Multi-vehicle Synchronized Arc Routing Problem to Restore Post-Disaster Network Connectivity

Vahid Akbari[a], F. Sibel Salman[*a]

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

Abstract

After a natural disaster roads can be damaged or blocked by debris, while bridges and viaducts may collapse. This commonly observed hazard causes some road sections to be closed and may even disconnect the road network. In the immediate disaster response phase work teams are dispatched to open a subset of roads to reconnect the network. Closed roads are traversable only after they are unblocked/cleared by one of the teams. The main objective of this research is to provide an efficient solution method to generate a synchronized work schedule for the road clearing teams. The solution should specify the synchronized routes of each clearing team so that: 1) connectivity of the network is regained, and 2) none of the closed roads are traversed unless their unblocking/clearing procedure is finished. In this study we develop an exact Mixed Integer Programming (MIP) formulation to solve this problem. Furthermore, we propose a matheuristic that is based on an MIP-relaxation and a local search algorithm. We prove that the optimality gap of the relaxation solution is bounded by K times the lower bound obtained from the relaxed model, where K is the number of teams. We show computationally that the matheuristic obtains optimal or near-optimal solutions.
Keywords: Humanitarian logistics, network connectivity, road clearance, disaster response

1. Introduction.

The experience from past and recent disasters reveals that the roadway elements are quite vulnerable and the damages can seriously affect the transportation of people and commodities. Such high-impact incidents typically cause the network to be disconnected due to blocked roads and impede accessibility to critical locations such as hospitals, supply and shelter locations. Depending on the damage, time to clear certain road segments may take long, and victims would be waiting to get connected as the teams work on the roads.

In the immediate disaster response phase, to facilitate emergency transportation, a critical subset of the blocked roads should be cleared or repaired to restore network connectivity as much as possible. When certain connections take too long to restore, alternative modes such as helicopters would be utilized to reach the isolated areas. Hence, they can be reconnected after on in the recovery phase. The required information about damaged roads and the amount of damages can be obtained by satellite images or Volunteered

[*]Corresponding author. Tel.: +90 (212) 338 17 07
Email addresses: vakbarighadkolaei@ku.edu.tr (Vahid Akbari), ssalman@ku.edu.tr (F. Sibel Salman[*])

Geographic Information (VGI) such as up-to-date maps by Open Street Map (Coleman et al. (2009)). Work-troops equipped with required machinery and personnel should be mobilized to clear the roads and regain the connectivity of the network in the shortest time. For instance, in the case of the mentioned earthquake in Japan, prior to the disaster, contracts with construction companies were made so that the teams could be dispatched as quickly as possible, e.g. within one day. We refer to these work-troops as vehicles from now on. In our problem we assume that the vehicles are originally positioned or gathered at specific nodes of the network, referred to as depots. There might be more than one vehicle positioned at a depot. After the catastrophe, vehicles are dispatched to traverse the roads by a walk and work on the blocked edges assigned to them.

The main goal of our problem is to find the set of closed roads that need to be opened to regain the connectivity of the network, and to construct synchronized routes for the vehicles to clear or unblock these edges in the shortest time. Note that since we consider a post-disaster situation, we assume that the required information about the damaged roads is available. As information is updated over time, the problem can be resolved for the remaining isolated network components. A complexity factor arises noticing that blocked edges are not traversable unless they are completely unblocked and a team 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 edges can be traversed multiple times by different teams.

We develop an MIP model to solve the addressed problem but the timing related elements restrain solution of even moderate-sized instances. Considering that the problem should be solved in short time after the incident takes place, we propose a heuristic approach that obtains a feasible solution from the output of a relaxed model and improves it by local search procedures. Since the problem considers the multi-vehicle case and combines arc routing and network design elements and the synchronization issue, it is called *Multi-vehicle Synchronized Arc Routing Problem to Restore Network Connectivity*. We will refer to this problem by *K-ARCP* from now on, where K is the number of vehicles.

In this study we contribute to the literature by defining and studying the multi-vehicle and multi-depot synchronized arc routing problem with the connectivity objective for the first time. The problem is at least as hard as the *Min-Max K-vehicle Windy Rural Postman Problem* but it also involves the selection of the required arcs and its cost structure is more complex. Our solution method resolves the difficulties arising from the synchronization of the vehicles. We provide an efficient heuristic method with a proven worst case bound on its objective value and show that it obtains optimal or near-optimal solutions on randomly generated instances and those based on city networks. The proposed algorithm can be utilized for decision support in disaster response.

In the next section, we provide a review of the related literature and state our contributions. Section 3 gives the problem definition. In Section 4, we present the MIP models and the heuristic algorithm in Section 5. Section 6 describes data generation and discusses the computational results. We close with some concluding remarks in Section 7.

2. Literature Review.

Recently, several studies focused on upgrading a road network or improving accessibility after a disaster. Some of these studies do not address routing and instead, they focus on selecting road segments to be upgraded or repaired. For instance, Duque and Sörensen (2011) investigate the case where there is a budget constraint, and there is a number of non-operative roads which need to be repaired after a disaster. They assign weights to the rural towns depending on the importance of the towns. The objective in this problem is to minimize the weighted sum of time to travel from each rural town to its closest regional centre. Liberatore et al. (2014) study the problem of distributing goods to a population affected by an earthquake in a damaged network. They decide which edges should be opened given the repair costs and a budget. They optimize multiple criteria such as delivery time, cost, reliability, security, and satisfied demand.

Yan and Shih (2009), address the scheduling of roadway repair operations and transportation of relief items at the same time. In their problem some road segments should be repaired before a specific time due to their importance. Therefore, those tasks have to be scheduled before the other repair tasks. Time windows are also imposed. There are supply points from which relief distribution is started. Demand points have 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, in this paper authors focus on upgrading a road network and are not concerned with routing of the vehicles.

Ozdamar et al. (2014) generate roadside debris clean up plans with a given number of vehicles in the post-disaster stage. They propose an accessibility measure that considers shortest path lengths between pair of nodes over time. The shortest paths may change while the roads are restored. They develop a rule-based constructive heuristic to maximize accessibility. This study does not consider the routing of the vehicles but only assigns the road restoration tasks to the vehicles. Aksu and Ozdamar (2014) propose a path-based time-indexed integer programming (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. Hence, they do not encounter the synchronization issue in our problem. They solve the IP model for the road networks of two districts in Istanbul, Turkey. The largest instance has around 500 paths and 18 vehicles at most 5% optimality gap is obtained within an hour of solution time.

Sahin et al. (2016) is a recent study in which the authors focus on providing emergency relief supplies to disaster affected regions as soon as possible by removing the accumulated debris. The aim is to provide assistance to critical nodes by travelling along a path that may include blocked edges to be unblocked. The authors considered the single vehicle problem and solved instances with a limited number of critical nodes. An exact mathematical formulation and a heuristic algorithm are developed and used to tackle the problem.

K-ARCP differs from their problem since they consider only a single vehicle where there is no need for synchronization. Moreover, we look into the problem of connecting a disconnected graph entirely while in Sahin et al. (2016) the input graph might be connected. Furthermore, the nodes that should be visited are known, whereas in *K-ARCP* we select which nodes should be visited.

Celik et al. (2015) define a multi-period *stochastic debris clearance problem* (SDCP) with limited information on the debris amounts along the roads. The information is updated in each period and the main decision is to determine a sequence of roads to clear in each period. The problem is also concerned with sending relief items from supply point to demand points and the objective is to maximize total relief demand satisfied. They develop a Markov Decision Process and a heuristic policy for larger instances. None of the above discussed problems addresses the routing of the vehicles explicitly nor aims to reconnect a disconnected network. **?** provides emergency relief supplies to disaster affected regions as soon as possible by removing the accumulated debris. 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.

Many researchers in the transportation domain have worked on measuring and/or improving accessibility (e.g. Sohn (2006) and Faturechi and Miller-Hooks (2015)). Since we focus on reconnecting the network, we can say that we also provide accessibility, specifically in terms of connectivity between every pair of nodes, by means of generating vehicle routes and scheduling the unblocking tasks. However, our focus is on road clearance operations.

K-ARCP was first introduced as a preliminary study to this work by Akbari and Salman (2014). An optimization model in which timing elements are relaxed was introduced. Furthermore, a heuristic algorithm was proposed to obtain feasible walks from the solution of the relaxed optimization model. Tested instances were limited to a set of data from a simplified version of the Istanbul road network and a set of randomly generated data. In this study, we develop a model without relaxation of timing parameters that can solve *K-ARCP* for small instances exactly within an hour of solution time. We also develop a matheuristic using the relaxed model as in Akbari and Salman (2014) but the heuristic solution is improved by a neighbourhood search algorithm with novel moves specific to the problem. We prove that the optimal objective value of *K-ARCP* is at most K times the lower bound obtained from the relaxed model. In our computational tests the lower bound coincides with the optimal value in most cases. To test the performance of the proposed methods, we generated three different networks from the Istanbul road network using ArcGIS and Google Earth, as well as a random data set.

K-ARCP is related to the *Rural Postman Problem* (RPP) which determines a minimum cost closed walk consisting of a subset of required edges (see Eiselt et al. (1995) for a detailed review and Ghiani and Laporte (2014) and Corberán et al. (2014) for recent ones). If there is a fleet of K vehicles, then the problem of covering the required edges with K tours that minimize the total cost is called *K-RPP*. In particular, *Min-Max K-vehicle Windy Rural Postman Problem* studied by Benavent et al. (2009) is closer to the problem in our study, *K-ARCP*, as it minimizes the maximum tour cost.

The main difference of *K-ARCP* from *Min-Max K-vehicle Windy Rural Postman Problem* and other related studies which look for tours including required edges is that our main concern is the connectivity of the network. In addition, the set of required arcs are not known in advance, and there is no requirement for the walks to be closed. Also, the elapsed time for traversal and serving an edge in *RPP* is assumed to be the same, but in our case, the first traversal of a blocked edge involves unblocking it and takes the vehicle more cost/time. Moreover, the edge is ready for further use only after being unblocked. In other words, traversing a blocked edge before unblocking it is infeasible. Also, the shortest path calculations are not static as in *RPP*, and the related versions mentioned above, since after each unblocking operation there can be potentially shorter paths.

Given a network of streets and a fleet of snow plowing vehicles based at a depot, synchronized arc routing snow plowing problem introduced in Salazar-Aguilar et al. (2012) consists of determining a set of routes such that all streets some of which have multiple lanes are plowed by 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 is interpreted as plowing all lanes of a particular street simultaneously. Although synchronization has different application in Salazar-Aguilar et al. (2012), it is similar to *K-ARCP* in terms of having timing concerns in the mathematical formulation and the heuristic algorithm. Another important difference between this problem and *K-ARCP* is that in *K-ARCP* we do not know about the required edges to be opened to enable connectivity, while in Salazar-Aguilar et al. (2012) all streets should be plowed.

Salazar-Aguilar et al. (2013) is another related study in terms of synchronization. It introduces the synchronized arc and node routing problem which is on a directed graph with two sets of arcs: 1) street segments that must be painted, 2) street segments that do not need to be painted. In this problem, the authors also use two types of vehicles in the painting process: 1) a replenishment vehicle to supply paint, 2) a set of homogeneous painting vehicles. 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 in this problem refers to choosing the appropriate refill nodes so that the painting vehicles and the replenishment vehicle can meet at the same time without incurring too much waiting. One of the main differences between this problem and *K-ARCP* is that in *K-ARCP* the set of edges that need to receive a service are not predetermined. Another important difference is the fact that in *K-ARCP* a blocked edge can not be traversed unless it is opened but in this problem, the vehicles can traverse an arc without providing the service.

Asaly (2013) proposed a model for the single vehicle case and Akbari and Salman (2014) introduced a relaxed MIP formulation to derive an initial solution for *K-ARCP*. Note that in none of these formulations timing conditions are considered. Consistent with the notation in Akbari and Salman (2014), we call the relaxed MIP formulation *R-MIP*. In this study an exact model considering the required timing conditions is developed for the first time for *K-ARCP* to the best of our knowledge. We call the exact formulation *E-MIP*.

3. Problem Definition.

We model our road network as an undirected connected graph $G = (V, E)$. There is a positive cost (time) c_{ij} 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}. After the disaster, a set of edges, B, will be blocked and removal of them from the graph G, will make it disconnected. 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 represent the associated extra opening cost (time) of a blocked edge $e = (i, j) \in B$ for $i < j$ as b_{ij}. This additional time or cost incurs when a blocked edge is traversed for the first time. We assume that the edge opening times will be estimated by collecting post-disaster information on road conditions; for instance by satellite images or volunteered geographical information (VGI). Considering that roads can be used in both directions in the disaster response phase, we assume that when an edge $(i, j) \in B$ is opened, it can be traversed in both directions with the same cost.

The edges that are not blocked form a subgraph $G_B = (V, E \setminus B)$ with Q connected components. Let C_q $(q = 1, ..., Q)$ be the q^{th} connected component with node set V_q and edge set E_q. We are looking for a subset of blocked edges, R, such that $G_R = (V, (E \setminus B) \cup R)$ is connected. The solution identifies R and constructs an open walk for each vehicle that starts at its depot. The walks collectively cover R. D denotes the set of depots and P_d is the set of vehicles which are initially positioned in depot $d \in D$. The objective is to minimize the maximum cost (time) of the walks.

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, $A(W_k)$. The total walk time for vehicle k is calculated as: $T_{W_k} = C(W_k) + B(W_k) + A(W_k)$. The objective is to minimize the maximum value of T_{W_k} over $\forall k \in \{1, 2, ..., K\}$, where K shows the number of vehicles.

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 solution to* K-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 if for $k \neq k'$; $T_{ek} = T_{ek'}$, then the* opener *of edge e is the vehicle with smallest k.*

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

Definition 3. *(Feasibility) A solution to* K-ARCP *represented by W_k and $T_{W_k}, k = 1, ..., K$, is feasible if and only if: i) $G_R = (V, (E \setminus B) \cup R)$ is connected. ii) The walks $W_k, k = 1, ..., K$ 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.*

K-ARCP is NP-hard since the single vehicle problem, ARCP, is known to be NP-hard (Kasaei (2015)).

4. Mathematical Models.

In this section, we first present a mathematical model to solve *K-ARCP* exactly. Then, in Section 5 we explain our heuristic method which takes a relaxed MIP model solution as input, and proceeds with feasibility and local improvement steps.

Calculating the arrival time of the vehicles to the nodes complicates the model since edges can be traversed multiple times. Therefore, we consider a relaxed problem (*R-MIP*) such that the timing of the vehicles is ignored. After solving *R-MIP*, we may encounter one of the following situations: 1) We do not have timing conflicts in our relaxed mathematical model solution and the optimal solution of *R-MIP* is in fact the optimal solution of *K-ARCP*; 2) If the relaxed model solution has timing conflicts, then we derive a feasible solution by modifying the assignment of opening tasks to the vehicles and inserting waiting times, as necessary. In the next step, we develop a local search algorithm to improve the obtained feasible solution. The solution of *R-MIP* gives a lower bound on the optimal solution to *K-ARCP*. An upper bound on the optimal solution of *K-ARCP* is obtained from the heuristic approach. In this way, we can derive an optimality gap for the returned feasible solution.

4.1. Formulations.

The MIP model for *K-ARCP* determines K open walks such that all the separated components in the network become connected after unblocking a subset of the blocked edges. The walks start from the depots and end in a dummy sink node, indexed as $n + 1$. We address the multi-depot and K vehicle case. The K walks altogether traverse a subset of the edges in B, denoted by R, so that the graph G_R defined in Section 3 is connected.

In *R-MIP* and *E-MIP* we will use the following dominance property.

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

Proof. Christofides et al. (1986), proved a similar dominance relation for the Rural Postman Problem. Now let us assume a feasible solution to *K-ARCP* opens a set of blocked edges, R_f. Moreover, each vehicle $k \in \{1, ..., K\}$ 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 dominance property holds. The resulting single vehicle problems of *K-ARCP* differ in two aspects from the corresponding RPP problems. The walks in *K-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 dominance 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. Therefore, only the traversal costs c_{ij} will be used in the corresponding

single vehicle problems. Hence, the cost structure becomes the same as in RPP. As a result, the dominance property holds for *K-ARCP*. □

In *R-MIP* and *E-MIP*, we distinguish the direction of traversal of the edges in the walks. Therefore, we first define an arc set A such that: $\forall(i,j) \in E : (i,j) \in A$ and $(j,i) \in A$ and furthermore, for all $j \in V : (j, n+1) \in A$.

4.1.1. E-MIP.

E-MIP is the exact MIP formulation that gives the optimal solution of *K-ARCP*. It prevents timing conflicts by calculating the arrival time to blocked edges. Note that if a vehicle unblocks an edge from B, it can be traversed in both directions. We define arc set $\bar{B}$ such that: $\forall(i,j) \in B : (i,j) \in \bar{B}$ and $(j,i) \in \bar{B}$. We define the decision variables in Table 4.1 and present the constraints next.

Table 4.1: Decision Variables Used in *E-MIP*

Notation	Type	Definition
$x^k_{ijl}, \forall(i,j),(j,l) \in A, k \in \{1,...,K\}$	Binary (Proposition 1)	If vehicle k crosses (j,l) right after (i,j) in its walk
$z^k_{ij}, \forall(i,j) \in \bar{B}$	Binary	If (i,j) is unblocked by vehicle k
$f^k_{ij}, \forall(i,j) \in A$	Continuous	The flow corresponding to vehicle k's route on (i,j) from node i to j
$v^k_i, i \in i \in V \cup \{n+1\}, k \in \{1,...,K\}$	Continuous	Number of times vehicle k visits node i
$t^k_{ijl}, \forall(i,j),(j,l) \in A, k \in \{1,...,K\}$	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
w	Continuous	The value of the objective function

Variable s^k_{ij} is used to impose the necessary waiting time for vehicle $k \in \{1,2,...,K\}$ before entering an arc $(i,j) \in \bar{B}$. The following relation should hold due to Definitions 1 and 2 in Section 3:

$$s^k_{ij} = \begin{cases} T_{(i,j)k} + \Omega_{(i,j)k}, & \text{if arc } (i,j) \in \bar{B} \text{ is crossed by vehicle } k = 1,2,...,K \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$Min \quad w$$

$$t^k_{ij(n+1)} \leq w, \quad k = 1,2,\ldots,K \quad ,\forall(i,j) \in A \tag{4.2}$$

$$x^k_{ijl} \leq \sum_{h:(l,h)\in A} x^k_{jlh}, \quad \forall(i,j) \in A, \quad k = 1,2,\ldots,K, \quad l : (j,l) \in A \quad \forall k \notin P_i \tag{4.3}$$

$$x^k_{ijl} \leq \sum_{h:(h,i)\in A} x^k_{hij}, \quad \forall(i,j) \in A, \quad k = 1,2,\ldots,K, \quad l : (j,l) \in A \quad \forall k \notin P_i \tag{4.4}$$

$$\sum_{h:(h,i)\in A} x^k_{hij} \leq 1, \quad \forall(i,j) \in A, \quad k = 1,2,\ldots,K \tag{4.5}$$

$$\sum_{l:(j,l)\in A} x^k_{ijl} \leq 1, \quad \forall(i,j) \in A, \quad k = 1,2,\ldots,K \tag{4.6}$$

$$\sum_{j:(d,j)\in A} \sum_{l:(j,l)\in A} x^k_{djl} = 1, \quad \forall d \in D, \quad \forall k \in P_d \tag{4.7}$$

$$\sum_{(i,j)\in A} x^k_{ij(n+1)} = 1, \quad k = 1,2,\ldots,K \tag{4.8}$$

$$\sum_{l:(j,l)\in A} x^k_{ijl} \geq z^k_{ij}, \quad \forall (i,j) \in \bar{B}, \quad k = 1,2,...,K \tag{4.9}$$

$$\sum_{l:(j,l)\in A} x^k_{ijl} + \sum_{h:(i,h)\in A} x^k_{jih} \leq 2(\sum_{\kappa=1}^{K} z^\kappa_{ij} + \sum_{\kappa=1}^{K} z^\kappa_{ji}), \quad \forall (i,j) \in B, \quad k = 1,2,\ldots,K \tag{4.10}$$

$$\sum_{\kappa=1}^{K} (z^\kappa_{ij} + z^\kappa_{ji}) \leq 1, \quad \forall (i,j) \in B \tag{4.11}$$

$$\sum_{j:\in V\cup\{(n+1)\}:(i,j)\in A} (f^k_{ij} - f^k_{ji}) = -v^k_i, \;\; k = 1,2,\ldots,K, \;\; \forall i \in V \cup \{(n+1)\} \setminus D \tag{4.12}$$

$$\sum_{j\in V\cup\{(n+1)\}} (f^k_{dj} - f^k_{jd}) = \sum_{i\in V\cup\{(n+1)\}\setminus\{d\}} v^k_i, \quad \forall k \in P_d, \quad \forall d \in D \tag{4.13}$$

$$\sum_{j\in V} f^k_{j(n+1)} = 1, \quad k = 1,2,\ldots,K \tag{4.14}$$

$$f^k_{ij} \leq (n-1) \sum_{l:(j,l)\in A} x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in A, \quad i,j \in V \cup \{(n+1)\} \tag{4.15}$$

$$f^k_{ij} \geq \sum_{l:(j,l)\in A} x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in A, \quad i,j \in V \cup \{(n+1)\} \tag{4.16}$$

$$\sum_{j:(j,i)\in A} \sum_{l:(i,l)\in A} x^k_{jil} = v^k_i, \quad k = 1,2,\ldots,K, \quad \forall i \in V \cup \{(n+1)\} \tag{4.17}$$

$$\sum_{\kappa=1}^{K} \sum_{(i,j)\in\delta(S)} z^\kappa_{ij} \geq 1, \quad \forall S \subset \{1,...,Q\}, S \neq \emptyset \tag{4.18}$$

$$t^k_{ijl} \leq M x^k_{ijl}, \quad k = 1,2,\ldots,K, \quad i : (i,j) \in A, \forall l : (j,l) \in A \tag{4.19}$$

$$s^k_{ij} \leq \sum_{h:(h,i)\in A} t^k_{hij} + M(1 - z^k_{ij}), \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in \bar{B} \tag{4.20}$$

$$s^k_{ij} \geq \sum_{h:(h,i)\in A} t^k_{hij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in \bar{B} \tag{4.21}$$

$$s^k_{ij} \geq \sum_{l:(j,l)\in A} t^\kappa_{ijl} - M(1 - z^\kappa_{ij}) - 2M(1 - \sum_{h:(h,i)\in A} x^k_{hij}),$$
$$\forall k,\kappa \in \{1,2,\ldots,K\}, \quad \forall (i,j) \in \bar{B}, \quad and \quad \kappa \neq k \tag{4.22}$$

$$\sum_{l:(j,l)\in A} t^k_{ijl} \geq s^k_{ij} + b_{ij} z^k_{ij} + \sum_{l:(j,l)\in A} x^k_{ijl} c_{ij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in \bar{B} \tag{4.23}$$

$$\sum_{l:(j,l)\in A} t^k_{ijl} \geq \sum_{h:(h,i)\in A} t^k_{hij} + \sum_{l:(j,l)\in A} x^k_{ijl} c_{ij}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in A \setminus \bar{B} \tag{4.24}$$

Constraint (4.2) sets the objective. Note that all walks end in the sink node indexed as $n+1$. Constraints from (4.3) to (4.6) are flow balance equations. A vehicle should enter an arc first, 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 (4.3) and (4.4) set these properties. Each arc is traversed at most once by a particular vehicle, Constraints (4.5) and (4.6) enforce this property. Each vehicle starts its walk in its depot and ends it in the sink node. This

property is ensured by Constraints (4.7) and (4.8). Constraints from (4.9) to (4.11) set the relation between x^k_{ijl} and z^k_{ij} decision variables. Constraint (4.9) forces that a blocked edge gets open only if one of the vehicles crosses it. Constraints (4.10) ensure that traversing a blocked edge is only possible after unblocking it by one of the vehicles. By (4.11), a blocked edge should be opened by at most one vehicle. 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 equals to the number of visits to the corresponding node. Figure 4.1 gives an example of the usage of the flow variables and how they enable the route of a vehicle to be connected. In the given walk in Figure 4.1, 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 (4.12) and (4.13) ensure that a vehicle

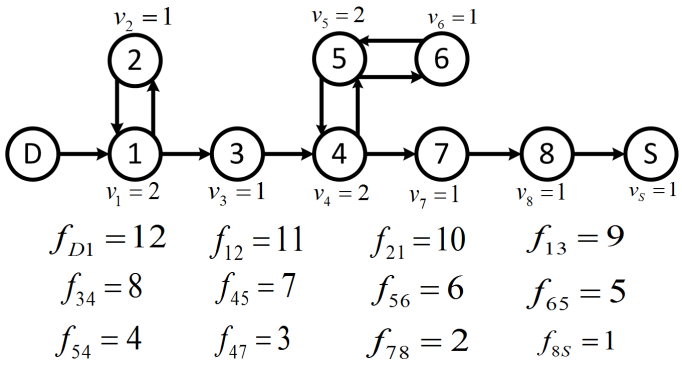


Figure 4.1: Demonstration of the Flow Formulation

leaves one unit of flow each time it visits a node. Constraints (4.14) ensures that walks end in the sink node. Constraint (4.15) does not allow flow on an edge unless it is traversed. Constraint (4.16) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Constraint (4.17) counts the number of times vehicle k visits node $i \in V$. In order to ensure the connectivity of the network, we consider the cutsets separating each subset of connected components from the rest of the nodes in G. Let $S \subset \{1, ..., Q\}$, $S \neq \emptyset$ and $V_S = \bigcup_{q \in S} V_q$ then $\delta(S) = (i, j) \in \bar{B} : i \in Vs, j \notin Vs$. Constraint (4.18) enforces that at least one edge in $\delta(S)$ should be unblocked to guarantee the connectivity of G_R. Constraint sets (4.19) to (4.24) are for setting time limits. Constraint (4.19) ensures that, if $x^k_{ijl} = 0$, then $t^k_{ijl} = 0$, where M can be set to $\sum_{(i,j)\in A} c_{ij} + \sum_{(i,j)\in B} b_{ij}$. Constraints (4.20), (4.21) and (4.22) are used to calculate s^k_{ij} values. 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 cross 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 (4.20) and (4.21), if vehicle k is the *opener* of edge $(i, j) \in B$, the earliest time that it is traversable by k is the time when it arrives to (i, j). On the other hand, if $z^k_{ij} = 0$, then (4.20) is redundant, and by (4.21) vehicle k can start traversing (i, j) no sooner than its arrival time to node i. If a vehicle is traverses 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 $(i, j) \in \bar{B}$. Constraint sets (4.23) and (4.24) calculate the arrival times to an edge (i, j) for $(i, j) \in \bar{B}$ and $(i, j) \in A \setminus \bar{B}$, respectively.

The following constraints are the restrictions on the variables.

$$x_{ijl}^k \in \{0, 1\}, \qquad k = 1, 2, \ldots, K, \quad i : (i, j) \in A, \forall l : (j, l) \in A \tag{4.25}$$

$$z_{ij}^k \in \{0, 1\}, \qquad \forall (i, j) \in \bar{B}, \;\; k = 1, 2, ..., K \tag{4.26}$$

$$f_{ij}^k \geq 0, \qquad \forall (i, j) \in A, \;\; k = 1, 2, ..., K \tag{4.27}$$

$$v_i^k \geq 0, \qquad \forall i \in V, \;\; k = 1, 2, ..., K \tag{4.28}$$

$$t_{ijl}^k \geq 0, \qquad k = 1, 2, \ldots, K, \quad i : (i, j) \in A, \forall l : (j, l) \in A \tag{4.29}$$

$$s_{ij}^k \geq 0, \qquad \forall (i, j) \in \bar{B}, \;\; k = 1, 2, ..., K \tag{4.30}$$

This formulation has $O(n^3 K)$ decision variables.

4.1.2. R-MIP.

In this section we present *R-MIP* which solves a relaxation of *K-ARCP* where timing conditions are ignored. We define the decision variables and present the constraints next.

Table 4.2: Decision Variables Used in *R-MIP*

Notation	Type	Definition
$x_{ij}^k, \forall (i, j) \in A, k \in \{1, ..., K\}$	Binary	If vehicle k traverses (i, j) from node i to j
$z_{ij}^k, \forall (i, j) \in B$	Binary	If (i, j) is unblocked by vehicle k
$f_{ij}^k, \forall (i, j) \in A$	Continuous	The flow corresponding to vehicle k's route on (i, j) from node i to j
$v_i^k, i \in i \in V \cup \{n + 1\}, k \in \{1, ..., K\}$	Continuous	Number of times vehicle k visits node i
y	Continuous	The value of the objective function

Minimize y

subject to

$$\sum_{(i,j)\in A} c_{ij} x_{ij}^k + \sum_{(i,j)\in B} b_{ij} z_{ij}^k \leq y, \quad k = 1, 2, \ldots, K \tag{4.31}$$

$$\sum_{j\in V\cup\{(n+1)\}:(d,j)\in A} (x_{dj}^k - x_{jd}^k) = 1, \quad \forall d \in D, \quad \forall k \in P_d \tag{4.32}$$

$$\sum_{j\in V\cup\{(n+1)\}:(i,j)\in A} (x_{ij}^k - x_{ji}^k) = 0, \quad k = 1, 2, \ldots, K, \quad \forall i \in V\setminus D \tag{4.33}$$

$$\sum_{j\in V} x_{j(n+1)}^k = 1, \quad k = 1, 2, \ldots, K \tag{4.34}$$

$$x_{ij}^k + x_{ji}^k \geq z_{ij}^k, \quad k = 1, 2, \ldots, K, \quad \forall (i, j) \in B \tag{4.35}$$

$$x_{ij}^k + x_{ji}^k \leq 2 \sum_{\kappa=1}^{K} z_{ij}^\kappa, \quad k = 1, 2, \ldots, K, \quad \forall (i, j) \in B \tag{4.36}$$

$$\sum_{\kappa=1}^{K} z_{ij}^{\kappa} \leq 1, \quad \forall (i,j) \in B \tag{4.37}$$

$$\sum_{j:(j,i)\in A} x_{ji}^{k} = v_i^k, \quad k = 1,2,\ldots,K, \quad \forall i \in V \cup \{(n+1)\} \tag{4.38}$$

$$f_{ij}^{k} \leq (n-1)x_{ij}^{k}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in A, \quad i,j \in V \cup \{(n+1)\} \tag{4.39}$$

$$f_{ij}^{k} \geq x_{ij}^{k}, \quad k = 1,2,\ldots,K, \quad \forall (i,j) \in A, \quad i,j \in V \cup \{(n+1)\} \tag{4.40}$$

$$\text{(4.12), (4.13), (4.14), (4.18)} \tag{4.41}$$

Constraints (4.32) and (4.33) are vehicle flow balance equations. Constraint (4.32) ensures that every vehicle leaves its depot. Constraint (4.34) forces each walk to end in the sink node, which is indexed by $n+1$, where n is the number of nodes. By constraint (4.35), if a blocked edge is opened, it must be traversed at least once from one of the directions. Constraint (4.36) prevents traversing a blocked edge, if it is not unblocked by any of the vehicles. Since the graph is assumed to be undirected, if a blocked edge is selected to be opened, it is enough to open it in one direction. This is enforced by (4.37). A node is visited if and only if an edge entering that node is traversed (4.38). Constraint (4.39) does not allow flow on an edge unless it is traversed. Constraint (4.40) shows that if an edge is traversed, then there must be a positive amount of flow passing through it. Note that since the dimension of variable x in *E-MIP* and *R-MIP* is different, all the constraints are not similar to each other. The dimension of variable x is decreased in *R-MIP* to obtain the solution faster. Note that it is not possible to formulate timing constraints with three dimensional x variables.

The following constraints are the restrictions on the variables.

$$x_{ij}^{k} \in \{0,1\}, \qquad \forall (i,j) \in A, \;\; k = 1,2,...,K \tag{4.42}$$

$$z_{ij}^{k} \in \{0,1\}, \qquad \forall (i,j) \in B, \;\; k = 1,2,...,K \tag{4.43}$$

$$f_{ij}^{k} \geq 0, \qquad \forall (i,j) \in A, \;\; k = 1,2,...,K \tag{4.44}$$

$$v_{i}^{k} \geq 0, \qquad \forall i \in V, \;\; k = 1,2,...,K \tag{4.45}$$

This formulation has $O(n^2K)$ binary variables and an exponential number of constraints due to (4.18). However, when the number of components is small (as expected in our application), the number of constraints is computationally manageable.

Proposition 2. *If Z^*_{R-MIP} shows the optimal objective value of* R-MIP *and $Z^*_{K\text{-}ARCP}$ denotes the optimal objective value of* K-ARCP. *Then the following inequality holds.*

$$Z^*_{R-MIP} \leq Z^*_{K\text{-}ARCP} \leq K \cdot Z^*_{R-MIP} \tag{4.46}$$

The proof to Proposition 2 is given in Appendix A.

5. The Relaxation-Based Heuristic (RBH) for K-ARCP.

After solving the relaxed model for *K-ARCP*, *R-MIP*, we need to extract the steps of the walks for every vehicle in the network. We can obtain the walk of each vehicle using the *Walk Extraction Algorithm* proposed in Akbari and Salman (2014). This algorithm is given in Appendix B. Since we ignored time-related parameters in *R-MIP*, these walks might not be synchronized. In Section 5.1, we explain the heuristic algorithm to obtain a feasible solution. Finally, in Section 5.2, we explain the local search algorithm developed to improve the feasible solution found according to Section 5.1.

5.1. The Feasibility Algorithm.

We derive a feasible solution by modifying the solution obtained from *R-MIP* as follows. First, we find the time at which the vehicles arrive to each blocked edge and determine those edges with timing conflict, that is, we detect whether a vehicle traverses an edge before it is unblocked. Next, we find the blocked edge with the smallest completion time of its unblocking task over all vehicles. We insert waiting times whenever it is necessary by shifting the corresponding completion times forward. Each time we process the next edge on the next smallest completion time over all vehicles. We repeat this procedure until all the time conflicts are prevented. In Figure 5.3, we present an illustrative example where the solution to *R-MIP* is infeasible to *K-ARCP* and a feasible solution is derived.

We next describe the *Feasibility Algorithm* in detail. With the results obtained from the *Walk Extraction Algorithm*, we form a solution table such as Table 5.1. In Table 5.1, edges in B that are traversed by different vehicles and the corresponding times are given. Here, $B_{i,j}$ shows the j^{th} blocked edge in the walk of vehicle i and $\tau_{i,j}$ is the time when traversing this edge is completed.

Table 5.1: Order of Edges from B Visited by Vehicles (1 to K)

Walk	**Order of Blocked Edges**	**Finishing Time of the walks**
W_1	$B_{1,1} \Rightarrow B_{1,2} \Rightarrow B_{1,3} \Rightarrow \ldots \Rightarrow B_{1,m(1)}$	$\tau_{1,1} \Rightarrow \tau_{1,2} \Rightarrow \tau_{1,3} \Rightarrow \ldots \Rightarrow \tau_{1,m(1)}$
W_2	$B_{2,1} \Rightarrow B_{2,2} \Rightarrow B_{2,3} \Rightarrow \ldots \Rightarrow B_{2,m(2)}$	$\tau_{2,1} \Rightarrow \tau_{2,2} \Rightarrow \tau_{2,3} \Rightarrow \ldots \Rightarrow \tau_{2,m(2)}$
.	.	.
.	.	.
.	.	.
W_K	$B_{K,1} \Rightarrow B_{K,2} \Rightarrow B_{K,3} \Rightarrow \ldots \Rightarrow B_{K,m(K)}$	$\tau_{K,1} \Rightarrow \tau_{K,2} \Rightarrow \tau_{K,3} \Rightarrow \ldots \Rightarrow \tau_{K,m(K)}$

Note that a blocked edge may be repeated in the walk of a vehicle, or it may appear in the walks of more than one vehicle. During the algorithm, as we process the edges given in Table 5.1, we shift the finishing time of those edges that are affected and remove the processed edges from Table 5.1. If vehicle k is not the *opener* of any edge, then $w_{k,1}$ consists of only the edge from the depot of vehicle k to $n+1$ and $t_{k,1} = 0$. Consequently, the k^{th} row of Table 5.1 is empty.

Feasibility Algorithm

Step 1: If Table 5.1 is empty, the routes are synchronized and the algorithm terminates. Otherwise, choose $\psi = min\{\tau_{i,j}\} - z^i_{uv} b_{uv} - c_{uv}$ over all $i = 1, ..., K$, $j = 1, ..., m(K)$, where edge (u, v) is the j^{th} blocked edge of W_i. Recall that b_{uv} and c_{uv} are the extra cost of unblocking (u, v) and the cost of traversing (u, v), respectively.

Step 2: Determine the vehicle index, κ, associated with ψ s.t. $\psi = min\{\tau_{\kappa,j}\} - z^\kappa_{uv} b_{uv} - c_{uv}$. Note that (u, v) is the j^{th} edge from the set of blocked edges, B, traversed by vehicle κ. Go to one of the following cases considering z^κ_{uv} value:

Case 1. **If $z^\kappa_{uv} = 1$:** No timing conflict happens since vehicle κ is opening edge (i, j) in its first visit and it is also the first time that (u, v) is traversed.

(a) Update Table 5.1 by removing all $B_{i,j}$ and $\tau_{i,j}$ values s.t. j is the step of W_i that is traversing (u, v). Go to Step 1.

Case 2. **If $z^\kappa_{uv} = 0$:** In this case, vehicle κ is the first vehicle that traverses (u, v) without unblocking it. Since we do not have a feasible solution for *K-ARCP* yet, we synchronize the walks by setting κ as the *opener* of (u, v) and update the walks of other affected vehicles as follows:

(a) Determine vehicle ρ such that $z^\rho_{uv} = 1$. It means that ρ is the *opener* of (u, v) in the optimal solution of *R-MIP*. (For simplicity of notation let us assume vehicle ρ completes opening edge (u, v) at time τ_ρ.)

(b) *(Making changes in the walk of vehicle κ.)* Since vehicle κ is the first vehicle that traverses edge (u, v), it should be set as the *opener* of (u, v) instead of ρ. Furthermore, let us assume that vehicle κ traverses edge (u, v) at step β of its walk. Next, we shift forward the time of all walk steps after β for vehicle κ by b_{uv}. Thus, $\forall \tau_{\kappa,j} : j \geq \beta$, set $\tau_{\kappa,j} = \tau_{\kappa,j} + b_{uv}$.

(c) *(Making changes in the walk of vehicle ρ.)* Vehicle ρ should not be the *opener* of (u, v) in the updated solution. Changes should be made for the steps of W_ρ that are after opening (u, v). These changes should handle either of the following situations: 1) vehicle ρ waits at edge (u, v) for vehicle κ to finish the unblocking task, or 2) when vehicle ρ arrives to edge (u, v), its opening is already finished by vehicle κ. Then, the shifts should be as follows: $\forall \tau_{\rho,j}, j \in 1, ..., m(\rho) : \tau_\rho \leq \tau_{\rho,j}$, set $\tau_{\rho,j} = \tau_{\rho,j} - \tau_\rho + max\{\tau_{\kappa,\beta} + c_{uv}, \tau_\rho - b_{uv}\}$. Note that $\tau_{\kappa,\beta}$ is the time when unblocking task of vehicle κ on edge (u, v) is completed due to the previous step of the algorithm. If $\tau_{\kappa,\beta} < \tau_\rho - b_{uv} - c_{uv}$, then when vehicle ρ arrives to edge (u, v), its unblocking task by vehicle κ is completed and ρ can finish traversing (u, v) at $\tau_\rho + c_{uv}$. On the other hand, if $\tau_{\kappa,\beta} > \tau_\rho - b_{uv} - c_{uv}$, vehicle ρ should wait at edge (u, v) for vehicle κ to finish the opening task, and ρ can enter (u, v) at time $\tau_{\kappa,\beta}$. Consequently, its finishing time on edge (u, v) should be at least $\tau_{\kappa,\beta} + c_{uv}$.

(d) *(Making changes in the walk of other affected vehicles)* If $\exists \eta : \eta \neq \kappa \neq \rho$ and $x^\eta_{uv} + x^\eta_{vu} \geq 1$. This means that vehicle η also traverses edge (u, v). Let us assume this traversal occurs at the θ^{th} blocked edge, traversed by η. Furthermore, if $\tau_{\eta,\theta} - c_{uv} \leq \tau_{\kappa,\beta}$, vehicle η traverses edge (u, v) before its unblocking task is finished. Therefore, the steps of W_η that are after θ should be shifted forward as follows: $\forall \tau_{\eta,j} : j \geq \theta$, set $\tau_{\eta,j} = \tau_{\eta,j} + \tau_{\kappa,\beta} - \tau_{\eta,\theta} + c_{uv}$.

(e) Update Table 5.1 by removing all $B_{i,j}$ and $\tau_{i,j}$ values s.t. j is the step of W_i that is traversing (u, v), then go back to Step 1. □

Let us denote the objective function value obtained from the result of feasibility algorithm by Z_f. Note that this value corresponds to the time of the longest walk among all the vehicles.

Proposition 3. *The Feasibility Algorithm always gives a feasible solution to K-ARCP whose objective value is at most $K \cdot Z^*_{R\text{-}MIP}$, which is at most $K \cdot Z^*_{K\text{-}ARCP}$. The following inequality always holds:*

$$Z_f \leq K \cdot Z^*_{K\text{-}ARCP} \tag{5.1}$$

Proof. The optimal solution of R-MIP corresponds to a set of walks, $w_{R\text{-}MIP}$. Let us assume the walk of the i^{th} vehicle in $w_{R\text{-}MIP}$ is shown by w^*_i. In consistence with current notations, $T_{w^*_i}$ shows the time/cost of the walk of vehicle i. In the worst case of the feasibility algorithm, each vehicle i should wait for another vehicle to finish its walk entirely and then start traversing w^*_i. It means that the length of the longest walk among all the vehicles in the updated solution after implementing feasibility algorithm on $w_{R\text{-}MIP}$ is at most the summation of all vehicles' walks in $w_{R\text{-}MIP}$, i.e. $Z_f \leq \sum\limits_{i \in 1,\ldots,K} T_{w^*_i}$. On the other hand, due to the definition of K-ARCP, $Z^*_{R\text{-}MIP} = \max\limits_{i=1,\ldots,K} T_{w^*_i}$. Since $\frac{\sum\limits_{i=1}^{K} T_{w^*_i}}{K} \leq \max\limits_{i=1,\ldots,K} T_{w^*_i}$, we obtain:

$$Z_f \leq \sum_{i=1}^{K} T_{w^*_i} \leq K \cdot \max_{i=1,\ldots,K} T_{w^*_i} = K \cdot Z^*_{R\text{-}MIP} \tag{5.2}$$

Since $Z^*_{R\text{-}MIP} \leq Z^*_{K\text{-}ARCP}$, hence;

$$Z_f \leq K \cdot Z^*_{R\text{-}MIP} \leq K \cdot Z^*_{K\text{-}ARCP} \tag{5.3}$$

This confirms the validity of (5.1) and proves Proposition 3. □

5.2. The Local Search Algorithm.

We developed a *Local Search* algorithm to improve the solution obtained by the *Feasibility Algorithm*. Various procedures were proposed and used for the design of heuristic methods in the arc routing context (see Hertz et al. (1999) and Hertz et al. (2000)). None of these procedures is directly applicable for *K-ARCP*.

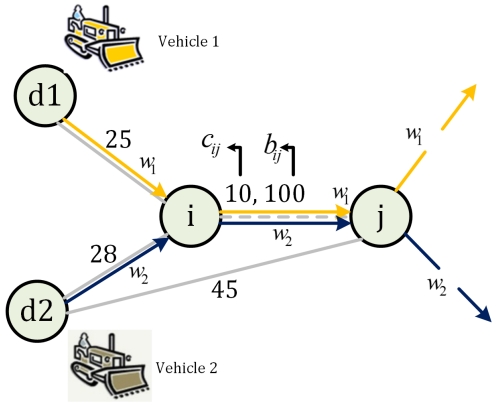
In the following we define new procedures that we developed particularly for *K-ARCP*. In the first three moves, *Remove*, *Shorten* and *Cut*, the input is a walk by a vehicle and the output is possibly a shorter walk for the same vehicle.

Procedure Remove: Find the last edge traversed in the walk of a particular vehicle. If this edge involves unblocking, the output is the current walk. On the other hand, if it does not involve unblocking, remove the last edge from the walk and repeat the procedure again.

Procedure SHORTEN: Consider the walks generated in Table 5.1 and the path between two consecutive blocked edges in each walk. Check if all such paths are shortest paths from their origin node to destination node. If not, modify the route to satisfy this property.

Procedure CUT: Using the walks generated in Table 5.1, check to see if there is any shorter path to avoid traversing blocked edges by vehicles that are not opening them. For instance, as it is shown in Figure 5.1, the optimal solution of the relaxed model chooses $d2 \rightarrow i \rightarrow j$ for the second vehicle since it can arrive to node j at time 38, considering that the first vehicle is opening the blocked edge (i, j) and ignoring the necessary waiting time. However, it should wait until the opener vehicle unblocks (i, j). Applying the *Cut* procedure, the second vehicle goes from $d2$ to j directly to avoid waiting at node i. Unlike the case in the *Shorten* procedure where we try to choose the shortest path from a blocked edge to the next one, in procedure *Cut*, to prevent waiting, the current vehicle may skip traversing some edges that require unblocking by other vehicles.

Figure 5.1: Illustration of Procedure Cut



In all of the following three procedures, namely, *Change Route*, *Change Opener* and *Change Route and Opener*, the input is two walks by two different vehicles traversing the same blocked edge and the output is also two walks where the maximum time of the new walks is at most the maximum time of the input walks.

Procedure CHANGE ROUTE: Let $W_{k'}$ be the longest walk among all vehicles. Check if there is any other vehicle, k'' whose walk intersects with $W_{k'}$ on a blocked edge (i, j). If so, pick one of those vehicles such that changing the routes of k' and k'' gives the best improvement in the objective function and then inter-change the walks of these two vehicles. Considering that an edge might be traversed by two vehicles k' and k'', either in the same direction or in the opposite direction, one of the cases shown in Figure 5.2 happens. In Case (A1) of Figure 5.2, vehicles 1 and 2 traverse (i, j) in the same direction, while edge $(i, j) \in B$ and vehicle 1 is its *opener* in the current solution. (A2) and (B2) show an implementation of *Change Route* procedure. In (A2) the walks of vehicle 1 and 2 are inter-changed after node j. A similar procedure can be implemented while two different vehicles traverse a common blocked edge in opposite directions. In Figure 5.2, Case (B2) shows an instance where vehicles 1 and 2 traverse $(i, j) \in B$ in opposite directions. It is also assumed that vehicle 1 is the *opener* of (i, j) in the initial solution (B1). In (B2) walks of vehicles 1 and 2

are inter-changed at node i. Note that in (A2) and (B2), although the walks of vehicles 1 and 2 are changed, the *opener* of (i, j) remains the same.

Procedure CHANGE OPENER: As in the *Change Route* procedure, in *Change Opener* we look into $W_{k'}$ such that it is the longest walk among all vehicles. We check if there is any other vehicle k'' that traverses a common blocked edge with k' either in the same direction or opposite directions. In the *Feasibility Algorithm*, we set the vehicle traversing an edge for the first time to unblock it. It is possible that if this vehicle waits for the second vehicle to unblock the edge, we can decrease the maximum time among the two vehicles. If the vehicle that includes the longest walk in the routes is one of these two vehicles, we can obtain a better solution. An instance of *Change Opener* procedure is given in Cases (A3) and (B3) of Figure 5.2. In (A1) and (B1), vehicle 1 is the *opener* of (i, j) in $i \rightarrow j$ and $j \rightarrow i$ directions, respectively. In (A3) and (B3), the *opener* is changed from vehicle 1 to vehicle 2.

Procedure CHANGE ROUTE AND OPENER: This procedure is a hybrid of *Change Route* and *Change Openers* procedures. This procedure is depicted in Figure 5.2 (Case (A4) and (B4)). In (A4), not only the *opener* of (i, j) changes, but also the walks of vehicles 1 and 2 are inter-changed after node j. Recall that in *Change Opener* only the *opener* of (i, j) changes from vehicle 1 to 2 and the vehicles follow their original walks after node j (A3). On the other hand, in *Change Route and Opener*, not only the *opener* of (i, j), but also the walks will change after node j. Case (B4) shows an instance of this procedure where vehicle 1 and 2 traverse (i, j) in the opposite directions. Note that it differs from Case (B2) of Figure 5.2, as the *opener* of (i, j) also changes from vehicle 1 to vehicle 2.

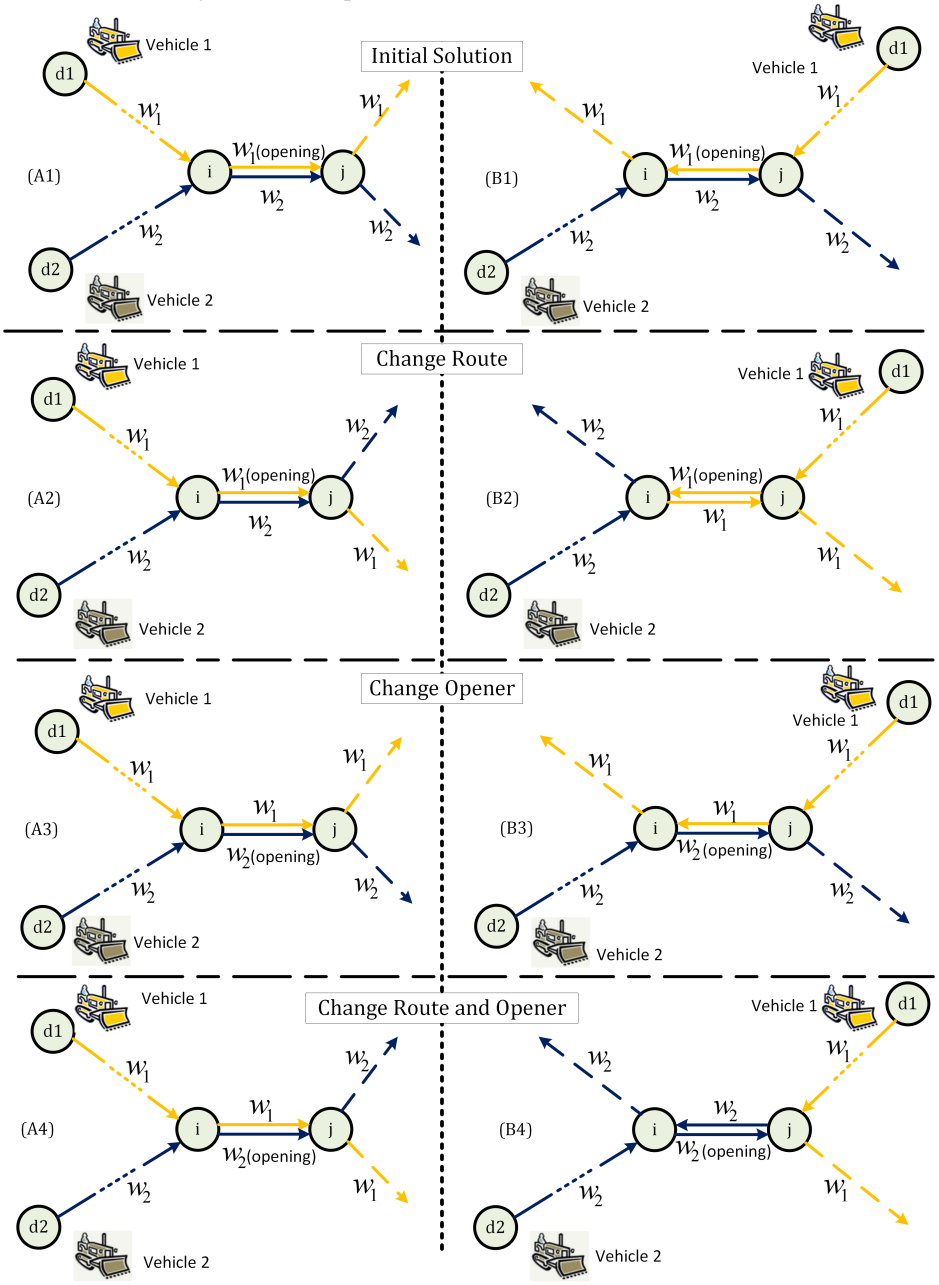
5.3. The Relaxation-Based Heuristic (RBH).

The inputs of this heuristic algorithm are the K walks derived from solving *R-MIP* and the output is the set of K updated and synchronized walks without timing conflicts. The steps of the proposed heuristic algorithm are given below.

The steps of this algorithm are explained on an instance in Figures from 5.3 to 5.7. In these figures, the walks of three vehicles are shown by w_1, w_2 and w_3. The two numbers on blocked edges show c_{ij} and b_{ij} corresponding to traversing and unblocking (i, j), respectively. For edges that are not blocked, only the c_{ij} value is given on the edge. The direction of traversing the edges and the current times in each walk are shown in each figure for every vehicle as well. For the considered instance, by using the *Walk Extraction Algorithm*, we obtain the solution of *R-MIP* shown in Figure 5.3, where timing conflicts occur on edges (6,9) and (4,5). For this infeasible solution, the optimal value, which is the longest walk length among the three vehicles, is 80.

In the second step, after implementing the *Feasibility Algorithm*, timing of w_2 changes. The first vehicle that arrives to a blocked edge opens it and the second vehicle that traverses the same edge, should wait until the unblocking procedure is finished. As we see in Figure 5.4, the length of the longest walk, i.e. the objective function value, is 100.

Figure 5.2: Example to Illustrate the Local Search Procedures



In the third step shown in Figure 5.5, the route of w_2 changes. The last step of w_2 is removed since it does not involve unblocking a blocked edge due to *Remove* procedure (the third step in *RBH*). The same vehicle follows the path $8 \rightarrow 5$ instead of $8 \rightarrow 6 \rightarrow 9 \rightarrow 5$ to avoid waiting on edge (6,9), due to *Cut* procedure (the fourth step of *RBH*).

In Figure 5.6, we implement the *Feasibility Algorithm* again and it involves changing the opener on edge (4,5). We achieve 85 as the objective value in this step.

In the final step, the procedure *Change Route* is applied at node 4 so that the routes of w_2 and w_3 are interchanged from node 4 onward (Figure 5.7). The final objective value which corresponds to a feasible

Algorithm 1 Relaxation Based Heuristic for K-ARCP

Input:

the solution of *R-MIP*

1: Apply the *Walk Extraction Algorithm* on the input.
2: Apply the *Feasibility Algorithm*.
3: Apply the *Remove Procedure* on each individual walk of the solution.
4: Apply the *Shorten and Cut Procedures* on each individual walk of the solution.
5: Apply the *Feasibility Algorithm* again on the solution to get a feasible solution.
6: Check if any of the *Change Route*, *Change Opener* or *Change Route and Opener* procedures are possible to apply for the current data.
7:
- **If No:** Current solution is the output of *RBH*.
- **If Yes:** Check if this change results in better bounds:
  - *If No:* Current solution is the output of *RBH*.
  - *If Yes:* Apply the best option out of *Change Route*, *Change Opener* or *Change Route and Opener* procedures considering the length of the walks after making them feasible using the *Feasibility Algorithm*.

8: Repeat Step 6.

Figure 5.3: R-MIP Solution

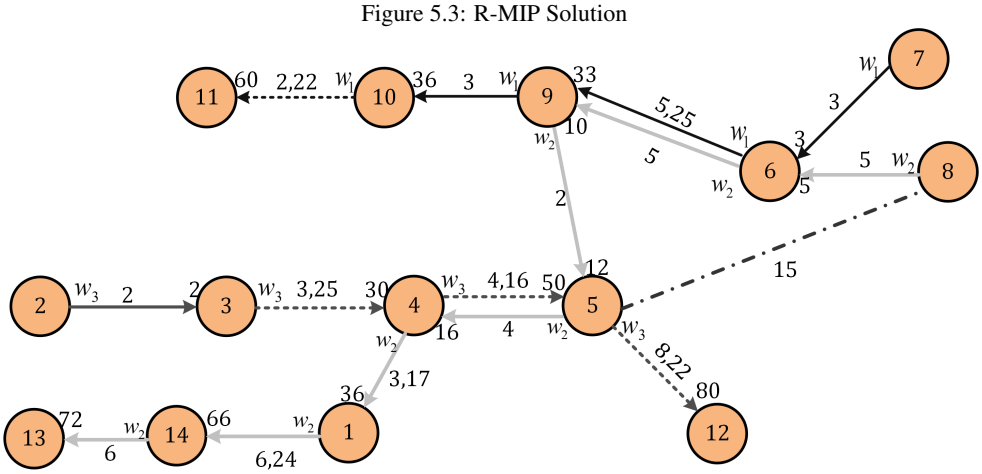


Figure 5.4: Make Routes Feasible for First Time

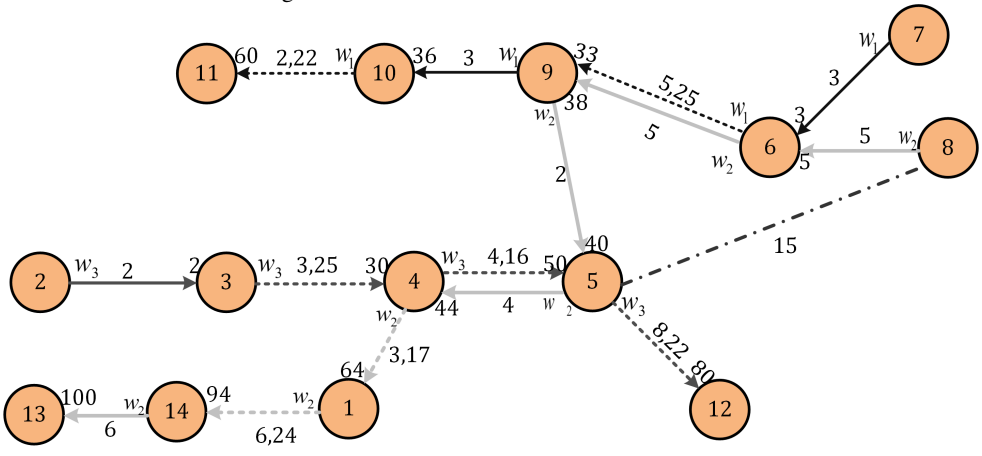


Figure 5.5: Cut and Remove Procedure

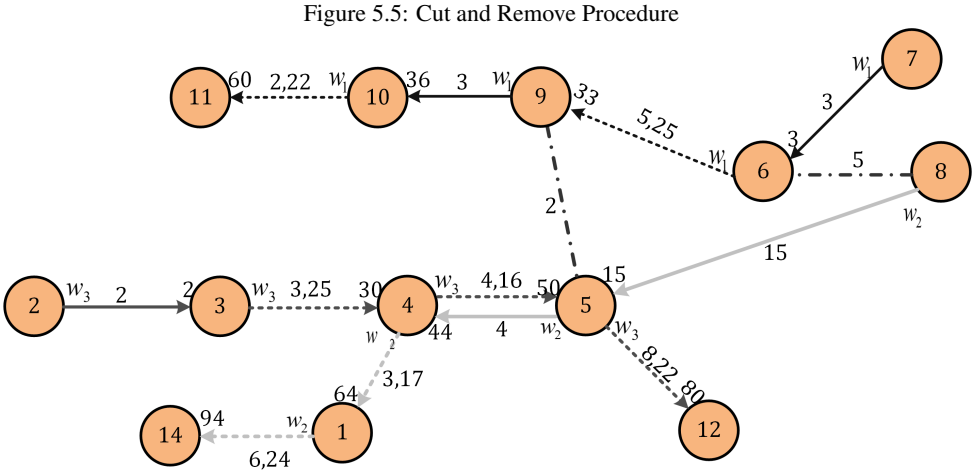


Figure 5.6: Make Routes Feasible for Second Time

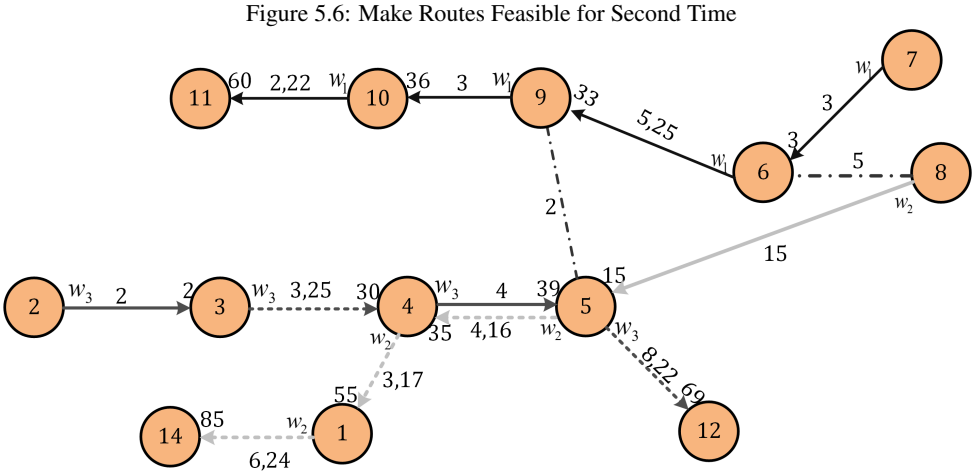
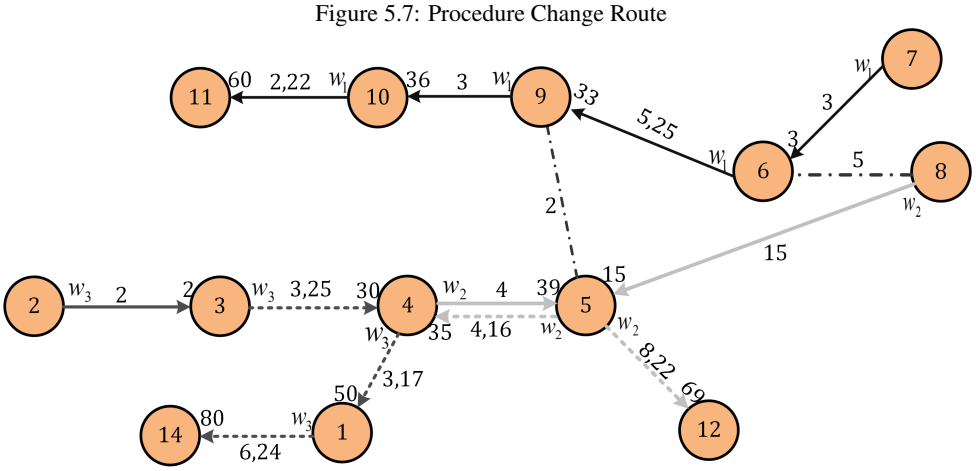


Figure 5.7: Procedure Change Route



solution derived in this example is 80, which is equal to the objective value of *R-MIP*. Hence, it is optimal to *K-ARCP*.

6. Data Generation and Computational Results.

In order to test the performance of the developed mathematical models and the heuristic algorithm, we generated two types of data: 1) based on Istanbul road network generated using ArcGIS and Google Earth, and 2) randomly generated with Euclidean distances. Furthermore, three different data sets are generated from the road network of Istanbul city. In the first data set, we generated a simplified Istanbul network with 74 nodes and 179 edges, considering the main highways and their connections to different demand centres (Figure 6.1). The second data set is extracted from a more detailed Istanbul road network with 349 nodes and 689 edges. Finally, the third data set is a detailed road network of the Southwestern region of Istanbul city with 250 nodes and 539 edges. We note that these instances are much larger than those tested in related studies in the literature (e.g. Ozdamar et al. (2014), Sahin et al. (2016)). While we obtain solutions with small optimality gaps using *RBH*, the exact model falls short of solving instances related to either of these three data sets. In order to test the performance of *E-MIP*, we generated 160 random networks with 20, 25 and 30 and 40 nodes and 1 to 4 vehicles. Furthermore, to analyse the performance of *RBH* with different number of connected components and vehicles, we tested 50 random networks with 50 nodes and 8, 10, 12, 14 and 16 connected components (10 instance for each of the number of connected components). We solved each of these instances with 1 to 6 vehicles. The edges are categorized into three different groups due to their proximity to the epicenter of the earthquake according to the scenarios predicted in JICA (2002) as high, medium and low-risk edges. The probability that an edge from the low-risk class is blocked after an earthquake is 0.1, while this probability is equal to 0.2 and 0.3 for medium and high-risk edges, respectively. Edges located at the top of the higher horizontal line in Figure 6.1 lie in the group of low-risk edges. Edges located between the two horizontal lines are in the group of medium-risk edges and the edges under the lower horizontal line are high-risk edges. If nodes of an edge are located in two different groups, we placed the particular edge in the more risky group.

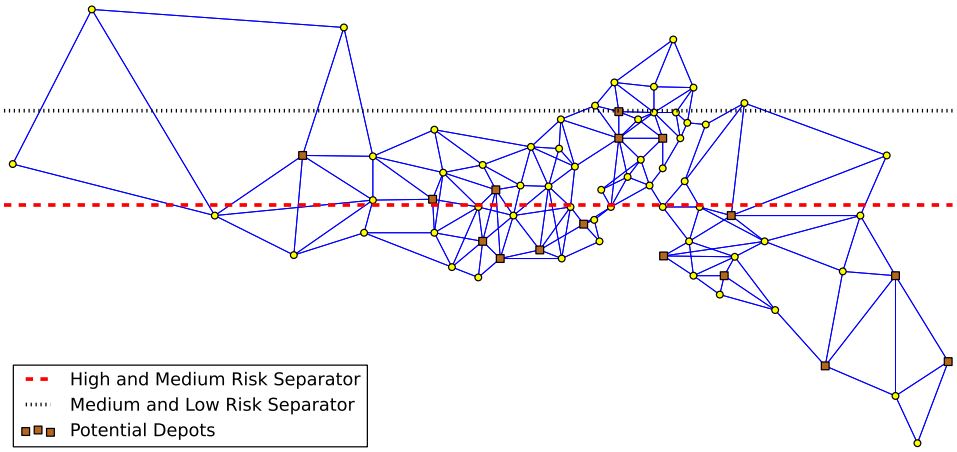


Figure 6.1: Simplified Istanbul Network

In *K-ARCP*, the vehicles should be dispatched from their depot nodes. For the simplified and detailed Istanbul road networks, we selected 16 potential depot nodes as shown in Figure 6.1. Among these 16 nodes, 5 of them are located in the Southwestern part of the Istanbul road network. In different scenarios, the depots of all the vehicles are chosen randomly among the potential depot nodes to observe their impact on the performance of the methods.

For the first three sets of data, traversal cost c_{ij} on edge (i, j) is equal to the time it takes for a work troop 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: $b_{ij} = c_{ij} \cdot X$, where X has uniform distribution between 100 and 300. 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 this connected graph.

Finally, since vehicles are teams of machinery with required personnel and equipment, it will cost significantly to increase their number. In order to avoid additional cost, we used at most $Q - 1$ vehicles in tested instances, where Q shows the number of connected components.

The results of the tested instances on the Istanbul roads networks are given in Table 6.1.

6.1. Results for the Simplified Istanbul Network.

The figure of the simplified version of the Istanbul road network is given in Figure 6.1. For this network we tested 15 instances with the number of connected components varying between 3 to 8. We examined the performance of *RBH* with 1,2,3 and 4 vehicles for each of these 15 instances.

Table 6.1: Results for the Istanbul Roads Networks

Simplified Roads Network					Detailed Roads Network					Southwest Roads Network				
Number(Q)	K	Time	Gap	LSI	Number(Q)	K	Time	Gap	LSI	Number(Q)	K	Time	Gap	LSI
1(7)	1	32.2	0	-	16(4)	1	2002.00	0	-	31(4)	1	58.66	0	-
	2	252.87	0	-		2	23.84	0	-		2	17.77	0	-
	3	160.11	0	-		3	118.50	0.0218	22.32%		3	16.75	0	-
	4	15.81	0.0159	18.34%		4	-	-	-		4		-	-
2(5)	1	4.87	0	-	17(5)	1	692.31	0	-	32(6)	1	41.97	0	-
	2	194.25	0	-		2	946.50	0	-		2	36.25	0	-
	3	59.22	0	-		3	3500.96	0	-		3	69.18	0	-
	4	8.95	0	22.84%		4	107.04	0	47.12%		4	196.93	0	-
3(6)	1	0.29	0	-	18(6)	1	90.80	0	-	33(5)	1	23.87	0	-
	2	3.68	0	-		2	105.82	0	-		2	35.84	0	-
	3	10.24	0	-		3	102.40	0	-		3	38.12	0	-
	4	15.2	0	30.64%		4	151.18	0	-		4	66.27	0	-
4(4)	1	1.41	0	-	19(4)	1	13.25	0	-	34(4)	1	42.61	0	-
	2	11.18	0	-		2	403.58	0	-		2	12.05	0	-
	3	2.26	0	-		3	43.03	0	-		3	36.63	0	-
	4	-	-	-		4	-	-	-		4	-	-	-
5(6)	1	1.9	0	-	20(4)	1	3.93	0	-	35(4)	1	13.91	0	-
	2	3.56	0	-		2	133.79	0	-		2	608.71	0	-
	3	4.21	0	-		3	32.81	0	-		3	48.24	0	-
	4	4.45	0	-		4	-	-	-		4	-	-	-
6(7)	1	2.62	0	-	21(4)	1	95.15	0	-	36(5)	1	1020.74	0	-
	2	75.62	0.0111	5.10%		2	26.67	0.004	35.39%		2	259.02	0	-
	3	665.68	0	-		3	100.15	0	-		3	35.81	0	-
	4	166.21	0.002	28.09%		4	-	-	-		4	37.35	0	-
7(5)	1	5.84	0	-	22(6)	1	104.97	0	-	37(6)	1	91.81	0	-
	2	20.12	0	-		2	3796.17	0	-		2	419.75	0	-
	3	34.12	0	-		3	547.24	0	-		3	2003.33	0	-
	4	24.12	0	-		4	444.72	0	-		4	750.01	0	-
8(4)	1	4.54	0	-	23(6)	1	19.36	0	-	38(3)	1	376.29	0	-
	2	3.42	0	-		2	2772.21	0	-		2	37.02	0.0021	25.03%
	3	6.9	0.0044	5.39%		3	1572.22	0	-		3	-	-	-
	4	-	-	-		4	1117.83	0.0451	36.93%		4	-	-	-
9(3)	1	0.37	0	-	24(5)	1	8.00	0	-	39(4)	1	6.75	0	-
	2	3.75	0.0051	21.66%		2	25.71	0	-		2	16.18	0.0018	15.26%
	3	-	-	-		3	226.60	0	-		3	22.16	0	-
	4	-	-	-		4	85.11	0	-		4	-	-	-
10(8)	1	0.93	0	-	25(4)	1	11.02	0	-	40(4)	1	9.61	0	-
	2	5.37	0	-		2	16.46	0.0041	31.95%		2	19.58	0	-
	3	6.68	0	-		3	29.09	0	-		3	30.61	0	-
	4	123.48	0.1818	35.61%		4	-	-	-		4	-	-	-
11(8)	1	5.65	0	-	26(6)	1	35.85	0	-	41(6)	1	86.32	0	-
	2	6.74	0.0185	10.99%		2	4044.42	0	-		2	27.42	0	-
	3	4.78	0.0277	15.08%		3	1095.40	0	-		3	120.22	0	-
	4	8.95	0.0043	33.25%		4	1303.07	0.1044	36.30%		4	59.25	0	-
12(6)	1	7.44	0	-	27(5)	1	11.48	0	-	42(6)	1	3.69	0	-
	2	3.42	0	8.41%		2	161.29	0	-		2	24.85	0	-
	3	15.21	0.0085	43.99%		3	54.17	0	-		3	63.54	0	-
	4	21.27	0.0063	34.71%		4	145.68	0	-		4	179.61	0	-
13(6)	1	9.91	0	-	28(4)	1	28.74	0	-	43(4)	1	18.76	0	-
	2	92.67	0	-		2	72.87	0	-		2	38.27	0.0114	18.41%
	3	258.91	0	-		3	30.30	0	-		3	43.06	0	40.64%
	4	6.42	0	-		4	-	-	-		4	-	-	-
14(8)	1	4.32	0	-	29(6)	1	31.63	0	-	44(3)	1	3.08	0	-
	2	36.18	0.0204	20.24%		2	35.78	0	-		2	112.73	0	-
	3	10.79	0	-		3	450.62	0	-		3	-	-	-
	4	9.34	0	-		4	10748.72	0	-		4	-	-	-
15(4)	1	38.14	0	-	30(6)	1	16.15	0	-	45(5)	1	10.2	0	-
	2	61.12	0	-		2	6219.57	0	-		2	16.71	0	-
	3	17.49	0	-		3	600.47	0	26.56%		3	17.43	0	-
	4	-	-	-		4	685.17	0.0253	45.54%		4	34.48	0	-

Left side of Table 6.1 shows the results for the first set of instances. While the first and the second columns in Table 6.1 show the instance number, number of connected components and the number of vehi-

cles in the corresponding instance, the third column shows the elapsed time by the Gurobi solver to solve *R-MIP* optimally for each scenario. The time is in seconds and all the scenarios were tested on an Intel Xeon E5-2643 0 CPU @ 3.3GHz 3.3GHz (two processors) computer with 32 GB RAM.

Recall that the solution to *R-MIP* gives a lower bound on the optimal objective value of *K-ARCP*. We show this lower bound as $Z^*_{R\text{-}MIP}$. *RBH* is the solution obtained after local improvement steps, that gives an upper bound for *K-ARCP*. We show the objective values of the feasibility algorithm by Z_f and the upper bound by *RBH*. Then the fifth column in Table 6.1, denoted by LSI, denotes the Local Search Improvement and gives the percentage of the improvement made by the local search algorithm introduced in Section 5.2. This value is derived as follows:

$$\text{LSI}\,(\%) = \frac{Z_f - Z_{RBH}}{Z_{RBH}}.100 \tag{6.1}$$

Note that if the optimal solution of *R-MIP* is feasible for *K-ARCP*, then we can not have any improvements. In such cases, this column is filled by '-' sign.

The fourth column in Table 6.1 shows the optimality gap for each scenario, which is a function of the upper and lower bounds as follows:

$$\text{Optimality Gap}(\%) = \frac{\text{Upper Bound - Lower Bound}}{\text{Lower Bound}}.100, \tag{6.2}$$

which is equivalent to $(Z_{RBH} - Z_{R\text{-}MIP})/(Z_{RBH})$.

In Table 6.1 we see that *RBH* obtained an optimal solution in 41 instances out of 55 of the Istanbul Simplified roads network. Note that the single vehicle problem is guaranteed to be solved optimally. Over all cases, this takes 8 seconds on the average. As the number of vehicles increases, solutions to certain instances yield optimality gaps. The optimal walks from *R-MIP* in 15 instances out of 55 tested instances were not synchronized. Hence the local search procedure is used in these 15 instances. The local search procedure improved the result in all of these instances. The average improvement among these 15 instances is 22.28%. As a result, in those instances where the solution is not optimal, the average optimality gap is 2.13%. Except for instance 10 having 4 vehicles, over all other instances the gap is at most 2.77%.

As the number of connected components change from 3 to 8, we see that usually the problem gets more difficult. While for any of these scenarios with multiple vehicles, *E-MIP* falls short to solve the problem optimally in two hours, we managed to derive good solutions using *RBH* in a very short time (12 minutes at most and 46.77 seconds on the average). We think that these times are reasonable in our context.

6.2. Detailed Istanbul Network.

As in the simplified Istanbul network case, we tested 15 different instances with 1,2,3 and 4 vehicles. The middle section of Table 6.1 gives the results for these instances. Columns are labelled as described in Section 6.1. Here we note that more than 4 vehicles may be used in post-disaster response. However, our purpose is to show the trend of performance of the algorithm. Therefore, we tested the algorithm with up to

4 vehicles.

In Table 6.1 we see that our method obtained an optimal solution in 48 instances out of 54 tested instances of the Detailed roads network. The optimal walks from *R-MIP* did not result in feasible walks in 8 instances out of the 54 tested cases. The local search improvement was 35.26 % in these 8 instances. Although in those instances where the solution is not optimal the average optimality gap is 3.41%, the average optimality gap over all instances is 0.37%. Similar to the results in Section 6.1, all instances with a single vehicle were solved optimally as timing conflict does not occur in them. As the number of vehicles changes from 2 to 4, we see that the optimality gap usually increases. While in instances with 2 or 3 vehicles, the largest gap is less than 0.5%, in instances with 4 vehicles, the average gap is 1.94%.

We see that even solving *R-MIP* is difficult with this size of the network (349 nodes and 689 edges), especially with more vehicles. The average run time to solve *R-MIP* is less than 15 minutes. Aside from instance 14 with 4 vehicles, which took almost 3 hours to solve, all other tested instances were solved in less than 2 hours. In cases where the run times are long, a reasonable approach would be to decompose the network.

6.3. Southwestern Istanbul Network.

We also tested 15 instances with 1,2,3 and 4 vehicles. The right side of Table 6.1 shows the results of the tested instances for the Southwest Istanbul road network. The columns of this part of the table are labeled as explained. In Table 6.1 for the tested instances of the southwest roads network we see that *RBH* derived an optimal solution in 47 out of 50 tested instances. In four of the tested instances, the optimal solution to *R-MIP* did not correspond to feasible walks. In these four instances, applying local search improved the result by an average of 24.83%. In instances where the solution is not optimal, the average optimality gap is only 0.51%. The maximum optimality gap is 1.14% and the average gap over all instances is 0.01%. Note that since the number of vehicles changes for a specific instance number, the depot location is chosen randomly for each vehicle. These random assignment causes positive optimality gaps in instances 9 and 13 with 2 vehicles, while there is no gap in these instances when more vehicles exist.

Instances of this data set are mostly easier to solve than instances of the detailed Istanbul road network that has a larger size. The average run time to solve *R-MIP* over all instances of the regional network is less than 3 minutes. In general, as the number of vehicles increases from 1 to 4, *R-MIP* run-time increases. The maximum elapsed time to solve *R-MIP* over all tested instances is less than 35 minutes.

6.4. Random Data Sets.

As the first set of random networks, we generated Euclidean random graphs with 20, 25, 30 and 40 nodes and 1 to 4 vehicles in order to test *E-MIP*. For each number of nodes and each number of vehicles selected, we tested 10 random instances. First, we assigned random coordinates in a $100 \cdot 100$ plane to every node. Costs, c_{ij}, are 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 distance between

nodes i and j is lower than 20 units. Since one of the primary assumptions is that the graph $G = (V, E)$ is connected, if the generated graph is not connected, we add random edges between connected components to make G connected. Depots are also chosen randomly with equal probability. According to the problem definition, $G_B = (V, E \setminus B)$ is a disconnected graph consisting of Q connected components. We impose $Q \geq 2K$ in our instances in order to increase the possibility of assigning at least one unblocking task to every vehicle. Edges are randomly added to the set B with equal probabilities to separate G_B into the desired number of connected components.

Table 6.2 shows the results comparing *E-MIP* formulation, *R-MIP* and *RBH*. While columns 1 to 6 show the properties of the generated instances for each set of nodes and vehicles, columns 7 to 10 give the results regarding *E-MIP* and *RBH*. In the first column, K shows the number of vehicles. In columns 3 and 4, $\bar{Q}$ and V show the average number of connected components and the number of nodes for every ten instances, respectively. $\bar{E}$ and $\bar{B}$ give the average number of edges and average number of blocked edges over ten instances, respectively. Note that when the number of nodes increases for a particular number of vehicles, the density of the graph increases considerably, as it can be seen in column $\bar{E}$. We set a time limit of 1 hour for all these 160 instances and tested whether *E-MIP* can solve them in this time limit or not. The *# of Solved Instances by E-MIP (in 1 hour)* column, reports this value among every ten instances with a particular number of vehicles and nodes. The column *Average E-MIP Time* gives the average time spent by *E-MIP* to solve ten instances. The last two columns show the average elapsed time by *R-MIP* to find an initial solution and the average gap over ten instances using *RBH*, respectively.

As we can see in Table 6.2, all instances with a single vehicle were solved with *E-MIP* within 3 minutes on average. As expected, *RBH* solved them all optimally in much shorter time (in less than a second on average). The elapsed time to solve *E-MIP* is considerably more than *R-MIP*, especially when the number of vehicles increases. For the case with 40 nodes and one vehicle, *E-MIP* takes 168.79 seconds on the average, whereas *R-MIP* solves them in only 0.27 seconds on the average.

In cases with two vehicles, *E-MIP* solves all the instances with 20 and 25 nodes and solves 9 instances out of 10 with 30 nodes. Meanwhile, we can see that *E-MIP* falls short of solving instances with 40 nodes in 7 instances out of 10. While *E-MIP* solves these 40 instances with two vehicles in 1205.08 seconds on the average, they were solved by *R-MIP* in very short time (the over all average among these 40 instances is less than 8 seconds). *RBH* achieved near-optimal solutions in these tested instances. The average optimality gap of *RBH* is 1.12% among all instances with two vehicles.

All the instances with three vehicles and 20 nodes were still solved optimally by *E-MIP*. However, as the density of the graph and the number of nodes increase, *E-MIP* falls short of solving instances optimally within 1 hour. While 9 and 7 out of 10 instances were solved by *E-MIP* with 25 and 30 nodes, respectively, *E-MIP* could not solve any of the instances with 3 vehicles and 40 nodes optimally, indicating its computational limits. Again, the average run time of *E-MIP* over all instances with 3 vehicles (1906.04 seconds) is significantly higher than that of *R-MIP* (29.40 seconds). *RBH* gives very good upper bounds for instances with 3 vehicles. The average optimality gap of *RBH* is almost 1.10% on the average for all the instances

with 3 vehicles.

In Table 6.2, we also see that none of the instances with 4 vehicles and 11 connected components could be solved to optimality by *E-MIP* in the given time limit of 1 hour. While *E-MIP* is able to solve instances with at most 3 vehicles and 30 nodes in a time limit of 1 hour, *R-MIP* solved all of the instances with 4 vehicles in an average time of 108.57 seconds. *RBH* still obtains very good upper bounds in instances with 4 vehicles. The average optimality gap of *RBH* is 1.11% over all instances with 4 vehicles.

Table 6.2: Results for the First Set of Random Data

K	Instance Number	$\bar{Q}$	V	$\bar{E}$	$\bar{B}$	No. of Solved Instances by E-MIP (in 1 hour)	Average E-MIP Time (seconds)	Average R-MIP Time (seconds)	Average RBH Optimality Gap
1	1 - 10	5	20	51.4	18.2	10	2.53	0.05	0%
	11 - 20	5	25	73.4	17.6	10	13.19	0.10	0%
	21 - 30	5	30	97.6	25	10	41.26	0.19	0%
	31 - 40	5	40	169	55.4	10	168.79	0.27	0%
2	41 - 50	8	20	45.4	19.4	10	88.23	0.36	0.8%
	51 - 60	8	25	71.4	31.6	10	389.17	0.87	1.1%
	61 - 70	8	30	95.6	41.8	9	1152.06	16.13	1.39%
	71 - 80	8	40	172.6	85.8	3	3190.90	13.86	1.22%
3	81 - 90	8.6	20	48.4	23.8	10	900.07	2.02	1.09%
	91 - 100	8	25	67.4	30	9	1101.43	1.42	0.14%
	101 - 110	8.6	30	90.2	35.4	7	2022.68	11.33	1.7%
	111 - 120	11	40	173.4	100.8	0	3600	102.86	1.5%
4	121 - 130	14	20	49	36.6	0	3600	18.58	2.0%
	131 - 140	14	25	77.5	53.2	0	3600	41.18	1.0%
	141 - 150	14	30	106	70.8	0	3600	70.99	0.84%
	151 - 160	14	40	158.2	96.4	0	3600	303.53	0.62%

In order to compare the performance of *RBH* with different number of connected components and vehicles, we generated 50 instances with 50 nodes using the same method as explained for Table 6.2. With each number of connected components varying from 8 to 16 (only even numbers) we generated 10 instances. We solved each of these instances with 1 to 6 vehicles. The first row of Table 6.3 gives the number of connected components. Second and third rows give the average number of edges and blocked edges, respectively, over all of the 10 instances with each specific number of components. While K gives the number of vehicles in the tested instance, the "time" column gives the average spent time spent by solver to apply *RBH* on the 10 instances with the specific number of vehicles and the "gap" column gives the corresponding average percentage gap of *RBH* over all of these instances.

In Table 6.3, we can see that generally it takes more time to solve the same sized instances with more vehicles, but there are several cases where the problem is solved in less time when more vehicle exists. We also observed that after a point, having more vehicles does not change the solution, especially with small number of components. When the number of connected components increases, on the average the time spent by solver increases. As a general pattern we can say that the number of components is an important indicator of running time. However, *RBH* tends to have smaller gaps when the number of components gets beyond 12.

We think that timing conflicts become less observable as the number of connected components get larger.

Table 6.3: Results for the Second Set of Random Data (50 nodes)

Q	8		10		12		14		16	
$\bar{E}$	114.2		114.0		114.4		119.2		109.4	
$\bar{B}$	18.2		25.4		31.2		38.4		37.6	
K	time	gap	time	gap	time	gap	time	gap	time	gap
1	0.122	0.00%	0.134	0.00 %	0.304	0.00%	1.032	0.00%	4.463	0.00%
2	1.016	0.21%	0.784	0.00%	5.385	0.37%	6.715	0.18%	18.796	0.20%
3	1.560	0.52%	2.768	0.67%	41.757	1.29%	41.875	1.05%	35.232	0.00%
4	3.560	0.26%	16.150	2.07%	104.355	2.72%	136.066	0.01%	141.400	0.35%
5	12.564	0.26%	43.282	2.21%	63.366	1.17%	257.794	0.36%	397.281	1.12%
6	5.830	0.26%	11.141	3.25%	160.868	1.25%	192.522	1.01%	262.798	0.14%

7. Conclusions and Remarks.

We defined a new arc routing problem to support planning of road clearance operations after a disaster. In this problem, we optimize the routes of K vehicles that traverse edges and open a subset of the blocked ones to reconnect the post-disaster road network in a synchronized way by accounting for necessary waiting times. We find which edges to unblock and the walks of K vehicles such that the longest walk is completed in minimum time. We call this problem Multi-Vehicle Synchronized Arc Routing for Connectivity Problem *(K-ARCP)*. We developed an exact mixed integer programming formulation called *E-MIP* that solves small sized instances, i.e. up to 30 nodes, 10 connected components and 3 vehicles within an hour. To solve larger instances, we developed a Relaxation-Based Heuristic *(RBH)* that solves a relaxed version of the main problem and implements a feasibility procedure followed by a neighbourhood search algorithm. We proved that the optimality gap of the relaxation is bounded by the number of vehicles. We also proved the feasible solution obtained from *(RBH)* is bounded by K times the optimal value. Additionally, we devised new and problem-specific local improvement moves.

We tested *E-MIP* and *RBH* on both randomly generated and Istanbul road network instances. The largest tested Istanbul instance has 349 nodes, 689 edges, 6 connected components and 4 vehicles. *RBH* showed very good performance in tested instances resulting in either an optimal or near-optimal solution in all of the tested instances. We could derive the optimal solution in more than 85% of the tested cases using *RBH* and obtained tight upper bounds in the others.

Table 7.1 shows a summary of the results presented in Section 6. While the exact mathematical model for *K-ARCP* falls short of solving even instances of the simplified Istanbul road network in reasonable time, the developed heuristic algorithm gives an optimal or near-optimal solution in less than an hour in most of the tested instances. It should be mentioned that the total run time of the algorithm mainly consists of solving *R-MIP* and other steps of the algorithm takes less than a minute even with very large instances.

For future work, methods can be developed to solve *R-MIP* more efficiently or it can be replaced by a fast constructive heuristic in larger instances. In the case that opening times are uncertain, we can use

Table 7.1: Summary of the Results on Istanbul Networks

Instance Category	Number of vehicles	Average *R-MIP* Time (seconds)	Average Optimality Gap of *RBH*
Simplified Istanbul	1	8.02	0%
	2	52.59	0.37%
	3	89.76	0.29%
	4	36.75	1.91%
Detailed Istanbul	1	77.51	0%
	2	1252.31	0.05%
	3	566.93	0.14%
	4	1643.16	1.94%
Southwestern Istanbul	1	116.64	0%
	2	112.14	0.10%
	3	195.77	0%
	4	189.12	0%

a scenario based approach to tackle the problem. In that case, we still need to solve the single scenario problem efficiently. We can also look into this problem in multiple periods. With multiple periods, we can use the same approach and update the condition of the network at the end of each period, and input it to the problem for the next period.

ACKNOWLEDGMENT

This research has been supported by TÜBİTAK grants 111M537 and 114M373. We thank Kaan Telciler and Çağan Ürküp for preparing the Istanbul networks.

References

Akbari, V., Salman, F. S., 2014. A model-based heuristic to the min max k-arc routing for connectivity problem. In: Proceedings of Student Conference on Operations Research (SCOR). pp. 76–88.

Aksu, D. T., Ozdamar, L., 2014. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. Transportation Research Part E: Logistics and Transportation Review 61, 56 – 67.

Asaly, A. N., 2013. Logistics planning for restoration of network connectivity after a disaster, thesis no: 397223. Master's thesis, Koc University.

Benavent, E., Corberan, A., Plana, I., Sanchis, J. M., 2009. Min-max k-vehicles windy rural postman problem. Networks 54 (4), 216–226.

Celik, M., Ergun, O., Keskinocak, P., 2015. The post-disaster debris clearance problem under incomplete information. Operations Research 63 (1), 65–85.

Christofides, N., Campos, V. Corberán, Á., Mota, E., 1986. An algorithm for the rural postman problem on a directed graph. Mathematical Programming Study 26, 155–166.

Coleman, D. J., Georgiadou, Y., Labonte, J., 2009. Volunteered geographic information: The nature and motivation of produsers. International Journal of Spatial Data Infrastructures Research 4 (1), 332–358.

Corberán, Á., Plana, I., Sanchis, J., 2014. The rural postman problem on directed, mixed, and windy graphs. in *Arc Routing Problems, Methods and Applications* (Á. Corberán and G. Laporte, eds.). MOS-SIAM Series on Optimization. Philadelphia, USA.

Duque, P. M., Sörensen, K., 2011. A GRASP metaheuristic to improve accessibility after a disaster. OR Spectrum 33, 525–542.

Eiselt, H. A., Gendreau, M., Laporte, G., 1995. Arc routing problems, part ii: The rural postman problem. Operations Research 43 (3), 399–414.

Faturechi, R., Miller-Hooks, E., 2015. Measuring the performance of transportation infrastructure systems in disasters: A comprehensive review. Journal of Infrastructure Systems 21 (1), 04014025 1–15.

Ghiani, G., Laporte, G., 2014. The undirected rural postman problem. in *Arc Routing Problems, Methods and Applications* (Á. Corberán and G. Laporte, eds.). MOS-SIAM Series on Optimization. Philadelphia, USA.

Hertz, A., Laporte, G., Hugo, P. N., 1999. Improvement procedures for the undirected rural postman problem. INFORMS Journal on Computing 11 (1), 53–62.

Hertz, A., Laporte, G., Mittaz, M., 2000. A tabu search heuristic for the capacitated arc routing problem. Operations Research 48 (1), 129–135.

JICA, 2002. 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.

Kasaei, M., 2015. Arc routing problems to restore connectivity of a road network, thesis no: 332214. Master's thesis, Koc University.

Liberatore, F., Ortuño, M. T., Tirado, G., Vitoriano, B., Scaparra, M. P., 2014. A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. Computers and Operations Research 42, 3–13.

Ozdamar, L., Aksu, D. T., Ergunes, B., 2014. Coordinating debris cleanup operations in post disaster road networks. Socio-Economic Planning Sciences 48 (4), 249 – 262.

Sahin, H., Kara, B. Y., Karasan, O., 2016. Debris removal during disaster response: A case for Turkey. Socio-Economic Planning Sciences 53, 49 – 59.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., 2012. Synchronized arc routing for snow plowing operations. Computers and Operations Research 39 (7), 1432 – 1440.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., 2013. The synchronized arc and node routing problem: Application to road marking. Computers and Operations Research 40 (7), 1708 – 1715.

Sohn, J., 2006. Evaluating the significance of highway network links under the flood damage: An accessibility approach. Transportation Research Part A: Policy and Practice 40 (6), 491 – 506.

Yan, S., Shih, Y. L., 2009. Optimal scheduling of emergency roadway repair and subsequent relief distribution. Computers and Operations Research 36, 2049–2065.