and a relaxation method to derive a tight upper bound to the *KPC-ARCP* which decomposes the problem into single vehicle problems. The heuristic is based on solving K single vehicle MIP models and is easily implementable. Furthermore, the optimality gaps of the heuristic have been found to be zero or very close to zero in the computational study.

* Corresponding author.
E-mail addresses: vakbarighadkolaei@ku.edu.tr (V. Akbari), ssalman@ku.edu.tr (F.S. Salman).

http://dx.doi.org/10.1016/j.cor.2017.01.007
0305-0548/

In the next section, we review the related literature and state our contributions. We define the problem in Section 3 and in Section 4 we present the developed mathematical model and in Section 5 we show our solution approach. Section 6 describes data generation and discusses the computational results. We close with concluding remarks in Section 7.

2. Literature review

The problem addressed in this study is in the class of *Arc Routing Problems* (ARP). The most widely known ARP is the *Chinese Postman Problem* (CPP) that seeks for a least-cost closed walk that traverses all the edges [15]. The *Rural Postman Problem* (RPP) is a widely studied extension of the CPP that determines a minimum cost closed walk containing the set of required edges, in addition to other edges that may be traversed [16]. For other variants of ARP, we refer the reader to Corberán and Laporte [13], which is a thorough and up-to-date book on ARP that also provides a categorization of the problems.

The problem we study is built upon the RPP but differs from it in aspects that we will discuss later. The main difference is that in *KPC-ARCP* achieving connectivity is the main concern and the required edges are not known initially and are selected to enable connectivity. Since our objective maximizes the sum of the collected prizes, below we discuss variants of RPP that have a prize collecting or profit maximization objective and other ARPs with synchronization of the vehicles.

While several versions of ARP concentrate on prize collection, node routing problems that address the same type of objectives were studied earlier. Balas [7] introduced the *Prize Collecting Traveling Salesman Problem* (PCTSP), where a salesman gets a prize for each city that he visits and pays a penalty for every city that he does not manage to visit while minimizing the corresponding travel costs and penalties. The *Orienteering Problem* (OP) is another node routing problem that considers a prize collecting type of objective function. In OP, the objective is to collect as much prize as possible in a given tour length limit. The *Team Orienteering Problem* (TOP) is an extension of OP in which multiple teams and their start and end nodes as well as scores (prizes) of the nodes are specified. The teams should leave the start node and traverse several nodes and end their walks in the end node in a given amount of time. The objective is to determine the path for each of the teams from the start point to the end point in order to maximize the total score. For references on TOP see Gavalas et al. [18].

Archetti et al. [5] introduced the arc routing version of TOP which is called *Team Orienteering Arc Routing Problem* (TOARP). Different from TOP in which the prizes can be collected by traversing nodes, in TOARP the prizes should be collected from the arcs of the input graph. Archetti et al. [5] proposed a metaheuristic that finds an optimal solution in almost all the tested instances having up to 27 nodes and 296 arcs.

Several ARPs having a single vehicle and prize collecting or profit maximization objectives have been studied within the last decade. Aráoz et al. [4] introduced the *Privatized Rural Postman Problem* (PRPP), which is also known as the *Prize Collecting Rural Postman Problem* (PCRPP). In PCRPP, a tour starting and ending at the depot, and maximizing the difference between the total prizes from serving required arcs and the total traversing costs is found. The authors developed an integer programming (IP) model with an exponential number of constraints. Aráoz et al. [3] proposed an iterative algorithm together with a heuristic procedure to solve this problem efficiently. The proposed algorithm solves a relaxation of the PCRPP with fewer number of constraints repetitively, while the heuristic generates feasible solutions that provide lower bounds at each iteration.

Feillet et al. [17] study the *Profitable Arc Tour Problem* (PATP) in which a maximum tour length is imposed and there is an upper bound on the number of times a prize can be collected from an arc. Instances with up to 65 nodes were solved using a branch-and-price algorithm. The *Time-Dependent Prize Collecting Arc Routing Problem* (TD-PARP) is another ARP that focuses on collecting prizes within a given tour length limit. Black et al. [10] introduced the TD-PARP in which time dependency refers to travel times that change with the time of the day. Two metaheuristic algorithms, one based on Variable Neighborhood Search and one based on Tabu Search are proposed and tested. The authors solved instances with up to 600 prize arcs using the metaheuristic algorithms.

KPC-ARCP differs from TOARP, PCRPP, PATP and TD-PARP in several ways. For instance, in *KPC-ARCP* connecting components to the supply (depot) node results in gaining prizes which is different from all of these problems. Furthermore, in *KPC-ARCP* a blocked edge can be traversed only after it is opened by a vehicle. This condition, together with the existence of multiple vehicles, complicates the problem since a certain type of synchronization (coordination) of the vehicle routes should be achieved. Although various synchronization issues have been addressed and studied in the literature as discussed below, to the best of our knowledge, this type of synchronization, which involves incurring extra costs only for the first traversal of an edge among multiple vehicles, has been addressed before only in Akbari and Salman [1] that solves a makespan minimization version of *KPC-ARCP*. In Akbari and Salman [1] the goal is to reconnect all the components of a disconnected road network in shortest time. An exact MIP model is given together with a matheuristic approach that includes a local search procedure. In the following we give two examples of ARP with a synchronization requirement.

The *Synchronized Arc Routing Snow Plowing Problem* proposed in Salazar-Aguilar et al. [24] consists of determining a set of routes such that all streets, some of which have multiple lanes, are plowed using fleets of synchronized vehicles. The objective is to minimize the duration of the longest route, namely the makespan. The street segments have one or two directions, and each direction has a number of lanes, typically between one and three. Synchronization occurs since all the lanes of a particular street should be plowed simultaneously. In another study, Salazar-Aguilar et al. [25] introduced the *Synchronized Arc and Node Routing Problem* which is on a directed graph with two sets of arcs: those that must be painted, and those that do not need to be painted. In this problem, a replenishment vehicle to supply paint and a set of homogeneous painting vehicles are used. While the painting is being processed by the painting vehicles, they might run out of paint and the replenishment vehicle should provide them with more paint. The synchronization refers to choosing the appropriate refill nodes such that the painting vehicles and the replenishment vehicle can meet at the same time or without incurring much waiting. Although synchronization has a different meaning in Salazar-Aguilar et al. [24] and [25], it is similar to *KPC-ARCP* in terms of having timing concerns in the mathematical formulation and the heuristic algorithm. While in Salazar-Aguilar et al. [24] all streets should be plowed and in Salazar-Aguilar et al. [25] the set of edges that require the painting are given, in *KPC-ARCP* the required edges to be opened are not given a priori and its part of the decisions. Moreover, while in *KPC-ARCP* a blocked edge can not be traversed unless it is opened, in the discussed problems the vehicles can traverse any arc without providing the services.

Recently, several studies focused on clearing a road network or improving accessibility after a disaster. Some of these studies do not address routing and instead, they focus on selecting the road segments to be cleared or repaired. Duque and Sörensen [14] consider the case where there is a budget constraint, and a number of non-operative roads need to be repaired after a disaster. Weights

are assigned to the rural towns depending on the importance of the towns. The objective is to minimize the weighted sum of time to travel from each rural town to its closest regional center. Liberatore et al. [21] address the problem of distributing goods to a population affected by an earthquake in a damaged network and decide which edges should be opened given the repair costs and a budget. They address multiple criteria such as delivery time, cost, reliability, security and satisfied demand.

Scheduling of roadway repair operations is studied by Yan and Shih [26], where transportation of relief items is optimized as well. The constraint that road segments should be repaired before specific times is imposed. Therefore, certain tasks have to be scheduled before the other repair tasks due to their importance. In addition, time windows exist and there are supply points from which relief distribution originates. A set of demand points is chosen and each point has to receive a minimum required amount of relief items. The objectives are to minimize the total time needed for emergency repair at all repair points and for sending relief supplies to all demand points. The authors construct a multi-objective, multi-commodity network flow model on a time-space network and develop a heuristic method. Different from our study, the authors focus on mainly upgrading a road network and are not concerned with routing of the vehicles.

Ozdamar et al. [22] introduced a debris clean up planning problem with a specified number of vehicles. An accessibility measure that considers shortest path lengths between pairs of nodes over time is introduced. The shortest paths may change while the roads are restored, as in our problem. However, this study only assigns the road restoration tasks to the vehicles and does not involve their routing. Aksu and Ozdamar [2] propose a path-based time-indexed IP model that identifies the blocked links to be restored over time until a deadline, using multiple vehicles. The objective is to maximize the total weighted earliness of all paths' restoration completion times. They do not consider the walks of the vehicles and are not concerned with their lengths. Instead, they impose that at most K links can be restored at a time.

Sahin et al. [23] study the problem of providing emergency relief supplies to disaster affected regions by removing the accumulated debris in the shortest time. The aim is to visit a set of critical nodes by travelling along a path that may include blocked edges to be unblocked. The authors consider the single vehicle problem and solve instances with a rather limited number of critical nodes. An exact mathematical formulation and a heuristic algorithm are proposed.Berktaş et al. [9] address another version of the same problem. They minimize the weighted sum of visiting times of the critical nodes, where the weights indicate the priorities of the critical nodes. An exact model and a heuristic algorithm using the shortest path distances between the critical nodes is proposed. Computational tests are analyzed on the same data sets with the Sahin et al. [23] study. *KPC-ARCP* differs from the problems defined in these two studies as they consider only a single vehicle and hence there is no need for synchronization. Moreover, while in *KPC-ARCP* a set of blocked edges should be opened to gain prizes, in Sahin et al. [23] and Berktaş et al. [9], the vehicle might not open any of the blocked roads.

Celik et al. [11] introduced a multi-period *Stochastic Debris Clearance Problem* (SDCP) with limited information on the required clearing times on the blocked edges. The information is updated in each period. The objective of SDCP is to maximize the total satisfied demand. The main decision is to determine a sequence of roads to clear in each period. They develop a Markov Decision Process model and a heuristic policy for larger instances. The problem in Celik et al. [11] does not address routing of the vehicles explicitly nor aims to reconnect a disconnected network.

The *Prize Collecting Arc Routing for Connectivity Problem* (PC-ARCP), which is the single vehicle case of *KPC-ARCP*, was first introduced in Kasaei and Salman [20]. In that study an MIP formulation which extends those offered by Asaly and Salman [6] is given and a heuristic algorithm is proposed. Note that the problems both in Asaly and Salman [6] and Akbari and Salman [1] have the makespan objective. Since *PC-ARCP* deals with the single vehicle case, naturally timing conflicts do not occur. In the current study, we develop an exact model, *E-MIP*, to *KPC-ARCP*. However, without relaxation of timing parameters, the model can be solved only for small-sized instances of *KPC-ARCP* in reasonable time. Since the timing conditions complicates the formulation significantly, we relax these conditions and develop a relaxed MIP (*R-MIP*) to derive an upper bound on *KPC-ARCP*. Yet, even *R-MIP* can not be solved in computationally reasonable time, which is in our case one hour. Hence, we develop a Lagrangian Relaxation procedure to derive an upper bound on *R-MIP* and a heuristic algorithm to derive a lower bound on *R-MIP* by means of feasible solutions to *R-MIP*.

We propose a matheuristic algorithm to obtain feasible solutions to *KPC-ARCP* that solves single vehicle models sequentially. The corresponding lower bound together with the upper bound obtained by decomposing *R-MIP* into single vehicle problems using the Lagrangian Relaxation procedure yields an optimality gap to *KPC-ARCP*. In this article we use the idea of decomposing the multi-vehicle problem into single vehicle problems and solving the single vehicle problems under problem-specific conditions to obtain both feasible solutions and upper bounds. This approach can be implemented in other multi-vehicle problems as well.

To test the performance of the proposed methods, we used three different networks from the Istanbul road network using ArcGIS and Google Earth as in Akbari and Salman [1] and in addition we generated a random Euclidean data set. We generated multiple instances with different sets of blocked edges for each network. Our results given in Section 6 show that we can obtain small optimality gaps in short computation times.

3. Problem definition

The road network is represented by an undirected connected graph $G = (V, E)$. A positive cost (time) c_{ij} is associated with traversing each edge, $e = (i, j) \in E$. We assume that traversing an edge in either direction has the same cost. After a natural disaster damages the road network, a set of edges, B, will be blocked. The blockage of edges in B makes the road network disconnected. In other words, $G_B = (V, E \setminus B)$ is a disconnected graph. Traversing a blocked edge is not possible unless it is opened by one of the vehicles. The opening procedure is associated with a cost (time) b_{ij} for each edge $e = (i, j) \in B$. This time incurs only for the first time that the corresponding blocked edge is traversed. We assume that the edge opening times will be estimated by collecting post-disaster information on road conditions. This information can be provided by various tools such as satellite images, Volunteered Geographic Information (VGI) systems or drones. Considering that roads can be used in both directions in the disaster response phase, we assume that the opening cost of an edge is the same in either direction and when an edge is opened, it can be traversed in both directions.

Edges that have not been blocked form the graph G_B having Q connected components. Let $\tilde{Q}$ be the index set of all connected components, $\tilde{Q} = \{1, \ldots, Q\}$ and C_q, $q \in \tilde{Q}$ be the qth connected component with node set V_q and edge set E_q. Furthermore, without loss of generality, we assume that the depot node denoted by D is in the first connected component ($D \in V_1$). K vehicles that are initially positioned in D are dispatched to open a subset of the blocked edges. It is possible for a vehicle to stay put in D. The objective is to maximize the total prize gained from connecting the components within a time limit, $T_{\max}$. The prize of the qth component, P_q, $q \in \tilde{Q}$ is gained when the component C_q is connected to the first component. The solution to *KPC-ARCP* constructs the

routes of the vehicles, starting from the depot node D and ending in a dummy sink node indexed as $n+1$, where n is the number of nodes. A vehicle that goes directly to the sink node from D is meant not to be dispatched.

We assume that if a vehicle arrives at a node incident to a blocked edge while another vehicle is opening it, then it has to wait until the edge is unblocked. Hence, if the walk of two vehicles include the same blocked edge, the one that arrives first to the edge will unblock it.

Let us represent the walk of vehicle k by W_k. The total time of W_k consists of: (1) time of traversing the edges, $C(W_k)$, (2) time of road clearance, $B(W_k)$, and (3) waiting time, $WT(W_k)$. The total walk time for vehicle k is calculated as:

$$T(W_k) = C(W_k) + B(W_k) + WT(W_k) \tag{1}$$

The objective is to maximize the total prize collected while making sure that $T(W_k) \le T_{\max}$ for all $k \in \{1, 2, \ldots, K\}$, where $T_{\max}$ is the time limit in which the routes should be completed.

Definition 1 (Opener)**.** Let T_{ek} be the arrival time of vehicle k to edge $e = (i, j)$ from either node i or j. In a feasible solution to *KPC-ARCP*, if both W_k and $W_{k'}$ include an edge $e = (i, j) \in B$, and if $T_{ek} < T_{ek'}$, then vehicle k is the *opener* of edge e. Furthermore, for $k \neq k'$; if $T_{ek} = T_{ek'}$, then the *opener* of edge e is the vehicle with smallest k.

Definition 2 (Waiting Time)**.** Let WT_{ek} be the waiting time of vehicle k at edge $e = (i, j) \in B$. If k is the *opener* of e, then $WT_{ek} = 0$. Otherwise, if k' is the *opener* of e, then: $WT_{ek} = \max(0\ ,\ T_{ek'} + c_{ij} + b_{ij} - T_{ek})$.

Definition 3 (Feasibility)**.** A solution to *KPC-ARCP* represented by W_k and $T(W_k), k = 1, \ldots, K$, is feasible if and only if the walks $W_k, k = 1, \ldots, K$ are *synchronized*, meaning that there is exactly one opener vehicle of each blocked edge and $T(W_k)$ are calculated according to Definitions 1 and 2.

4. Modeling the problem

In this section we first present an MIP formulation to solve *KPC-ARCP* exactly and then show a relaxation of it.

4.1. E-MIP

E-MIP is the exact MIP formulation that gives an optimal solution to *KPC-ARCP*. It prevents timing conflicts among the vehicle routes by calculating the arrival times to blocked edges and the required waiting times.

In our mathematical formulations, we need to distinguish the direction s of the edges in the walks. Therefore, even though our input graph is undirected, we define an arc set A such that for each (i, j) in E, we add arcs in both directions, i.e. (i, j) and (j, i) to A. Furthermore, for each node j in the node set V, we add an arc from j to the sink node ($j \in V : (j, n+1) \in A$). We also define an arc set $\bar{B}$ for the blocked edges such that for each (i, j) in B we add arcs (i, j) and (j, i) to $\bar{B}$. We assume that if a vehicle unblocks an arc from $\bar{B}$, it will open the corresponding edge and the edge can be traversed in both directions.

We utilize the following property in the formulations.

Proposition 1. *There exists an optimal solution such that an edge will not be traversed in the same direction more than once.*

Proof. [12] proved a similar property for the RPP. Now let us assume a feasible solution to *KPC-ARCP* opens a set of blocked edges, R_f. Moreover, each vehicle $k \in \{1, \ldots, K\}$ opens a set of blocked edges, R_f^k (R_f^k can be empty). For each vehicle k, we can construct an RPP problem when the set of edges that vehicle k should open are determined and the set R_f^k becomes the set of required edges in the RPP instance. In these RPP instances, the property holds. Note that RPP finds the shortest closed walk containing the required edges, but in *KPC-ARCP* the requirement is that the walk length should be less than $T_{\max}$. Since this solution is feasible to *KPC-ARCP*, the obtained shortest route from the single vehicle RPP instance should be less than $T_{\max}$ as well. The resulting single vehicle problems of *KPC-ARCP* differ in two other aspects from the corresponding RPP problems. The walks in *KPC-ARCP* are open and the first traversal of a blocked edge incurs the additional cost b_{ij}. However, these differences do not cause the violation of the property. A closed walk can be turned into an open one by removing one of the traversals, hence the property still holds. Secondly, the total unblocking cost (time) of all selected blocked edges, R_f^k, should be added to the length of the walk of vehicle k, which is a constant when the set R_f^k is known. Moreover, this extra cost is calculated while obtaining the feasible solution and adding it to the tour length does not make it larger than $T_{\max}$. Therefore, only the traversal costs c_{ij} will be used in the corresponding single vehicle problems. Thus, the cost structure becomes the same as in RPP. As a result, the property holds for *KPC-ARCP*. □

We define the decision variables and present the constraints next.

Variable s_{ij}^k imposes the necessary waiting for vehicle $k \in \{1, 2, \ldots, K\}$ on arc $(i, j) \in \bar{B}$. The following relation should hold due to Definitions 1 and 2 in Section 3:

$$s_{ij}^k = \begin{cases} T_{(i,j)k} + WT_{(i,j)k}, & \text{if arc } (i, j) \in \bar{B} \text{ is traversed} \\ & \text{by vehicle } k = 1, 2, \ldots, K \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$Max \quad \sum_{q=2}^{Q} P_q y_q \tag{3}$$

$$t_{ij(n+1)}^k \le T_{max}, \forall (i, j) \in A : i, j \neq n+1, \quad k = 1, 2, \ldots, K \tag{4}$$

The objective function maximizes the total prize collected by connecting components to the component with the depot node. Since all walks end in the sink node indexed as $n+1$, there exists an edge,$(i, j) \in A, j \neq n+1$ such that $t_{ij(n+1)}^k > 0$. Then $t_{ij(n+1)}^k$ gives the length of the kth vehicle's walk and since it should be less than the maximum allowable length, $T_{\max}$, Constraint (4) holds.

$$x_{ijl}^k \le \sum_{h:(l,h)\in A} x_{jlh}^k, \quad \forall (i, j) \in A, \quad l : (j, l) \in A, \quad k = 1, 2, \ldots, K, \quad i \neq D \tag{5}$$

$$x_{ijl}^k \le \sum_{h:(h,i)\in A} x_{hij}^k, \quad \forall (i, j) \in A, \quad l : (j, l) \in A, \quad k = 1, 2, \ldots, K, \quad i \neq D \tag{6}$$

$$\sum_{h:(h,i)\in A} x_{hij}^k \le 1, \quad \forall (i, j) \in A, \quad k = 1, 2, \ldots, K \tag{7}$$

$$\sum_{l:(j,l)\in A} x_{ijl}^k \le 1, \quad \forall (i, j) \in A, \quad k = 1, 2, \ldots, K \tag{8}$$

Constraints from (5) to (8) are vehicle flow balance equations. A vehicle should enter an arc first in order to leave it in the next step of its walk. It also can not leave an arc unless it is traversed in the previous step of the walk. Constraints (5) and (6) set these

properties. Each arc is traversed at most once by a particular vehicle, Constraints (7) and (8) enforce this requirement.

$$\sum_{l:(j,l)\in A}\sum_{j:(D,j)\in A} x^k_{Djl} = 1, \quad k = 1,2,\ldots,K \tag{9}$$

$$\sum_{(i,j)\in A, j\neq n+1} x^k_{ij(n+1)} = 1, \quad k = 1,2,\ldots,K \tag{10}$$

$$\sum_{(i,j)\in A} x^k_{(n+1)ij} = 0, \quad k = 1,2,\ldots,K \tag{11}$$

Each vehicle starts its walk from the depot node and ends it in the sink node. The vehicles do not leave the sink node. These requirements are ensured by Constraints (9)–(11).

$$\sum_{l:(j,l)\in A} x^k_{ijl} \geq z^k_{ij}, \quad \forall (i,j)\in \bar{B}, \quad k = 1,2,\ldots,K \tag{12}$$

$$\sum_{l:(j,l)\in A} x^k_{ijl} + \sum_{h:(i,h)\in A} x^k_{jih} \leq 2\left(\sum_{\kappa=1}^{K} z^k_{ij} + \sum_{\kappa=1}^{K} z^k_{ji}\right), \quad \forall (i,j)\in B, \quad k = 1,2,\ldots,K \tag{13}$$

$$\sum_{k=1}^{K}(z^k_{ij} + z^k_{ji}) \leq 1, \quad \forall (i,j)\in B \tag{14}$$

Constraint (12) forces a blocked edge to be opened only if one of the vehicles crosses it. Constraint (13) ensures that traversing a blocked edge is only possible after unblocking it by one of the vehicles. Since when a blocked edge is cleared in one direction it can be used in both directions, Constraint (14) holds.

$$\sum_{j:\in V\cup\{(n+1)\}:(i,j)\in A} (f^k_{ij} - f^k_{ji}) = -v^k_i, k = 1,2,\ldots,K, \quad \forall\, i\in V\cup\{(n+1)\}\setminus D \tag{15}$$

$$\sum_{j\in V\cup\{(n+1)\}} (f^k_{Dj} - f^k_{jD}) = \sum_{i\in V\cup\{(n+1)\}\setminus\{D\}} v^k_i, \quad k = 1,2,\ldots,K \tag{16}$$

$$\sum_{j\in V} f^k_{j(n+1)} = 1, \quad k = 1,2,\ldots,K \tag{17}$$

$$f^k_{ij} \leq (n-1)\sum_{l:(j,l)\in A} x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in A \tag{18}$$

$$f^k_{ij} \geq \sum_{l:(j,l)\in A} x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in A \tag{19}$$

$$\sum_{j:(j,i)\in A}\sum_{l:(i,l)\in A} x^k_{jil} = v^k_i, \quad k = 1,2,\ldots,K, \quad \forall i\in V \tag{20}$$

In order to ensure connectivity of the walks, we define flow variables f^k_{ij} for every vehicle and for each arc $(i, j) \in A$. The net flow out of a depot node is the total number of visits to all nodes except the depot. For other nodes, net flow into a node equals to the number of visits to the corresponding node. Fig. 1 gives an example of the usage of the flow variables and how they enable the route of a vehicle to be connected. In the walk given in Fig. 1, the 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 (15) and (16) ensure that a vehicle leaves one unit of flow each time it visits a node.

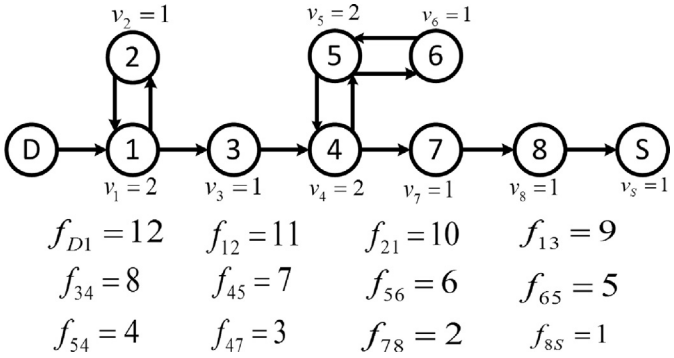

Fig. 1. Demonstration of the Flow Formulation.

Constraint (17) ensures that the walks end in the sink node. Constraint (18) does not allow positive flow on an edge unless it is traversed. Constraint (19) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Constraint (20) counts the number of times vehicle k visits node $i \in V$. There may be different ways of ensuring connectivity. Use of flow variables for connectivity of the walks was also suggested in Benavent et al. [8].

$$t^k_{ijl} \leq M x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad i:(i,j)\in A, \forall l:(j,l)\in A \tag{21}$$

$$s^k_{ij} \leq \sum_{h:(h,i)\in A} t^k_{hij} + M(1 - z^k_{ij}), \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in \bar{B} \tag{22}$$

$$s^k_{ij} \geq \sum_{h:(h,i)\in A} t^k_{hij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in \bar{B} \tag{23}$$

$$s^k_{ij} \geq \sum_{l:(j,l)\in A} t^\kappa_{ijl} - M(1 - z^\kappa_{ij}) - 2M\left(1 - \sum_{h:(h,i)\in A} x^\kappa_{hij}\right), \quad \forall k,\kappa\in\{1,2,\ldots,K\}, \quad \forall (i,j)\in\bar{B}, \quad \text{and} \quad k\neq\kappa \tag{24}$$

$$\sum_{l:(j,l)\in A} t^k_{ijl} \geq s^k_{ij} + b_{ij} z^k_{ij} + \sum_{l:(j,l)\in A} x^k_{ijl} c_{ij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in\bar{B} \tag{25}$$

$$\sum_{l:(j,l)\in A} t^k_{ijl} \geq \sum_{h:(h,i)\in A} t^k_{hij} + \sum_{l:(j,l)\in A} x^k_{ijl} c_{ij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j)\in A\setminus\bar{B} \tag{26}$$

$$y_q \leq \sum_{(i,j)\in\bar{B}: i\notin V_q, j\in V_q}\sum_{k=1}^{K} z^k_{ij}, \quad q = 2,\ldots,Q \tag{27}$$

Constraints (21)–(26) are for setting time limits. Constraint (21) ensures that if $x^k_{ijl} = 0$ then $t_{ijl} = 0$, where M can be set to $\sum_{(i,j)\in A} c_{ij} + \sum_{(i,j)\in B} b_{ij}$. Constraints (22)–(24) are used to calculate s^k_{ij} values. The variable s^k_{ij} shows the earliest time that edge $(i, j) \in B$ is traversable by vehicle k, if vehicle k is planned to traverse it. On the other hand, s^k_{ij} should be forced to zero if vehicle k does not traverse $(i, j) \in B$. Traversing a blocked edge can only start by a vehicle when either the particular vehicle is the *opener* of the edge, or the edge is already unblocked by another vehicle. Due to Constraints (22) and (23), if vehicle k is the *opener* of edge $(i, j) \in B$, the earliest time that (i, j) is traversable by k is the time when

Table 1
Decision variables used in *E-MIP*.

Notation	Type	Definition
x^k_{ijl}, $\forall (i,j),(j,l) \in A, k \in \{1,\dots,K\}$	Binary (Proposition 1)	Equals 1 if vehicle k crosses (j,l) right after (i,j) in its walk; 0, otherwise.
z^k_{ij}, $\forall (i,j) \in \bar{B}$	Binary	Equals 1 if (i,j) is unblocked by vehicle k; 0, otherwise.
f^k_{ij}, $\forall (i,j) \in A$	Continuous	The flow corresponding to vehicle k's route on (i,j) from node i to j.
$v^k_i, i \in V \cup \{n+1\}, k \in \{1,\dots,K\}$	Continuous (Integrality Relaxed)	Number of times vehicle k visits node i.
t^k_{ijl}, $\forall (i,j),(j,l) \in A, k \in \{1,\dots,K\}$	Continuous	The time that vehicle k arrives to node j from i before going to l.
s^k_{ij}, $\forall (i,j) \in \bar{B}$	Continuous	If arc (i,j) is in the walk of vehicle k; it is the earliest time that vehicle k can enter $\{i,j\}$. Otherwise, it is 0 .
y_q, $q \in \{2,\dots,Q\}$	Binary	Equals 1 if component q is connected to the first component; 0, otherwise.

Table 2
Decision variables used in *R-MIP*.

Notation	Type	Definition
x^k_{ij}, $\forall (i,j) \in A, k \in \{1,\dots,K\}$	Binary	Equals 1 if vehicle k traverses (i,j) from node i to j; 0, otherwise.
z^k_{ij}, $\forall (i,j) \in B$	Binary	Equals 1 if (i,j) is unblocked by vehicle k; 0, otherwise.
f^k_{ij}, $\forall (i,j) \in A$	Continuous	The flow corresponding to vehicle k's route on (i,j) from node i to j.
$v^k_i, i \in V \cup \{n+1\}, k \in \{1,\dots,K\}$	Continuous (Integrality Relaxed)	Number of times vehicle k visits node i.
y_q, $q \in \{2,\dots,Q\}$	Binary	Equals 1 if component q is connected to the first component; 0, otherwise.

it arrives to (i,j). On the other hand, if $z^k_{ij} = 0$, then (22) is redundant and by (23) vehicle k can start traversing (i,j) no sooner than its arrival time to node i. If a vehicle is planned to cross a blocked edge without opening it, it is only possible if the edge is unblocked earlier by another vehicle. In addition, $s^k_{ij} = 0$ should be satisfied, if vehicle k is not planned to cross $(i,j) \in \bar{B}$. Constraints (25) and (26) calculate the arrival times to an edge (i,j) for $(i,j) \in \bar{B}$ and $(i,j) \in A \setminus \bar{B}$, respectively. Constraint (27) ensures that y_q is set to 1 only if component q is connected to other components that are connected to the depot node. Since all the vehicles start their walk from the depot node, which is located in the first component, if they enter an isolated component, the component will be connected to the depot node. In order to enter an isolated component, a blocked edge with only one node in the component (i.e. a cutset edge) should be cleared.

The restrictions on the variables were given in Table 1. Note that this formulation has $O(n^3K)$ decision variables.

4.2. R-MIP

Although *E-MIP* is an exact formulation to *KPC-ARCP*, since calculating the arrival times to the nodes complicates the model, only a few of very small instances can be solved in an hour, as we explain at the end of Section 6.5. Therefore, we first relax the timing related components in *E-MIP*. We call this relaxed formulation the Relaxation Mixed Integer Program, *R-MIP*. We can change the four-indexed x variables of *E-MIP* to three-indexed variables in *R-MIP* since we do not need to find the waiting times and node arrival times. In the following, first we define the decision variables in Table 2 and then present the objective and the constraints of *R-MIP*.

$$Max \quad \sum_{q=1}^{Q} P_q y_q \tag{28}$$

subject to

$$\sum_{(i,j)\in A} c_{ij} x^k_{ij} + \sum_{(i,j)\in B} b_{ij} z^k_{ij} \le T_{max}, \quad k = 1,2,\dots,K \tag{29}$$

$$\sum_{j:(D,j)\in A} (x^k_{Dj} - x^k_{jD}) = 1, \quad k = 1,\dots,K \tag{30}$$

$$\sum_{j:(i,j)\in A, j\neq n+1} (x^k_{ij} - x^k_{ji}) = 0, \quad k = 1,2,\dots,K, \quad \forall i \in V\setminus D \tag{31}$$

$$\sum_{j\in V} x^k_{j(n+1)} = 1, \quad k = 1,2,\dots,K \tag{32}$$

$$x^k_{ij} \ge z^k_{ij}, \quad k = 1,2,\dots,K, \quad \forall (i,j) \in B \tag{33}$$

$$\sum_{k=1}^{K} (x^k_{ij} + x^k_{ji}) \le 2K \sum_{k=1}^{K} (z^k_{ij} + z^k_{ji}), \quad \forall (i,j) \in B \tag{34}$$

$$\sum_{j:(j,i)\in A} x^k_{ji} = v^k_i, \quad k = 1,2,\dots,K, \quad \forall i \in V \cup \{(n+1)\} \tag{35}$$

$$f^k_{ij} \le (n-1) x^k_{ij}, \quad k = 1,2,\dots,K, \quad \forall (i,j) \in A \tag{36}$$

$$f^k_{ij} \ge x^k_{ij}, \quad k = 1,2,\dots,K, \quad \forall (i,j) \in A \tag{37}$$

Constraints (15), (16), (17), (27)

In *R-MIP*, the length of the kth vehicle's walk is calculated as $\sum_{(i,j)\in A} c_{ij} x^k_{ij} + \sum_{(i,j)\in B} b_{ij} z^k_{ij}$. We know that this value should be at most $T_{\max}$. Constraint (29) sets this relation for each vehicle $k \in \{1,\dots,K\}$. Constraints from (30) to (32) are vehicle flow balance equations. Constraint (30) ensures that each vehicle leaves the depot. Constraint (31) prevents the vehicles to stop in any intermediate node. Constraint (32) forces each walk to end in the sink node. By Constraint (33), if a blocked edge is opened, it must be traversed. Constraint (34) prevents traversing a blocked edge if it is not unblocked by any of the vehicles. We count the number of times a node is visited by a vehicle by means of Constraint (35). In order to ensure the connectivity of the walks, we used flow constraints in *E-MIP*. We use the same constraints with minor modifications for *R-MIP*. We add Constraint (27) to calculate the values of the y_q variables.

The restrictions on the decision variables are given in Table 2.

5. Solving the problem

In this section we present our approaches for obtaining upper and lower bounds on *KPC-ARCP*

5.1. Lagrangian Relaxation method to solve R-MIP

In the following we apply the Lagrangian Relaxation method to be able to solve larger instances of *R-MIP* in shorter time. While this method gives an upper bound to *R-MIP* and hence to *E-MIP*, we also obtain a lower bound to *R-MIP* by a heuristic algorithm. As a result. we can solve *R-MIP* within a proven optimality gap.

In the Lagrangian Relaxation method, a set of constraints which causes the most difficulties in the model is selected to be relaxed. In this method, a penalty cost is added to the objective function for violations of the relaxed constraints using Lagrange multipliers.

Considering that in the formulation of *R-MIP* only Constraints (27) and (34) relate the walks of the vehicles to each other, we relax these constraints. The Lagrange multipliers for constraints (27) and (34) are shown with $\alpha_q \geq 0$, $q \in Q$ and $\mu_{ij} \geq 0$, $(i,j) \in \bar{B}$, respectively. Let us show the Lagrangian model with $L(\mu, \alpha)$. The objective function of $L(\mu, \alpha)$ is given in (38).

$$Max \quad \sum_{q=1}^{Q} P_q y_q - \sum_{(i,j)\in\bar{B}} \mu_{ij} \sum_{k=1}^{K} (x_{ij}^k + x_{ji}^k - 2K(z_{ij}^k + z_{ji}^k)) - \sum_{q=2}^{Q} \alpha_q \left(y_q - \sum_{(i,j)\in\bar{B}: i\notin V_q, j\in V_q} \sum_{k=1}^{K} z_{ij}^k \right) \tag{38}$$

The decision variables and the constraints of $L(\mu, \alpha)$ are those of the given *R-MIP* model without Constraints (27) and (34). $L(\mu, \alpha)$ decomposes into K single vehicle problems. The constraints of the decomposed problems are similar to those of $L(\mu, \alpha)$ but without the vehicle index k.

The next step in the Lagrangian Relaxation algorithm is to choose the initial value of the Lagrange multipliers and to determine their search procedure. We use the subgradient optimization technique to update the Lagrange multipliers in each step. In the initial step of the Lagrangian Relaxation method, we solve the linear programming relaxation of the *R-MIP* model. We set the Lagrange multipliers, μ_{ij}^0, $(i,j) \in \bar{B}$ and α_q^0, $q \in Q$ of the initial step equal to their corresponding dual values derived from Constraints (27) and (34). Noting that s denotes the step number and we need to keep the Lagrange multipliers positive to impose the necessary penalties, (39) and (40) define the updating of the Lagrange multipliers in each step.

$$\mu_{ij}^{s+1} = max\left\{0, \mu_{ij}^s + \theta^s \left(\sum_{k=1}^{K} (x_{ij}^{k\,(s)} + x_{ji}^{k\,(s)} - 2K(z_{ij}^{k\,(s)} + z_{ji}^{k\,(s)})) \right)\right\}, \quad \forall (i,j) \in \bar{B} \tag{39}$$

$$\alpha_q^{s+1} = max\left\{0, \alpha_q^s + \theta^s \left(y_q^{(s)} - \sum_{(i,j)\in\bar{B}: i\notin V_q, j\in V_q} \sum_{k=1}^{K} z_{ij}^{k\,(s)} \right)\right\}, \forall q \in Q \tag{40}$$

In (39) and (40), θ^s is the *step length* in the $(s+1)$th step of the subgradient optimization procedure. The value of θ^s can be calculated as given in (41).

$$\theta^s = \frac{LB_{R-MIP} - L^*(\mu_{ij}{}^s, \alpha_q{}^s)}{||Ax^s - b||^2} \tag{41}$$

We will explain each term in (41) next. LB_{R-MIP} is a lower bound for *R-MIP*. We derive this lower bound using Algorithm 1 presented in the following. In Algorithm 1, *PC-ARCP-I*(P^k, b_{ij}) denotes the solution of the single vehicle case of *KPC-ARCP* where only vehicle k is used. Note that in the single vehicle case, timing conflicts do not occur and consequently it is noticeably easier and faster to solve. It is worth mentioning that it is possible to obtain the set of cleared roads, E_k, the set of connected components by each vehicle k, O_k, and the optimal route of each vehicle, R_k, from the optimal solution of each single vehicle instance of *KPC-ARCP* defined above.

Algorithm 1 LB_{R-MIP} Algorithm.

Input:

b_{ij}, $(i,j) \in B$ ▷ Original costs of unblocking blocked edges.
P_q^1, $q \in \tilde{Q}$ ▷ Original set of prizes, obtained by connecting each component $q \in \tilde{Q}$.
K ▷ Number of vehicles

```
k ← 1
while k ≤ K do
    Solve PC-ARCP-I(P^k, b_ij)   ▷ PC-ARCP-I(P^k, b_ij) is the single vehicle instance of KPC-ARCP using prize set P^k and unblocking costs b_ij
    Extract E_k, O_k and R_k from the solution of PC-ARCP-I(P^k, b_ij).
        ▷ E_k is the set of unblocked blocked edges by vehicle k
        ▷ O_k is the set of components that are connected by vehicle k
        ▷ R_k is the optimal route for vehicle k
    for (i, j) ∈ E_k do
        b_ij ← 0
    end for
    for q ∈ O_k do
        P_q^{k+1} ← 0
    end for
    k ← k+1
end while
return R_k, k = 1, ..., K
```

In (41), $L^*(\mu_{ij}{}^s, \alpha_q{}^s)$ is the optimal objective value of $L(\mu, \alpha)$ in the sth step of the subgradient optimization procedure. Furthermore, $||Ax^s - b||$ in (41) denotes the Euclidean norm of the vector $Ax^s - b$ defined in (42). Note that $(i,j) \in \bar{B}$ and $q \in Q$, and

$$Ax^s - b = \begin{pmatrix} \sum_{k=1}^{K} (x_{ij}^{k\,s} + x_{ji}^{k\,s} - 2K(z_{ij}^{k\,s} + z_{ji}^{k\,s})) \\ y_q{}^s - \sum_{(i,j)\in\bar{B}: i\notin V_q, j\in V_q} \sum_{k=1}^{K} z_{ij}^{k\,s} \end{pmatrix}. \tag{42}$$

Given the initial values of the Lagrange multipliers and the updating procedure for these multipliers, we implement the Lagrangian Relaxation method. First we solve $L(\mu, \alpha)$ with the initial multiplier values. Then in each step first we check if the current solution is optimal. If not, we update the Lagrange multipliers and then solve $L(\mu, \alpha)$ until a time limit is reached. For our tested instances we set the time limit to be one hour. In order to check the optimality of a solution $(\hat{x}, \hat{z}, \hat{y})$ and Lagrange multipliers $(\hat{\mu}_{ij}, \hat{\alpha}_q)$, we check the feasibility and the complementary slackness conditions given from (43) to (46). If a solution satisfies these conditions, then the Lagrangian Relaxation algorithm terminates. Otherwise, it terminates when the time limit of one hour is reached.

$$\sum_{k=1}^{K} (\hat{x}_{ij}^k + \hat{x}_{ji}^k - 2K(\hat{z}_{ij}^k + \hat{z}_{ji}^k)) \leq 0, \forall (i,j) \in \bar{B} \tag{43}$$

$$\hat{\mu}_{ij} \left(\sum_{k=1}^{K} (\hat{x}_{ij}^k + \hat{x}_{ji}^k - 2K(\hat{z}_{ij}^k + \hat{z}_{ji}^k)) \right) = 0, \forall (i,j) \in B \tag{44}$$

$$\hat{y}_q - \sum_{(i,j)\in\bar{B}: i\notin V_q, j\in V_q} \sum_{k=1}^{K} \hat{z}_{ij}^k \leq 0, \forall q \in Q \tag{45}$$

$$\hat{\alpha}_q \left(\hat{y}_q - \sum_{(i,j)\in \bar{B}: i\notin V_q, j\in V_q} \sum_{k=1}^{K} \hat{z}_{ij}^k \right) = 0, \forall q \in Q \quad (46)$$

In Algorithm 1, we solve the single vehicle problem repeatedly and update the network from the results obtained for each vehicle. These updates are related to unblocking a blocked edge and connecting a component to the depot. Since we ignore timing conditions in *R-MIP*, the walks of the K vehicles collectively give a feasible solution to *R-MIP*. This means that LB_{R-MIP} is a lower bound on the optimal objective value of *R-MIP*.

Proposition 2. *In the solution of Algorithm 1, if a vehicle contributes zero units of prize, then the solution of Algorithm 1 gives an upper bound to the optimal objective values of both* R-MIP *and* KPC-ARCP *at the same time. Hence the solution of Algorithm 1 is an optimal solution to* R-MIP.

Proof. When a vehicle v contributes zero units of prize, it means that vehicle v does not reach any of the components that are still separated. Thus, all the components that can be reconnected in $T_{\max}$ are already connected by the routes of vehicles from 1 to $v-1$. Let us furthermore assume that vehicle $\hat{v} \in \{1, \ldots, v-1\}$ connects the components in the set $C_{\hat{v}}$ in the solution output by Algorithm 1; where $C_{\hat{v}} = \{c \in \{2, \ldots, Q\} : \sum_{(i,j)\in \bar{B}: j\in V_c, i\notin V_c} z_{ij}^{\bar{v}} \geq 1\}$. Then, $PC = \bigcup_{\hat{v}\in\{1,\ldots,v-1\}} C_{\hat{v}}$ is the set of all components that are reachable in $T_{\max}$. Since none of the components in the set $Q\backslash$PC can be connected in $T_{\max}$, reconnecting all the components in PC gives an upper bound on the optimal objective value of *KPC-ARCP* as well as the optimal objective value of *R-MIP*. □

5.2. A Matheuristic For KPC-ARCP

In this section, we propose an easily implementable matheuristic algorithm to obtain near-optimal feasible solutions to large *KPC-ARCP* instances. In this algorithm, we solve a single vehicle problem repeatedly at each step and update the prizes of the visited components to zero. With this procedure, we prevent multiple vehicles to connect the same components. We call this algorithm the *Successive Single Vehicle* algorithm and denote it as *SSV*. Note that since we use a single vehicle in each step, a timing conflict does not happen and the routes of the vehicles altogether give a feasible solution to *KPC-ARCP*. The reason will be explained later on when we compare Algorithms 1 and 2.

Algorithm 2 Successive Single Vehicle Algorithm (*SSV*).

Input:

$P_q^1, q \in \tilde{Q}$ ▷ Original set of prizes, obtained by connecting each component $q \in \tilde{Q}$.

K ▷ Number of vehicles

1: $k \leftarrow 1$
2: **while** $k \leq K$ **do**
3: Solve *PC-ARCP-II*(P^k) ▷ *PC-ARCP-II*(P^k) is the single vehicle instance of *KPC-ARCP* with prize set P^k.
4: Extract O_k and R_k from the solution of *PC-ARCP-II*(P^k).
▷ O_k is the set of components that are corrected by vehicle K
▷ R_k is the optimal route for vehicle k
5: **for** $q \in O_k$ **do**
6: $P_q^{k+1} \leftarrow 0$
7: **end for**
8: k ← k+1
9: **end while**
10: **return** $R_k, k = 1, \ldots, K$

In Algorithm 2, namely *SSV*, *PC-ARCP-II*(P^k) denotes the solution of the kth single vehicle problem with the prize set P^k. Note that we can obtain both O_k and R_k from *PC-ARCP-II*(P^k), where O_k is the set of connected components and R_k is the optimal route for vehicle k obtained by solving *PC-ARCP-II*(P^k).

Proposition 3. *The output of Algorithm 2 gives a set of feasible walks to* KPC-ARCP.

Proof. The obtained walks from Algorithm 2 gives a set of edges and blocked edges to traverse for each vehicle. In these walks, traversing a blocked edge incurs the extra cost of unblocking it in all the cases. However, a blocked edge should be opened only with one vehicle. Let us assume that multiple vehicles traverse a common blocked edge, $(i, j) \in$ B. We can choose the opener of (i, j) using Definition 1 which implies that the vehicle that arrives to (i, j) earlier than other vehicles is the opener of (i, j). Let us assume vehicle k is set to be the opener of (i, j) and it arrives to (i, j) at $T_{(i,j)k}$. Hence the unblocking process of (i, j) will be finished at $T_{(i,j)k} + b_{ij} + c_{ij}$. It implies that other vehicles can finish traversing (i, j) at least at $T_{(i,j)k} + b_{ij} + c_{ij}$. Since the arrival time of other vehicles to (i, j) is larger than $T_{(i,j)k}$ and they should spend $c_{ij} + b_{ij}$ for traversing (i, j), they can not finish traversing (i, j) sooner than $T_{(i,j)k} + b_{ij} + c_{ij}$. which is the time that unblocking process of (i, j) is finished. Thus, timing conflicts does not occur and the walks are synchronized and feasible to *KPC-ARCP*. □

The difference between Algorithms 1 and 2 is that, Algorithm 2 gives a lower bound to *KPC-ARCP* but Algorithm 1 gives a lower bound to *R-MIP*. This is because in Algorithm 2 we do not update the opened blocked edges in each step. Hence, none of the vehicles traverses an edge unless its unblocking process is finished or the same vehicle is responsible for unblocking it. As a result, the collection of the routes obtained from Algorithm 2 gives a feasible solution to *KPC-ARCP* and the feasible solution to *KPC-ARCP* gives a lower bound on it. We demonstrate Algorithms 1 and 2 by an example in the Appendix.

Proposition 4. *If a vehicle contributes zero units of prize in the solution output by Algorithm 2, then the solution of Algorithm 2 is an optimal solution to* KPC-ARCP.

Proof. As preprocessing, a component in G_B such that the shortest length of a path from D to each of its nodes is more than $T_{\max}$ can be removed. Hence, if a vehicle v cannot connect any of the components in Algorithm 2, it means that it does not leave the depot. A vehicle stays at the depot only when all the components that are reachable in $T_{\max}$ have been connected by the vehicles before v. It means that a collection of the routes of vehicles from 1 to $v-1$ connects all the components that can be connected within $T_{\max}$ and is also feasible to *KPC-ARCP* by Proposition 3. Hence, these routes give an optimal solution to *KPC-ARCP* according to Definition 3. □

We utilize Propositions 2 and 4 in our computational results.

6. Computational study

In order to test the performance of the matheuristic and derive optimality gaps for it we conducted a computational study.

6.1. Data generation and settings

We generated two types of data: (1) based on the Istanbul road network that was generated using ArcGIS and Google Earth, and (2) randomly generated with Euclidean distances. Three different data sets are generated from the road network of Istanbul city. The first data set is based on a simplified Istanbul network

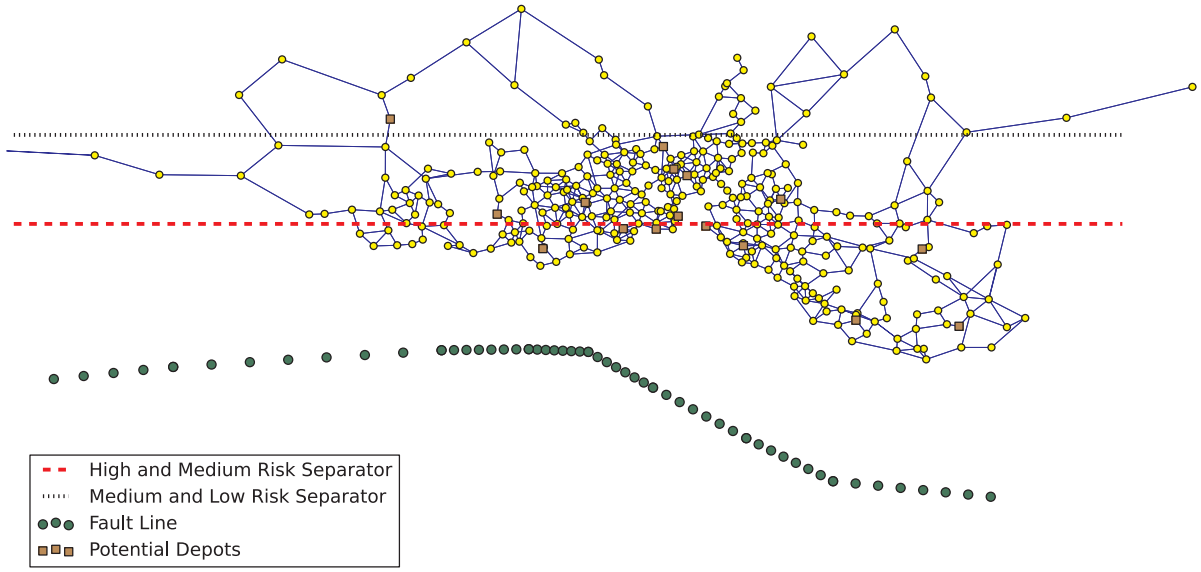

Fig. 2. Detailed Istanbul network and the Fault line.

with 74 nodes and 179 edges, considering the main highways and their connections to various demand centers. The second data set is based on a more detailed Istanbul road network with 349 nodes and 689 edges (Fig. 2). Finally, the third data set is based on a more detailed road network of the Southwestern region of Istanbul city with 250 nodes and 539 edges. These networks were first generated in Akbari and Salman [1]. Although we use the same initial networks, but here our instances have different sets of blocked edges and hence different sets of components. Both the number of blocked edges and the components are higher in our data sets. We also generated 40 random networks with 200, 250, 300 and 350 nodes to see the impact of many different topologies and network sizes on the performance of the heuristic algorithm. The average degrees of the nodes in the Istanbul networks after the edge blockages are 3.95, 4.31 and 4.84 for the Detailed, Southwest and Simplified networks respectively. The average degrees of the nodes for random instances with 200 to 350 nodes are 4.22, 4.62, 5.10 and 5.17 on the average over the 10 tested instances. We observed that the Euclidean networks become disconnected by blocking fewer numbers of edges. Another goal of having the random data set is to observe how the performance of the heuristic changes as the number of blocked edges changes. We tested all data sets with 1 to 5 vehicles keeping all the other inputs the same.

Fig. 2 shows the detailed Istanbul network and the active North Anatolian Fault line that is under the Marmara Sea that lies to the south of Istanbul. The edges are categorized into three groups due to their proximity to the epicenter of the earthquake according to the scenarios predicted in the JICA report [19], as high, medium and low-risk edges. The probability that an edge from the low-risk class is blocked after an earthquake is set to 0.1, while this probability is set to 0.2 and 0.3 for medium and high-risk edges, respectively. Edges located above the higher horizontal line in Fig. 2 lie in the group of low-risk edges; those located between the two horizontal lines are in the group of medium-risk edges; and those under the lower horizontal line are high-risk edges. If nodes of an edge are located in two different groups, we placed that particular edge in the more risky group.

In *KPC-ARCP*, the vehicles should be dispatched from the depot node. For the simplified and detailed Istanbul road networks, we selected 16 potential depot nodes as shown in Fig. 2. Among these 16 nodes, 5 of them are located in the Southwestern part of the Istanbul road network. In different instances, the depot of all the vehicles is chosen randomly among the potential depot nodes and remain the same for all tests with 1 to 5 vehicles to observe the impact of having more vehicles on the performance of the methods.

For the first three sets of data, the traversal cost c_{ij} on edge (i, j) is equal to the time it takes for a vehicle to go from node i to node j. We assumed that the speed of vehicles is 50 km/h on the average. The distance between nodes i and j is calculated using ArcGIS. The opening cost, b_{ij}, on edge (i, j) is a random number generated according to: $b_{ij} = c_{ij} \times X$, where X has uniform distribution between 10 and 30. Note that, if b_{ij} is too large (e.g. larger than a threshold value), edge (i, j) can be removed. If this causes the initial graph to be disconnected, we can remove the components disconnected from the depot nodes and solve the problem on the remaining connected graph.

The prizes for each component are set to be equal to the number of nodes in that component. Finally, the value of $T_{\max}$ is determined by solving the single vehicle problem of minimizing the time to connect all the components according to the MIP model provided in Kasaei and Salman [20]. If we show this tour length by TL, then we set $T_{\max} = 0.8 \times \frac{TL}{3}$. We chose this value to create a trade off since having $T_{\max}$ too small might prevent connecting any of the components or having it too large results in connecting all the components.

All the scenarios were tested on an Intel Xeon E5-2643 CPU @ 3.30GHz computer with 32 GB RAM. We used version 6.5.1 of Gurobi to solve the MIP problems.

6.2. Results of the detailed Istanbul network instances

Table 3 gives the results of the tested instances for the detailed Istanbul road network. For each tested instance, Q is the number of connected components, $|B|$ is the number of blocked edges and K is the number of vehicles. The columns under (1), titled *R-MIP Exact*, are related to the solution of *R-MIP* by the Gurobi solver. In these two columns, if an instance is solved optimally within the one hour time limit, the run time is given; otherwise, the run time is 3600 s and the optimality gap obtained within one hour is reported. Algorithm 1 provides a feasible solution and a lower bound to *R-MIP*, namely LB_{R-MIP}. For each particular instance the run time of Algorithm 1 is given in column (2). The Lagrangian Relaxation method is used to obtain an upper bound on *R-MIP* instances and

Table 3
Results for detailed Istanbul network instances.

run number	Q	|B|	K	R-MIP (Upper bound on KPC-ARCP)					Lower bound on KPC-ARCP		P2	P4
				(1)		(2)	(3)	(4)	(5)	(6)		
				R-MIP		Lower bound	Upper bound	*R-MIP*	*SSV*	KPC-ARCP		
				Exact		LB_{R-MIP}	Lagrangian Relaxation	Heuristic	Time (s)	Gap (%)		
				Gap (%)	Time (s)	Time (s)	Time (s)	Gap (%)				
1	9	153	1	0	9.24	17.95	17.95	0	17.78	0	*	
			2	10.00	3600	24.53	173.28	0	23.93	0		
			3	9.09	3600	29.54	178.29	0	26.72	0		
			4	9.09	3600	82.83	82.83	0	48.3	0	*	
			5	9.09	3600	138.19	138.19	0	71.28	0	*	
2	9	139	1	0	4.18	6.03	6.03	0	6.12	0	*	
			2	0	437.28	20.38	22.1	0	15.98	0		
			3	45.45	3600	1663.34	1663	0	74.67	0	*	
			4	41.66	3600	3300.48	3300.48	0	134.86	0	*	
			5	41.66	3600	4939.27	4939.27	0	193.42	0	*	
3	10	150	1	0	0.88	0.42	0.42	0	0.43	0	*	
			2	0	3524.27	26.75	39.62	0.96	0.86	0		*
			3	0	3545.4	92.6	112.7	0.32	1.29	0		*
			4	7.88	3600	125.12	148.09	0	1.71	0		*
			5	8.20	3600	186.51	186.51	0	2.12	0		*
4	10	161	1	0	4.98	1.9	1.9	0	1.91	0	*	
			2	0	649.45	4.85	16.35	2.17	4.89	0		
			3	2.12	3600	20.59	39.67	2.13	9.67	2.13		
			4	4.16	3600	21.89	48.91	0	11.04	0		
			5	4.16	3600	65.23	65.23	0	29.97	0	*	
5	9	164	1	0	3.76	1.62	1.62	0	1.64	0	*	
			2	0	137.18	3.29	4.7	25.00	4.91	0		
			3	20.00	3600	6.09	8.45	0	8.27	0		
			4	30.00	3600	48.66	48.66	0	22.03	0	*	
			5	40.00	3600	90.35	90.35	0	34.98	0	*	
6	8	164	1	0	3.09	4.48	4.48	0	4.53	0	*	
			2	0	435.27	5.39	6.86	0	5.41	0		
			3	0	72.34	6.15	9.25	0	6.25	0		
			4	0	158.14	7.66	7.66	0	7.22	0	*	
			5	0	537.23	8.12	8.12	0	7.91	0	*	
7	8	157	1	0	6.98	6.38	6.38	0	6.4	0	*	
			2	0	3600	21.66	23.67	0	11.88	0		
			3	1.40	3600	92.77	92.77	0	20.65	0	*	
			4	1.40	3600	161.93	161.93	0	28.83	0	*	
			5	2.81	3600	229.54	229.54	0	37.32	0	*	
8	13	161	1	0	210.12	180.47	180.47	0	179.24	0	*	
			2	8.33	3600	188.46	190.01	0	192.76	0		
			3	15.38	3600	199.93	202.47	0	189.91	0		
			4	52.38	3600	208.06	378.72	0	205.61	0		
			5	113.33	3600	225.81	230.5	0	218.91	0		
9	13	153	1	0	12.18	11.14	11.14	0	10.81	0	*	
			2	5.26	3600	15.94	17.44	5.26	17.54	5.26		
			3	15.00	3600	44.01	61.46	0	22.73	0		
			4	23.80	3600	618.83	618.83	0	385.73	0	*	
			5	31.81	3600	1201.15	1201.15	0	734.99	0	*	
10	13	166	1	0	45.32	43.34	43.34	0	43.34	0	*	
			2	0	3600	51.85	53.79	0	51.72	0		
			3	14.08	3600	562.93	566.01	0	107.02	0		
			4	14.08	3600	904.07	908.15	0	113.65	0		
			5	0	3600	1190.56	1194.66	0	114.74	0		

the run time of the procedure is given in column (3). Since solving Algorithm 1 is part of the Lagrangian relaxation procedure, the time of column (2) is included in the time of column (3). Column (4) gives the optimality gap for each instance of *R-MIP* calculated by the obtained lower and upper bounds. In column (5), the run time of the *SSV* algorithm (Algorithm 2) which gives a lower bound and a feasible solution to *KPC-ARCP* is given. Column (6) presents the best optimality gap obtained for each individual instance of *KPC-ARCP*. This gap is derived from the lower bound obtained by the *SSV* algorithm and with respect to the upper bound from the Lagrangian Relaxation problem. The last two columns, P2 and P4 show if we used Proposition 2 and Proposition 4, respectively, in the results for each instance or not. For instance, Proposition 2 is used in run number 1 with four vehicles. It means that in the output of Algorithm 1 one of the vehicles remained unused and consequently, this algorithm gives the optimal solution to *R-MIP*. Hence, we did not solve the Lagrangian Relaxation of this run and the run time in column (3) is the same as the one in column (2). An example of using Proposition 4 is in run number 3 with 2 vehicles. It means that since in the solution to Algorithm 2 there exists a vehicle that connects none of the components, the solution to *SSV* gives the optimal solution to *KPC-ARCP*.

In Table 3, as the number of vehicles increases, the *R-MIP* formulation can not be solved exactly within one hour. For instances with four and five vehicles, only one of the instances out of 10 was solved optimally within one hour and for three vehicles only two of the instances were solved in the given time limit.

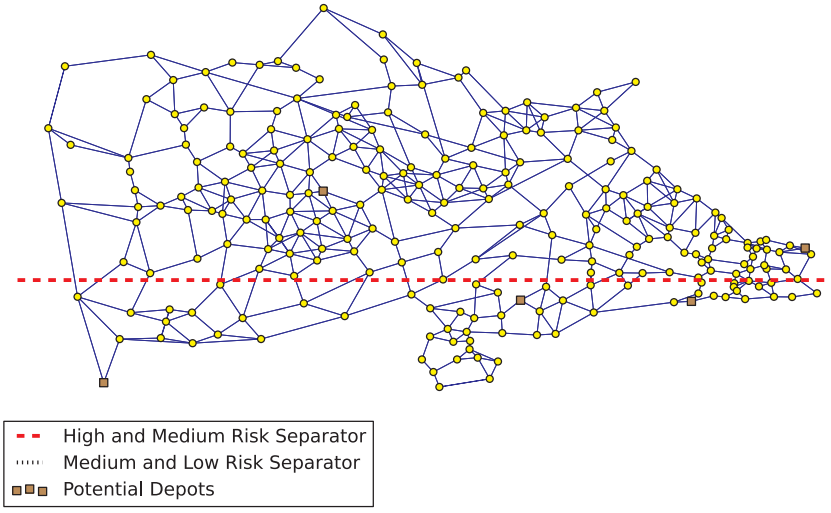

Fig. 3. Southwest Istanbul road network.

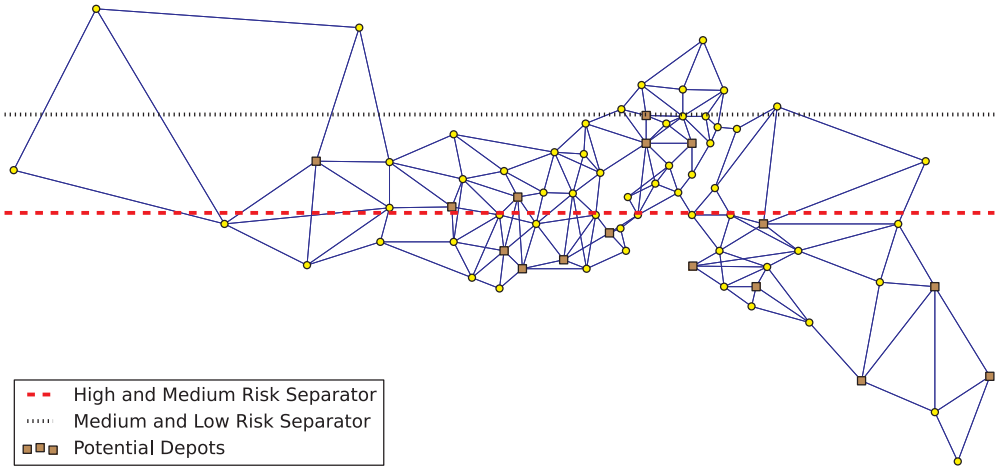

Fig. 4. Simplified Istanbul network.

In run number 2, LB_{R-MIP} was found in 3300.48 s and 4939.27 s with four and five vehicles, respectively. Apart from run number 2, the maximum run time over other instances was 1190.56 s in run number 10 with 5 vehicles. The average run time of LB_{R-MIP} increases from 22.37 s with a single vehicle to 827.47 with five vehicles over all of the 10 runs. The maximum run time of *SSV* algorithm is 734.99 in run number 9 with 5 vehicles. The average run time of *SSV* is 69 s over all instances with 1 to 5 vehicles. Column (6) shows that 48 instances out of the 50 tested instances were solved optimally. Despite the difficulty of solving even *R-MIP* exactly, our solution method managed to solve *KPC-ARCP* optimally in all but two instances. In these two instances, the obtained optimality gap for *R-MIP* is not zero as well.

6.3. Results of the Southwestern Istanbul network instances

Fig. 3 shows the road network of the southwest region of Istanbul, where the two risk categories are divided by a horizontal line. All the edges in this region are located either in the high risk or medium risk zones. Again, we tested 10 instances with 1 to 5 vehicles for each of these 10 instances for this network. From Table 4 we can see that as the number of vehicles and the number of components increase, *R-MIP* becomes more difficult to solve. While all the runs with a single vehicle were solved in 20.53 s on the average, none of the runs with five vehicles could be solved optimally in one hour. With four vehicles, *R-MIP* was solved optimally within the one hour time limit only for one instance out of 10 tested runs. With two and three vehicles, seven and two instances were solved optimally within the time limit, respectively. These results attest to the difficulty of solving *R-MIP* directly by the Gurobi solver.

While *R-MIP* can not be solved for larger instances having more vehicles, Algorithm 1 runs in much shorter time. The largest run time of Algorithm 1 is in run number 5, where the number of components is the largest and 5 vehicles exist. The average Algorithm 1 run time over 10 instances for a single vehicle is 9.30 s and this value increases to 277.86 with five vehicles. We show in column (4) that many of the Algorithm 1 solutions are in fact optimal solutions to *R-MIP* by means of Proposition 2.

The average run time of the Lagrangian Relaxation procedure over all of the instances with 1 to 5 vehicles is 307.06 s. Among all of the instances, in three of them, the Lagrangian Relaxation procedure stopped by reaching the time limit of one hour.

In column (5) we see that the *SSV* algorithm was solved in a very short time for each of the instances. The maximum run time of this algorithm was 219.16 s in run number 8 with 5 vehicles.

Table 4
Result for Southwestern Istanbul road network instances.

run number	Q	|B|	K	R-MIP (Upper bound on KPC-ARCP)					Lower bound on KPC-ARCP		P2	P4
				(1)		(2)	(3)	(4)	(5)	(6)		
				R-MIP		Lower bound	Upper bound	*R-MIP*	*SSV*	KPC-ARCP		
				Exact		LB_{R-MIP}	Lagrangian Relaxation	Heuristic				
				Gap (%)	Time (s)	Time (s)	Time (s)	Gap (%)	Time (s)	Gap (%)		
1	9	175	1	0	4.46	2.55	2.55	0	2.62	0		
			2	0	366.27	4.74	5.80	7.14	5.02	0		
			3	46.66	3600	25.78	27.32	0	10.4	0		
			4	33.33	3600	78.68	78.68	0	23.79	0	*	
			5	13.63	3600	130.68	130.68	0	37.36	0	*	
2	10	165	1	0	0.56	0.34	0.34	0	0.34	0		
			2	0	9.79	2.35	2.35	0	1.82	0	*	
			3	0	461.55	4.13	4.13	0	3.3	0	*	
			4	33.33	3600	6.05	6.05	0	4.77	0	*	
			5	42.85	3600	8.01	8.01	0	6.27	0	*	
3	10	183	1	0	2.13	1.45	1.45	0	1.41	0		
			2	0	15.05	2.3	19.75	0	3.19	0		
			3	0.41	3600	2.91	59.44	0	4.64	0		*
			4	0	470	4.15	88.99	0	6.28	0		*
			5	0.82	3600	5.38	5.38	0	7.64	0	*	*
4	10	170	1	0	1.8	1.1	1.1	0	1.07	0		
			2	0	80.1	1.51	2.49	0	1.52	0		
			3	0	294.59	2.21	9.79	0	2.68	0		
			4	9.09	3600	2.71	5.62	9.09	3.15	9.09		
			5	8.33	3600	3.25	6.99	0	3.57	0		
5	17	182	1	0	0.63	0.42	0.42	0	0.4	0		
			2	0	933.08	52.3	53.61	2.59	0.84	0		*
			3	1.69	3600	119.52	122.28	2.54	1.2	0		*
			4	2.52	3600	134.67	137.25	2.10	2.78	0		*
			5	2.07	3600	1378.75	1382.82	3.75	2.51	0		*
6	10	179	1	0	5.47	3.71	3.71	0	3.8	0		
			2	0	220.77	6.97	8.07	0	7.17	0		
			3	300.00	3600	22.74	22.74	0	20.8	0	*	
			4	325.00	3600	39.36	39.36	0	34.53	0	*	
			5	325.00	3600	55.78	55.78	0	48.84	0	*	
7	11	167	1	0	91.89	25.68	25.68	0	20.54	0		
			2	6.66	3600	24.06	25.2	13.33	25.37	6.66		
			3	18.75	3600	26.23	28.11	25	26.67	6.25		
			4	18.75	3600	27.58	30.66	17.65	28.01	0		
			5	11.76	3600	31.17	34.86	5.26	30.23	0		*
8	13	166	1	0	9.29	8.37	8.37	0	8.52	0		
			2	1.69	3600	74.04	3600	1.27	13.54	1.70		
			3	1.66	3600	158.27	163.2	4.22	20.07	0		
			4	2.48	3600	253.04	514.49	2.53	110	0		*
			5	1.22	3600	278.63	323.5	2.06	219.16	0		*
9	11	169	1	0	86.03	44.78	44.78	0	43.73	0		
			2	9.09	3600	90.02	91.25	30.00	70.55	9.09		
			3	23.07	3600	100.13	140.69	8.33	79.73	8.33		
			4	12.50	3600	127.13	130.4	0	87.61	0		
			5	12.50	3600	657.15	661.46	0	176.68	0		*
10	11	177	1	0	3.05	4.62	5.54	0	4.62	0		
			2	0	28.31	6.17	16.62	0	6.16	0		
			3	3.70	3600	9.36	15.36	7.41	8.6	0		
			4	14.28	3600	25.46	3600	3.57	29.4	3.57		
			5	0.34	3600	229.83	3600	0	64.8	0		

The average *SSV* run time is 8.44 s with a single vehicle and it increases to only 59.71 s with 5 vehicles.

Finally, we are able to show that an optimal solution to *KPC-ARCP* was found in 43 instances out of the 50 tested instances in very short times. The average optimality gap over all of the tested instances with 1 to 5 vehicles is 0.89%.

6.4. Results of the simplified Istanbul network instances

Fig. 4 is the simplified version of the Istanbul road network, where different risk categories are distinguishable with horizontal lines. For this network we tested 10 instances with different number of connected components as seen in Table 5. We examined the performance of the developed methods and algorithms with 1 to 5 vehicles for each of these 10 instances.

In Table 5, we can see that as the number of vehicles increases for the same instance, on the average, the problem becomes more difficult. While the average run time of solving *R-MIP* formulation exactly with the Gurobi solver over all 10 instances with a single vehicle is 1.51 s, the average *R-MIP* run time increases to 1994.30 s over all instances. Although the run time of solving *R-MIP* exactly increases significantly with more vehicles, the run times of Algorithms 1 and 2 remain small in each of the instances and these times are less than 9 s for each individual instance. In the Lagrangian Relaxation procedure, in general, small run times of at most 55 s are obtained. The largest run times are associated with run number 1 due to larger number of connected components. *SSV*

Table 5
Results for simplified Istanbul network instances.

run number	Q	|B|	K	R-MIP (Upper bound on KPC-ARCP)					Lower bound on KPC-ARCP		P2	P4
				(1)		(2)	(3)	(4)	(5)	(6)		
				R-MIP		Lower bound	Upper bound	*R-MIP*	*SSV*	KPC-ARCP		
				Exact		LB_{R-MIP}	Lagrangian Relaxation	Heuristic				
				Gap (%)	Time (s)	Time (s)	Time (s)	Gap (%)	Time (s)	Gap (%)		
1	13	102	1	0	2.23	1.42	1.42	0	1.46	0		
			2	0	268.59	2.51	2.87	0	2.39	0		
			3	10.52	3600	3.05	55.03	10.53	2.96	5.00		
			4	10.00	3600	3.44	7.9	5.00	3.17	0		
			5	4.76	3600	3.7	9.07	0	4.06	0		*
2	8	89	1	0	1.45	0.67	0.8	0	0.7	0		
			2	0	23.01	1.56	8.2	20.00	1.19	0		
			3	0	288.08	2.1	4.96	9.09	2.87	0		*
			4	8.33	3600	2.38	6.19	0	4.51	0		*
			5	8.33	3600	2.86	2.86	0	6.31	0		*
3	7	78	1	0	0.66	1.01	1.01	0	1.04	0		
			2	0	15.48	1.14	2.35	0	1.61	0		
			3	0	10.28	1.43	1.43	0	1.67	0	*	*
			4	0	5.97	1.73	1.73	0	2.1	0	*	
			5	0	36.43	1.96	1.96	0	2.57	0	*	
4	7	84	1	0	0.26	0.2	0.2	0	0.2	0		
			2	0	16.59	0.37	4.91	0	0.42	0		
			3	0	120.01	0.57	2.89	0	0.61	0		
			4	0	15.27	0.99	10.28	0	1.34	0		
			5	0	266.19	1.11	1.11	0	1.32	0	*	
5	8	89	1	0	3.62	2.59	2.59	0	2.52	0		
			2	0	2944.16	5.68	5.93	3.33	3.59	1.75		
			3	0	390.21	6.48	6.87	1.64	5.38	0		
			4	1.61	3600	7.4	7.91	0	6.31	0		*
			5	1.61	3600	8.78	9.41	0	7.11	0	*	*
6	8	79	1	0	1.36	0.68	0.68	0	0.72	0		
			2	0	29.78	1.38	1.63	0	1.22	0		
			3	0	58.18	1.59	1.99	0	2.42	0		*
			4	0	337.54	1.7	1.7	0	3.5	0	*	
			5	0	956.35	1.83	1.83	0	4.73	0	*	
7	9	90	1	0	1.4	0.89	0.89	0	0.9	0		
			2	0	38.44	1.66	1.95	6.67	1.69	6.67		
			3	5.88	3600	2.6	2.97	0	2.77	0		
			4	11.11	3600	3.82	3.82	0	3.52	0	*	
			5	5.26	3600	5.08	5.08	0	4.3	0		
8	8	83	1	0	2.96	1.04	1.04	0	1.04	0		
			2	0	98.28	2.04	2.33	0	1.99	0		
			3	0	112.12	2.11	2.11	0	2.06	0	*	
			4	0	125.31	2.23	2.23	0	2.22	0	*	
			5	0	129.56	2.32	2.32	0	2.36	0	*	
9	9	97	1	0	0.91	0.85	0.85	0	0.83	0		
			2	0	12.34	1.22	2.26	0	1.22	0		
			3	0	112.76	1.32	4.51	0	1.45	0		
			4	0	537.24	1.52	1.79	0	1.68	0		*
			5	0	554.45	1.77	1.77	0	1.98	0		*
10	9	88	1	0	0.28	0.28	0.28	0	0.27	0		
			2	0	121.89	0.43	14.8	0	0.42	0		
			3	0	981.91	0.92	5.47	0	0.59	0		
			4	1.81	3600	1.51	1.51	0	0.78	0	*	
			5	1.81	3600	1.97	1.97	0	0.92	0	*	

gives an optimal solution to *KPC-ARCP* in all but three instances. We utilize Propositions 2 and 4 to verify optimality in many instances.

In Tables 3–5 we can see that the proposed *SSV* algorithm yields an optimal solution to *KPC-ARCP* in most of the instances. The average gap over all tested instances for the detailed road network is 0.15%, for the southwest region it is 0.89%, and for the simplified road network this average gap is 0.14%.

Table 6 shows a summary of the results of Istanbul instances presented in Tables 3–5. While the exact mathematical model for *KPC-ARCP* cannot find a feasible solution to even instances of Simplified Istanbul road network in an hour, the developed heuristic algorithm gives an optimal or near-optimal solution in less than a minute in all of the tested instances.

6.5. Results of random data sets

For instances having 200, 250, 300 and 350 nodes we tested 10 random instances for each and solved each of these instances with 1 to 5 vehicles and a fixed depot node. First, we assigned random coordinates to each node in a 1000 × 1000 plane. The costs, c_{ij}, are taken equal to the Euclidean distances. The extra cost, b_{ij}, is generated according to: $b_{ij} = c_{ij} \cdot X$ where X has uniform distribution between 10 and 30. In each case, an edge (i, j) exists if the distance between nodes i and j is at most a threshold value. For 200 nodes we set the distance threshold to be 85; for 250 nodes, 80; for 300 nodes, 75 and finally for 350 nodes it is set to 70. These random networks are denser than the Istanbul road networks. Since one of the primary assumptions is that the graph $G = (V, E)$ is con-

Table 6
Summary of the results (Istanbul networks).

Category	K	R-MIP (Upper bound on KPC-ARCP)					Lower bound on KPC-ARCP	
		R-MIP		Lower bound	Upper bound	*R-MIP*	*SSV*	*KPC-ARCP*
		Exact		LB_{R-MIP}	Lagrangian Relaxation	Heuristic		
		Avg. Gap (%)	Avg. Time (s)	Avg. Time (s)	Avg. Time (s)	Avg. Gap (%)	Avg. Time (s)	Avg. Gap (%)
Detailed	1	0	30.07	27.37	27.37	0	27.22	0
	2	2.36	2318.35	36.31	54.78	3.34	32.99	0.53
	3	12.25	3241.77	271.80	293.41	0.24	46.72	0.21
	4	18.45	3255.81	547.95	570.43	0	95.90	0
	5	25.11	3293.72	827.47	828.35	0	144.56	0
Southwest	1	0	20.53	9.30	9.39	0	8.44	0
	2	1.74	1245.34	26.45	382.51	5.43	13.52	1.75
	3	39.55	2955.61	47.13	59.31	4.75	17.81	1.46
	4	45.13	3287.00	69.88	463.15	3.49	32.56	1.27
	5	41.77	3600	277.86	620.95	1.11	59.71	0
Simplified	1	0	1.51	0.96	0.98	0	0.97	0
	2	0	356.86	1.80	4.72	3.00	1.57	0
	3	1.64	927.36	2.22	8.82	2.13	2.28	0.50
	4	3.29	1902.13	2.33	4.51	0.50	2.60	0
	5	2.18	1994.30	3.14	3.74	0.00	3.57	0

Table 7
Random data results.

$\lvert V\rvert$	$\lvert\bar{E}\rvert$	$\lvert\bar{B}\rvert$	$\lvert\bar{Q}\rvert$	K	R-MIP (Upper bound on KPC-ARCP)					Lower bound on KPC-ARCP	
					R-MIP		Lower bound	Upper bound	*R-MIP*	*SSV*	*KPC-ARCP*
					Exact		LB_{R-MIP}	Lagrangian Relaxation	Heuristic		
					Avg. Gap (%)	Avg. Time (s)	Avg.Time (s)	Avg. Time (s)	Avg. Gap (%)	Avg. Time (s)	Avg. Gap (%)
200	422.40	12.9	13	1	0	10.01	5.85	5.87	0	4.61	0
				2	0	208.16	6.94	7.58	1.21	7.00	1.31
				3	0.14	578.35	8.65	10.07	2.91	14.11	1.96
				4	1.28	3600	9.10	12.90	1.32	14.60	2.96
				5	0	19.37	9.35	10.73	0	15.85	1.60
250	578.11	14.9	10	1	0	8.76	6.01	6.01	0	5.96	0
				2	2.22	471.11	7.26	8.51	0.48	9.87	0.46
				3	2.91	1249.34	97.13	99.09	1.75	10.99	1.08
				4	0.49	946.43	47.13	50.03	1.69	11.58	1.32
				5	0	9.05	87.41	88.58	0	8.16	0
300	766.37	26.8	10.25	1	0	3.88	3.06	1.31	0	2.76	0
				2	0.64	1383.12	11.61	3.72	5.94	6.33	0.31
				3	2.00	1084.53	15.27	5.89	3.06	8.04	2.28
				4	0.63	2252.94	15.95	8.07	0.92	8.81	0.63
				5	0.08	469.68	16.53	5.26	0.29	9.27	0.10
350	950	12.6	10.5	1	0	5.39	4.41	4.41	0	4.31	0
				2	0	188.03	19.55	25.19	3.09	8.67	0.37
				3	0.12	853.42	27.76	37.34	1.73	10.64	0.83
				4	0.72	2083.92	35.72	43.90	2.17	11.61	1.67
				5	0	29.14	43.24	47.77	1.07	13.03	1.12

nected, if the generated graph is not connected, we add random edges between connected components to make G connected. The depot is also chosen randomly in each instance. According to the problem definition, $G_B = (V, E \setminus B)$ is a disconnected graph consisting of Q connected components. Edges are randomly added to the set B with equal probabilities to separate G_B into the desired number of connected components. The numbers of edges and blocked edges do not vary much over the 10 instances except in a particular instance with 300 nodes where 240 edges are blocked.

As the size of the network increases from 200 nodes to 350 nodes, the difficulty of the problem does not increase monotonically. In particular, while both the Southwest network generated in Section 6.3 and the second set of random networks has 250 nodes, and the average density of the random networks is larger than the Southwest network, the Southwest instances are harder in terms of being solved by *R-MIP*. For instance, *R-MIP* was solved for none of the Southwest instances with 5 vehicles in one hour but all the instances of the random network with 250 nodes were solved in 9.05 s on the average. Since the Detailed Istanbul road network has 349 nodes, random instances with 350 nodes are comparable with them in terms of size. The average degree of the nodes of the random instances with 350 nodes is 5.17 and its higher than the average degree of the nodes for the Detailed Istanbul network which is 3.95. The difficulty of the Detailed Istanbul road network is significantly higher than the randomly generated Euclidean networks with 350 nodes. It can be seen in Table 7 that Euclidean networks have fewer blocked edges. Since the average number of connected components in both of these sets are almost similar, it appears that the number of blocked edges can significantly impact the hardness of the problem.

The results of the experiments of the proposed algorithm on the Euclidean random networks which has been tested over 200 distinct topologies with 200, 250, 300 and 350 nodes with 1 to 5 vehicles in one hour time limit verify the good performance of the proposed methods and algorithms in terms of both run times and optimality gaps.

We also wanted to test the performance of *E-MIP* on these instances. However, *E-MIP* was far from even obtaining a feasible so-

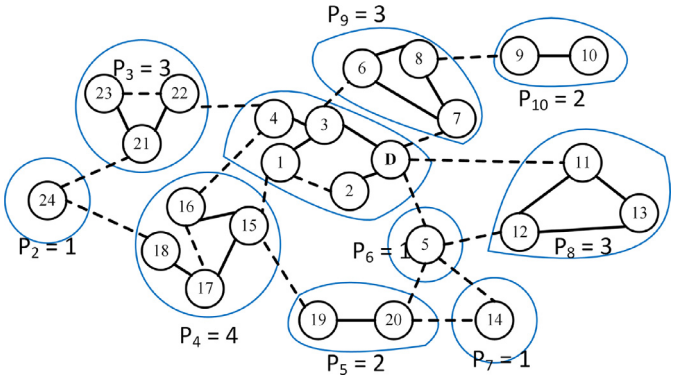

Fig. A5. Example.

lution within one hour of run time on the smallest of these instances. To be able to test the validity and computational limits of *E-MIP*, we ran it on smaller instances generated in the same way. Namely, with 40 nodes. When 4 vehicles exist, again no feasible solution could be obtained. However, when 2 vehicles exist, *E-MIP* starts to yield feasible solutions. This time 2 out of 10 instances could be solved to optimality and for the remaining 8 instances, no feasible solution could be found. This performance justifies our focus on the heuristic method in this study.

7. Conclusions and remarks

We studied a new arc routing problem in which we plan for the logistics of a road clearing operation after a disaster. In this problem, we optimize the routes of K vehicles that traverse edges of a disconnected input network due to blocked edges. The vehicles positioned in a depot node will be assigned a subset of the blocked edges and open them to maximize the number of people for whom access will be restored in a limited time, T_{max}. We find which edges to unblock and the walks of K vehicles such that all the walks are completed before T_{max}. We call this problem the Multi-vehicle Prize Collecting Arc Routing for Connectivity Problem, *KPC-ARCP*. We developed an exact MIP formulation called *E-MIP* using which only small sized instances can be solved. To solve larger instances, we developed a relaxed MIP formulation called *R-MIP*, where we ignore the timing constraints on the routes of the vehicles. Even *R-MIP* can not be solved for instances, with more than 300 nodes and 3 vehicles in a reasonable time, i.e. less than an hour. In order to obtain an upper bound on *KPC-ARCP*, we solved the Lagrangian Relaxation of *R-MIP* and developed a matheuristic that gives a lower bound to *R-MIP* (Algorithm 1). Using these bounds on *R-MIP*, we can obtain an optimality gap to *R-MIP*. Since *R-MIP* is a relaxation of *E-MIP*, the results also give an upper bound to *KPC-ARCP*. In addition, we developed a matheuristic method to obtain a feasible solution to *KPC-ARCP*. This method is called the Successive Single Vehicle Algorithm (Algorithm 2). In this algorithm, we solve single vehicle problems while updating the obtained prize of the connected components to zero at every iteration.

We tested *R-MIP*, the Lagrangian Relaxation of *R-MIP*, Algorithms 1 and 2 on both randomly generated Euclidean and Istanbul road network instances. We used three Istanbul road networks and tested 10 instances for each of them with 1 to 5 vehicles. We also tested with random networks having 200, 250, 300 and 350 nodes and 1 to 5 vehicles. The largest tested Istanbul instance has 349 nodes, 689 edges, 13 connected components and 5 vehicles. Our methods to derive an optimality gap on *KPC-ARCP* showed very good performance in tested instances resulting in either an optimal or near-optimal solution in all of the tested instances. We found an optimal solution in more than 90% of the tested instances.

The work in this study can be extended to the case where vehicles are heterogeneous with respect to their specialties and each vehicle can be assigned to a suitable subset of blocked roads. The case with multiple depots can also be analyzed. However, in this case how the connectivity prizes will be determined should be clarified.

We can solve our problem consecutively in multiple periods as new information arrives. In each period, we can use the same solution approach and update the condition of the network at the end of each period, and input it to the problem for the next period. Cases with uncertain opening times or uncertain set of blocked edges can be studied as well. If the opening times are uncertain, a scenario based stochastic programming or a robust optimization approach can be used to tackle the problem. In such cases, we still need to solve the deterministic problem efficiently as a first step. The problem of choosing the depot prior to the disaster, considering a set of damage scenarios, can also be interesting. The latter case falls into the group of location routing problems.

Acknowledgment

This research has been supported by TÜBİTAK grant 114M373.

Appendix A. Example to illustrate Algorithms 1 and 2.

Fig. A.5 is an example of *KPC-ARCP*. Four vehicles are originally positioned in the depot node denoted by *D*. These vehicles should be dispatched and restore the connectivity of components to maximize the total prize gained in a given time limit. The traversal time (c_{ij}) and the clearing time (b_{ij}) is given for this example in Table A.8. These times are in minutes and the time limit is set to 5 h ($T_{max} = 300$). The prize of each component is set to be equal to the number of nodes in that particular component.

Fig. A.6 gives the solution obtained using Algorithm 1 for the given example. In these graphs, if a vehicle opens an edge, it is shown by a dashed arrow and otherwise, it is shown by a solid arrow. The figure with caption "a) Vehicle1" shows the walk of the first vehicle. It opens (3,6) and (8,9). The total traversal time for

Table A1
Traversing and clearing times of the example.

Intact edges	c_{ij}	Intact edges	c_{ij}	Blocked edges	c_{ij}	b_{ij}	Blocked edges	c_{ij}	b_{ij}
(D,2)	10	(11,13)	15	(D,7)	20	100	(5,14)	30	135
(D,3)	20	(12,13)	30	(D,11)	50	240	(5,20)	20	120
(1,3)	5	(15,16)	15	(D,5)	25	100	(8,9)	25	140
(3,4)	5	(15,17)	15	(1,2)	15	80	(14,20)	30	160
(6,7)	30	(17,18)	10	(1,15)	15	50	(15,19)	40	200
(6,8)	15	(19,20)	10	(3,6)	15	80	(16,17)	15	85
(7,8)	20	(21,22)	15	(4,16)	40	230	(18,24)	30	150
(9,10)	10	(21,23)	10	(4,22)	20	260	(21,24)	30	150
(11,12)	20			(5,12)	20	100	(22,23)	15	85

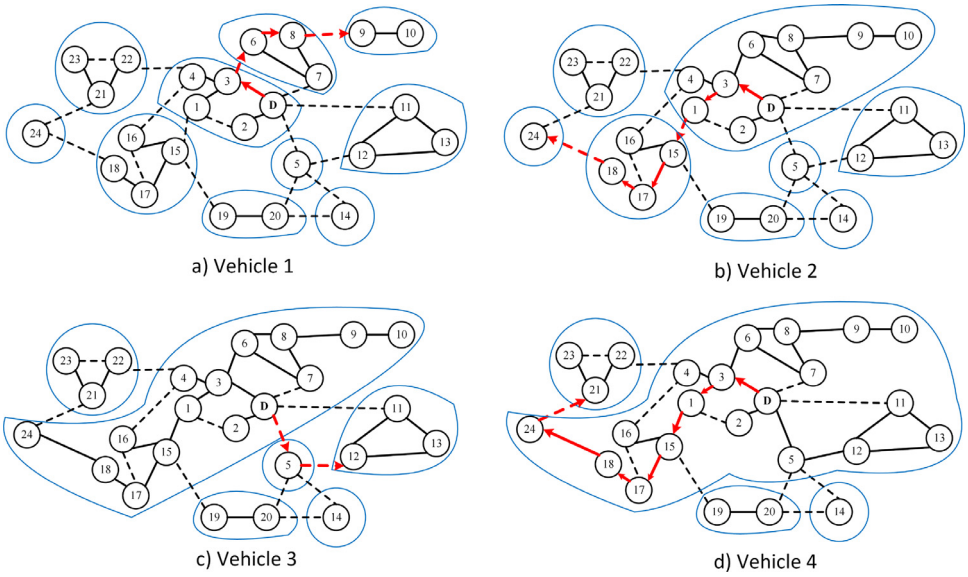

Fig. A6. Demonstration of Algorithm 1.

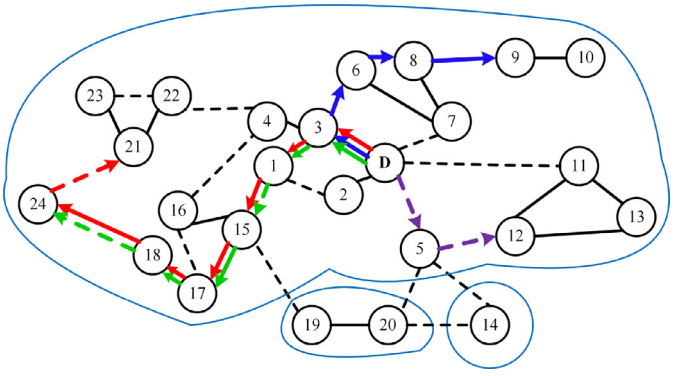

Fig. A7. Solution output by Algorithm 1.

vehicle 1 is 30 ($c_{6,7}$) + 15 ($c_{3,6}$) + 80 ($b_{3,6}$) + 15 ($c_{6,8}$) + 25 ($c_{8,9}$) + 140 ($b_{8,9}$) = 295 and since its less than 300, it does not violate the tour-length constraint. Vehicle 1 connects nodes 6,7,8,9 and 10 to the depot and obtains 5 units of prize. After the walk of the first vehicle is finished, the network will be updated and edges (6,8) and (8,9) are not blocked anymore. In "b) Vehicle2", $D \rightarrow 3 \rightarrow 1 \rightarrow 15 \rightarrow 17 \rightarrow 18 \rightarrow 24$ is the path of the second vehicle. The second vehicle unblocks (1,15) and (18,24) and obtains 5 units of prize by connecting 15,16,17,18 and 24 to the depot node. The tour length of the second vehicle is 295 and is less than T_{max}. "c) Vehicle3" shows the walk of the third vehicle. It unblocks (D,5) and (5,12) in 245 units of time and obtains 4 units of prize by connecting 5,11,12 and 13 to the depot node. Note that it can not open any other components in T_{max} hence, it finishes its walk after visiting node 12. In "d) Vehicle4", the fourth vehicle traverses $D \rightarrow 3 \rightarrow 1 \rightarrow 15 \rightarrow 17 \rightarrow 18 \rightarrow 24 \rightarrow 21$. It only opens (24,21) since (1,15) and (18,24) are already opened by the second vehicle. The tour length of vehicle 4 is 275, which is less than T_{max}; hence, we have a feasible walk. Vehicle 4 obtains 3 units of prize by connecting 21, 22 and 23 to the depot. Fig. A.7 shows the obtained walks of all of the 4 vehicles using Algorithm 1 in the given example. The vehicles all

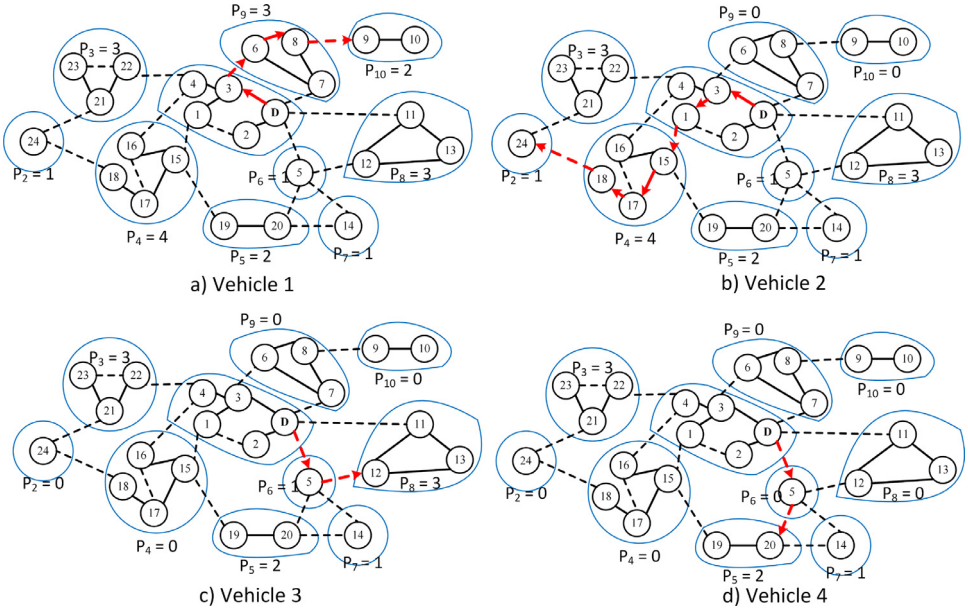

Fig. A8. Demonstration of Algorithm 2.

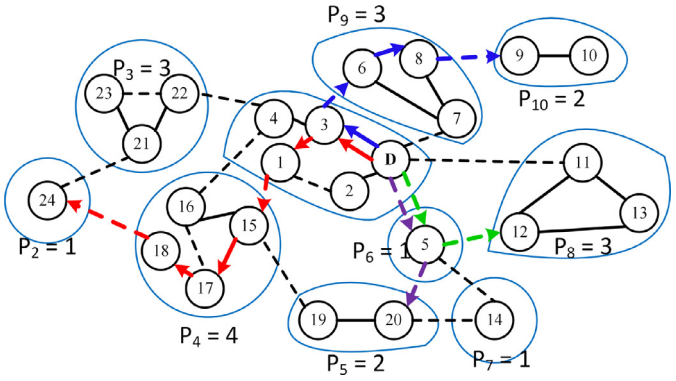

Fig. A9. Solution output by Algorithm 2.

together obtain 17 units of prize in 300 min. This corresponds to a feasible solution to *R-MIP*, in which time conflicts are allowed.

Fig. A.8 illustrates the implementation of Algorithm 2 on the same example. Different from Algorithm 1 in Fig. A.6, here the blocked edges will not be updated after they are traversed by a vehicle. For instance, vehicle 1 in "a) Vehicle1" unblocks (3,6) and (8,9) but in the next step in "b) Vehicle2" these edges are not assumed to be cleared but the prizes of their corresponding components are updated to zero to avoid connecting the same components in the walks of the next vehicles. Note that different from Fig. A.6, the fourth vehicle can not reach nodes 21 or 22 in $T_{\max}$ and it connects nodes 19 and 20 instead and obtains less total prize.

Fig. A.9 gives the obtained walks of all of the 4 vehicles using Algorithm 2 in the given example. The vehicles all together obtain 16 units of prize in 300 min. Vehicle 1 connects 6,7,8,9 and 10, vehicle 2 connects 15,16,17,18 and 24, vehicle 3 connects 5,11,12 and 13, and finally vehicle 4 connects 19 and 20 to the depot node. This is a feasible solution to *KPC-ARCP*.

References

[1] Akbari V, Salman FS. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. Eur J Oper Res 2017;257(2):625–40.
[2] Aksu DT, Ozdamar L. A mathematical model for post-disaster road restoration: enabling accessibility and evacuation. Transp Res Part E 2014;61:56–67.
[3] Aráoz J, Fernández E, Meza O. Solving the prize-collecting rural postman problem. Eur J Oper Res 2009;196(3):886–96.
[4] Aráoz J, Fernández E, Zoltan C. Privatized rural postman problems. Comput Oper Res 2006;33(12):3432–49.
[5] Archetti C, Speranza MG, Corberán Á, Sanchis JM, Plana I. The team orienteering arc routing problem. Transp Sci 2013;48(3):442–57.
[6] Asaly AN, Salman FS. Global logistics, new directions in logistics management. In: Arc selection and routing for restoration of network connectivity after a disaster. Taylor and Francis; 2014. p. 165–94.
[7] Balas E. The prize collecting traveling salesman problem. Networks 1989;19(6):621–36.
[8] Benavent E, Corberán Á, Gouveia L, Mourão MC, Pinto LS. Profitable mixed capacitated arc routing and related problems. TOP 2015;23(1):244–74.
[9] Berktaş N, Kara BY, Karaşan OE. Solution methodologies for debris removal in disaster response. EURO J Comput Optim 2016:1–43.
[10] Black D, Eglese R, Wøhlk S. The time-dependent prize-collecting arc routing problem. Comput Oper Res 2013;40(2):526–35.
[11] Celik M, Ergun O, Keskinocak P. The post-disaster debris clearance problem under incomplete information. Oper Res 2015;63(1):65–85.
[12] Christofides N, Campos VCÁ, Mota E. Netflow at Pisa. Berlin, Heidelberg: Springer Berlin Heidelberg; 1986. p. 155–66. doi:10.1007/BFb0121091.
[13] Corberán A, Laporte G. Arc routing: problems, methods, and applications. SIAM; 2015.
[14] Duque PM, Sörensen K. A GRASP metaheuristic to improve accessibility after a disaster. OR Spectrum 2011;33:525–42.
[15] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, part i: the chinese postman problem. Oper Res 1995a;43(2):231–42.
[16] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, part ii: the rural postman problem. Oper Res 1995b;43(3):399–414.
[17] Feillet D, Dejax P, Gendreau M. The profitable arc tour problem: solution with a branch-and-price algorithm. Transp Sci 2005;39(4):539–52.
[18] Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Vathis N. Heuristics for the time dependent team orienteering problem: application to tourist route planning. Comput Oper Res 2015;62:36–50.
[19] JICA. The study on a disaster prevention/mitigation basic plan in istanbul including seismic micronization in the republic of turkey. Tech. Rep.. Japan International Cooperation Agency; 2002.
[20] Kasaei M, Salman FS. Arc routing problems to restore connectivity of a road network. Transp Res Part E 2016;95:177–206.
[21] Liberatore F, Ortuño MT, Tirado G, Vitoriano B, Scaparra MP. A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. Comput Oper Res 2014;42:3–13.
[22] Ozdamar L, Aksu DT, Ergunes B. Coordinating debris cleanup operations in post disaster road networks. Socio-Econ Plann Sci 2014;48(4):249–62.
[23] Sahin H, Kara BY, Karasan O. Debris removal during disaster response: a case for Turkey. Socio-Econ Plann Sci 2016;53:49–59.
[24] Salazar-Aguilar MA, Langevin A, Laporte G. Synchronized arc routing for snow plowing operations. Comput Oper Res 2012;39(7):1432–40.
[25] Salazar-Aguilar MA, Langevin A, Laporte G. The synchronized arc and node routing problem: application to road marking. Comput Oper Res 2013;40(7):1708–15. doi:10.1016/j.cor.2013.01.007.
[26] Yan S, Shih YL. Optimal scheduling of emergency roadway repair and subsequent relief distribution. Comput Oper Res 2009;36:2049–65.