A Model-Based Heuristic to the Min Max K-Arc Routing for Connectivity Problem

Vahid Akbari and Sibel Salman

College of Engineering, Koc University
Istanbul, Turkey
`{vakbarighadkolaei, ssalman}@ku.edu.tr`

Abstract

We consider the post-disaster road clearing problem with the goal of restoring network connectivity in shortest time. Given a set of blocked edges in the road network, teams positioned at depot nodes are dispatched to open a subset of them that reconnects the network. After a team finishes working on an edge, others can traverse it. The problem is to find coordinated routes for the teams. We generate a feasible solution using a constructive heuristic algorithm after solving a relaxed mixed integer program. In almost 70 percent of the instances generated both randomly and from Istanbul data, the relaxation solution turned out to be feasible, i.e. optimal for the original problem.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, G.2.3 Applications

Keywords and phrases Arc Routing Problem, Mixed Integer Programming, Heuristic, Network Connectivity, Road Clearance

Digital Object Identifier 10.4230/OASIcs.SCOR.2014.76

1 Introduction

Arc routing problems have attracted the interest of researchers and have many application areas such as snow plowing, street sweeping, garbage collection, mail delivery, school bus routing and meter reading (see [8]). The problem addressed in this paper falls into the class of arc routing problems, but also contains network design and scheduling aspects. The main motivation of this section is to give an overview of arc routing problems and to introduce some problems which are related to our study.

In the *Chinese Postman Problem* (CPP), given a graph $G = (V, E)$, the problem is to determine a minimum cost closed walk, traversing each edge of G at least once. CPP can be solved in polynomial time if it is defined on an undirected or directed Network [7]. It can also be solved in polynomial time if it is defined on an mixed and even network [7], or on windy and Eulerian networks [11]. If there is a fleet of identical vehicles, say K vehicles, then the problem of finding K tours such that all the edges are covered with minimum total cost is called *K-CPP* (see [4] and [10]).

When only a subset of the edges are required to be traversed, the problem is called *Rural Postman Problem* (RPP). Eiselt et al. [8] thoroughly review studies on *RPP*. This class of problems are defined on a graph or on a multi-graph $G = (V, A)$, where V is the vertex set, A is the arc/edge set, and a non-negative cost matrix is associated with A. The graph may be directed, undirected or both (mixed). 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* [5] is closer to the problem defined in our study as it minimizes the maximum tour cost.

4th Student Conference on Operational Research (SCOR'14).
Editors: Pedro Crespo Del Granado, Martim Joyce-Moniz, and Stefan Ravizza; pp. 76–88
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Given a connected graph $G = (V, E)$ and required subsets $E_R \subseteq E$ and $V_R \subseteq V$, *General Routing Problem* (GRP) is to find a minimum cost closed walk traversing the edges in E_R and visiting the vertices in V_R at least once [9]. A large number of well-known arc routing and vehicle routing problems are special cases of the *GRP*. For instance, When $V_R = \emptyset$ and $E_R = R$ we obtain *Rural Postman Problem*. In addition, if $E_R = E$, we have the *Chinese Postman Problem*. On the other hand, if $E_R = \emptyset$, the problem reduces to *Travelling Salesman Problem*. Corberan et al. [6] provide a branch and cut algorithm for the Windy GRP that solves optimally RPP instances with around 1000 nodes and 3000 edges.

The experience of past earthquakes reveals that the roadway elements are quite vulnerable, while the damages can seriously affect the transportation of products and people. A notable recent case of these earthquakes happened in Japan in 2011. The 2011 earthquake off the Pacific coast of Tohoku was a magnitude 9.0 (Mw) undersea megathrust earthquake off the coast of Japan that occurred at 14:46 JST (05:46 UTC) on Friday 11 March 2011. As a result of this earthquake Japan's transport network suffered severe disruptions. Almost 4000 road segments, 78 bridges and 29 railway locations were reported to be damaged. Accumulated debris in the downtown of Kamaishi City, Iwate Prefecture, and a damaged arterial (national highway 45) virtually isolated the community from rescue efforts and about 76 percent of the highways in the area were closed due to damage. These type of high-impact incidents cause the network to be disconnected due to blocked roads, impeding accessibility to hospitals and critical supply and shelter locations. In the immediate disaster response phase, to facilitate emergency transportation, a critical subset of the blocked roads should be cleared or opened to restore network connectivity in shortest time. Different from studies which are looking for tours including required edges, our main concern is 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.

Recently, several studies focused on upgrading a road network or improving accessibility after a disastrous situation (e.g., [12] and [1]). To the best of our knowledge, the restoration of the roads after a disaster by routing a fleet of K vehicles in order to ensure connectivity of a network has not been addressed in the literature. In this paper, we define a new network optimization problem to address this topic. Since the problem combines arc routing and network design elements, it is called *Arc Routing for Connectivity Problem (ARCP)*, and since we are considering the case with K vehicles, we call it *K-ARCP*. The case with a single vehicle was defined and studied by Asaly and Salman [3], where a mixed integer programming (MIP) model was developed and applied to the case of Istanbul.

This paper is organized as follows: Next section gives a complete description of *K-ARC*. In the third section, we present the relaxed MIP model developed to solve *K-ARC*. We refer to this model as *R-MIP*. The fourth section gives an algorithm to extract the walk of every vehicle from the solution of *R-MIP*. In Section 5, we discuss the *feasibility algorithm* and how we make the solution of *R-MIP* feasible for *K-ARC*. Section 6 is devoted to data generation and results. We close with some final remarks in Section 7.

2 Problem Definition

We model our road network as an undirected connected graph $G = (V, A)$. There is a cost (time) c_{ij} associated with traversing all edges $(i, j) \in A : c_{ij} = c_{ji}$ and $c_{ij} > 0$. 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 $(i,j) \in B$ as b_{ij}, where $\forall (i,j) \in B, b_{ij} = b_{ji}$ and $b_{ij} > 0$. 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.

According to $(i,j) \in A : c_{ij} = c_{ji}$ and $\forall (i,j) \in B, b_{ij} = b_{ji}$, roads can be used in both directions in the disaster response phase. Since our model does not rely on these assumptions and it can handle the non-asymmetric case as well, these presumptions does not jeopardize the generality of the model.

Teams consisting of required equipment, machinery and personnel should be mobilized to clear the roads in shortest time. We refer to these teams as vehicles from now on. K vehicles are initially positioned at specified vertices. After the catastrophe, they complete a walk by working on the blocked edges assigned to them. We refer to the initial position of a vehicle as its *depot*. Note that there might be more than one vehicle positioned in a particular depot initially.

The edges that are not blocked form a graph $G_B = (V, A/B)$ with $|Q|$ components, where Q is the set of disconnected components. (Each Q is a connected sub-graph.) We are looking for a subset of blocked edges, R, such that $G_R = (V, A/B \cap R)$ is connected. In fact every subset R, is included of particular required edges which unblocking them, guarantee the connectivity of G_R. The solution identifies R and constructs a walk for each vehicle that starts at its depot. The objective is to minimize the maximum cost (time) walk among the vehicles.

Without loss of optimality, we can assume that if the walk of two vehicles includes the same blocked edge, the one that arrives first to the edge will unblock it. 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.

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:

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

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.

A simple example with two vehicles is shown in Figure 1. The blocked edges are shown with dashed lines. On the left, we see three components arising due to blocked edges. On the right, we see a feasible solution. The vehicles leave their depots, open several blocked edges and cross over some healthy edges in their walks. The walks end when the graph becomes connected.

3 A Mathematical Model for K-ARCP

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 the *R-MIP*, we may encounter two situations: 1) We do not have timing problems in our relaxed mathematical model solution and the optimal solution of *R-MIP* is in fact the optimal solution of the *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 time as necessary. The solution of the *R-MIP* gives a lower bound on the optimal solution to *K-ARCP*. An upper

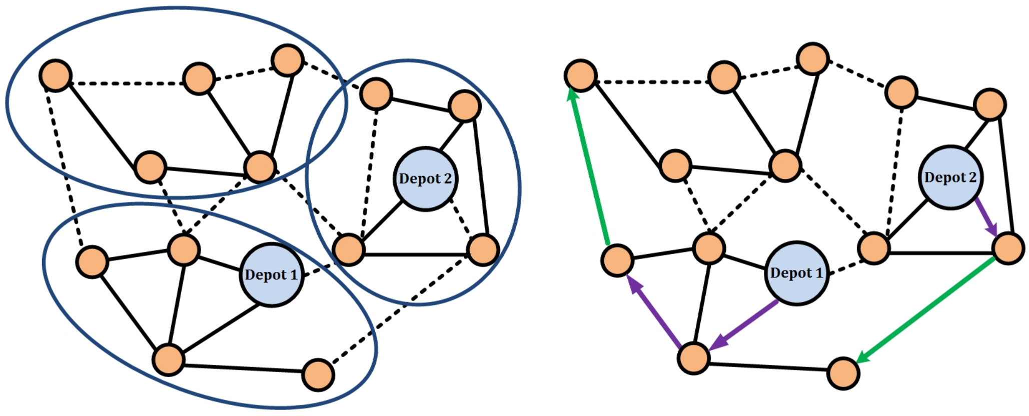

Figure 1 An illustrative example.

bound on the optimal solution of *K-ARCP* is obtained from the algorithm that constructs a feasible solution. In this way, we can derive an optimality gap for the feasible solution obtained.

Asaly [2] developed a model for the directed version of the same problem. All the time related constraints are also included as part of her model. However, since the number of variables and constraints are rapidly increasing with respect to the number of vehicles K and the size of the network, she could solve the model only for the single vehicle case, which does not require any waiting time. Asaly [2] proved that this problem is NP-hard even when a single vehicle exists. The NP-hardness of the symmetric *K-ARC* follows from this result.

The mixed integer programming (MIP) model for *K-ARCP* determines K open walks such that the disconnected components in the network are connected after unblocking a subset of the blocked edges. These walks start from depots and end in a dummy sink node, indexed as $n+1$. The model is formulated for the multi-depot and K vehicle case, where $K > 1$. The K walks altogether traverse a subset of the edges in B, denoted by R, so that the graph $G_R = (V, A/B \cap R)$ is connected. We define the decision variables and present the constraints next.

Decision Variables

$x_{ij}^k \in \{0,1\}$: indicates whether vehicle k traverses edge $(i,j) \in A$ from node i to j

$z_{ij}^k \in \{0,1\}$: indicates whether edge $(i,j) \in B$ is unblocked by vehicle k from node i to j

$f_{ij}^k \in \mathbb{N}_0$: flow of vehicle k on edge $(i,j) \in A$ from node i to j

$v_i^k \in \mathbb{N}_0$: number of times vehicle k visits node $i \in V$

Objective Function

Minimize y

subject to

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

Constraints. Let D denotes the set of depots and P_d shows the set of vehicles which are initially positioned in depot $d \in D$. Constraints (3),(4) and (5) are vehicle flow balance equations. Constraint (3) ensures that every vehicle leaves its depot and (4) and (5) enforces vehicles not to stop in any intermediate nodes. Constraint (6) forces every walk to end in the sink node. Each vehicle visits the sink node exactly once and does not get out. The latter case is satisfied by constraints (7).

$$\sum_{j \in V \cup \{(n+1)\}} (x^k_{dj} - x^k_{jd}) = 1, \quad \forall d \in D, \quad \forall k \in P_d \tag{3}$$

$$\sum_{j \in V \cup \{(n+1)\}} (x^k_{dj} - x^k_{jd}) = 0, \quad \forall d \in D, \quad \forall k \notin P_d \tag{4}$$

$$\sum_{j \in V \cup \{(n+1)\}} (x^k_{ij} - x^k_{ji}) = 0, \quad \forall k = 1, 2, \dots, K, \quad \forall i \in V \backslash D \tag{5}$$

$$\sum_{j \in V} x^k_{j(n+1)} = 1, \quad \forall k = 1, 2, \dots, K \tag{6}$$

$$x^k_{(n+1)i} = 0, \quad \forall i \in V, \quad \forall k = 1, 2, \dots, K \tag{7}$$

The following sets of constraints establish a relation between z^k_{ij} and x^k_{ij} variables. Constraint (8) shows that if a blocked edge is opened it must be traversed at least once. Constraint (9) prevents traversing a blocked edge if it is not unblocked by any of the vehicles. Let us show the set of blocked edges in the cut-sets between components with C. When a vehicle leaves its depot, it may open all the blocked edges that are in C. We show this property with (10). Since our graph is undirected, it is enough to open a blocked edge in one direction, if it is selected to be opened. This is shown by (11).

$$x^k_{ij} \geq z^k_{ij}, \quad \forall k = 1, 2, \dots, K, \quad \forall (i,j) \in B \tag{8}$$

$$x^k_{ij} \leq \sum_{\kappa=1}^{K} (z^\kappa_{ij} + z^\kappa_{ji}), \quad \forall k = 1, 2, \dots, K, \quad \forall (i,j) \in B \tag{9}$$

$$\sum_{(i,j) \in C} z^k_{ij} \leq |(i,j) \in C| \sum_{j \in V : (d,j) \in A} x^k_{dj}, \quad \forall d \in D, \quad \forall k \in P_d \tag{10}$$

$$\sum_{\kappa=1}^{K} (z^\kappa_{ij} + z^\kappa_{ji}) \leq 1, \quad \forall (i,j) \in B \tag{11}$$

In order to ensure connectivity of the walks, we define flow variables f^k_{ij} for every vehicle and for each edge that it passes through. For depot vertices, the net flow into a depot vertex is the total number of visits to all vertices except the depot. For the other vertices, it is equal to the number of visits to the corresponding node. In other words, a vehicle leaves one unit of flow each time it visits a node. Constraints (15) ensure that walks end in sink node by sending one unit of flow to the sink node. (16) prevent backward flow from the sink node to any other node.

$$\sum_{j:(i,j)\in A,\{i,j\}\in V\cup\{(n+1)\}} (f^k_{ij} - f^k_{ji}) = -v^k_i,$$
$$\forall k = 1,2,\dots,K, \quad \forall i \in V \cup \{(n+1)\} \setminus D \tag{12}$$
$$\sum_{j\in V\cup\{(n+1)\}} (f^k_{dj} - f^k_{jd}) = \sum_{i\in V\cup\{(n+1)\}\setminus\{d\}} v^k_i, \quad \forall k \in P_d, \quad \forall d \in D \tag{13}$$
$$\sum_{j:(d,j)\in A,\{i,j\}\in V\cup\{(n+1)\}} (f^k_{dj} - f^k_{jd}) = -v^k_d, \quad \forall d \in D, \quad \forall k \notin P_d \tag{14}$$
$$f^k_{(n+1)j} = 0, \quad \forall j \in V, \quad \forall k = 1,2,\dots,K \tag{15}$$
$$\sum_{j\in V} f^k_{j(n+1)} = 1, \quad \forall k = 1,2,\dots,K \tag{16}$$

Here, n shows the total number of nodes in our network. Constraints (17) do not allow flow on an edge unless it is traversed. Constraints (18) show that if an edge is traversed, then there must be a positive amount of flow passing through it.

$$f^k_{ij} \le (n-1)x^k_{ij}, \quad \forall k = 1,2,\dots,K, \quad \forall (i,j) \in A, \quad \{i,j\} \in V \cup \{(n+1)\} \tag{17}$$
$$f^k_{ij} \ge x^k_{ij}, \quad \forall k = 1,2,\dots,K, \quad \forall (i,j) \in A, \quad \{i,j\} \in V \cup \{(n+1)\} \tag{18}$$

A vertex is visited if and only if an edge entering that vertex is traversed, by constraint (19). If we cross a particular vertex more than one time, we should go to open another blocked edge that is in C. Constraint (20) enforces this.

$$\sum_{j:(i,j)\in A} x^k_{ji} = v^k_i, \quad \forall k = 1,2,\dots,K, \quad \forall i \in V \cup \{(n+1)\} \tag{19}$$
$$v^k_i = \sum_{(i,j)\in C} z^k_{ij}, \quad \forall k = 1,2,\dots,K, \quad \forall i \in V \cup \{(n+1)\} \tag{20}$$

Constraints (21) enforce connectivity of the graph. It is a sub-tour elimination constraint.

$$\sum_{\kappa=1}^{K} \sum_{(i,j)\in\delta(s)} z^\kappa_{ij} \ge 1, \quad \forall S \subset V, S \ne \emptyset \tag{21}$$

4 Walk Extraction Algorithm

After solving *R-MIP* we obtain values of x^k_{ij} and f^k_{ij} and v^k_i but we do not know about the order of edges to be visited by each vehicle. What we know is that whether a vehicle crosses an edge in a particular direction or not.

We define the *Walk Extraction Algorithm* for k^{th} vehicle as follows.

Walk Extraction Algorithm

- **step 1:** For the k^{th} vehicle if $\exists i \in V : v^k_i \ge 2$ go to step 3 and otherwise go to step 2.
- **step 2:** The decreasing order of f^k_{ij} values gives us the walk.
- **step 3:** In the graph $\Gamma = (V, \hat{A})$ such that V is the set of all vertices as before and $\hat{A} = \{(i,j) | x^k_{ij} = 1\}$; find the shortest path between P_d and the sink node.
- **step 4:** Using a *cycle detection* method, find all the loops in Γ.

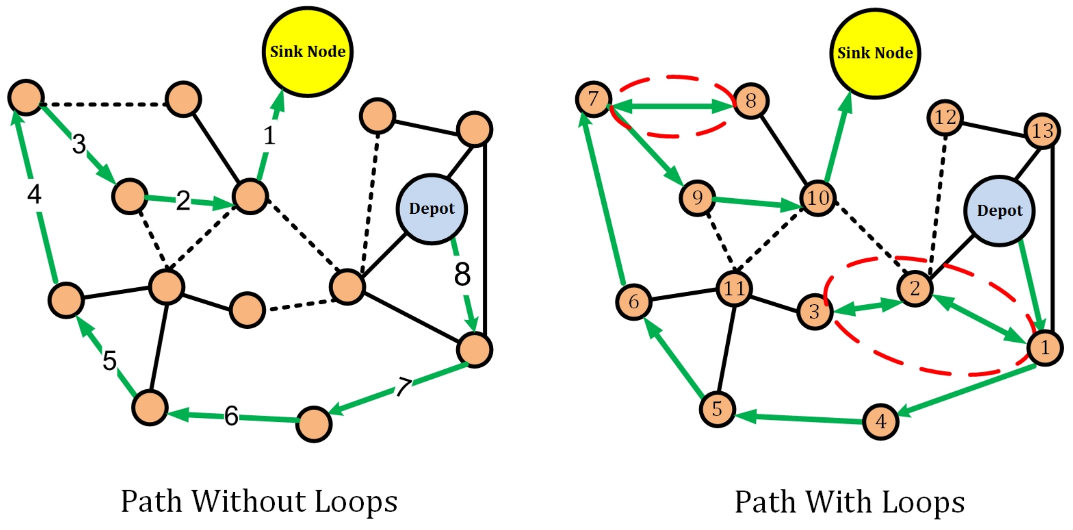

Figure 2 Samples of paths with and without loops.

- **step 5:** Go through nodes in the shortest path obtained in step 3 and if any of the nodes are in one of the loops derived from step 4; add the loop to the shortest path and remove the loop from the set of loops and then start step 5 again. If the size of loops set is 0, the path is complete.

There is an important property with the walks; they either contain loops or not. In the walk of the k^{th} vehicle a loop occurs if $\exists i \in V : v_i^k \geq 2$. It means that starting from node i, vehicle k visited a set of edges and came back to node i again. On the other hand, if in the k^{th} vehicle's walk $v_i^k \leq 1$ for all $i \in V$ there is no loop in it's walk. Step 1 determines whether the walk includes walks or not. Second step relation follows by the structure of flow variables. A sample of walk without loops is shown in the left side of Figure 2. Flow variable values on the edges are determining the path. In the right side network of Figure 2, a *cycle-included* walk is shown. In this case, the result of third step is: $Start \Rightarrow 1 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 7 \Rightarrow 9 \Rightarrow 10 \Rightarrow End$. For step 4, we used $simple-cycles(G)$ function from $NetworkX$ library in $Python$. For the given sample the loops are: $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 2 \Rightarrow 1$ and $7 \Rightarrow 8 \Rightarrow 7$. Due to step 5 the output of the algorithm is as follows: $Start \Rightarrow 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 2 \Rightarrow 1 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 7 \Rightarrow 8 \Rightarrow 7 \Rightarrow 9 \Rightarrow 10 \Rightarrow End$.

▶ **Lemma 1** (Walk Extraction Algorithm). *Given the values of x_{ij}^k and f_{ij}^k and v_i^k $\forall k = 1, \ldots, K$, we can find the walk of each vehicle with the Walk Extraction Algorithm.*

Proof. The proof follows by the structure of the variables in the mathematical model and observations discussed in the algorithm about the properties of the walks. ◀

▶ **Lemma 2** (*R-MIP* Solution). *For $W_k, k = 1, \ldots, k$ showing the walk of the k^{th} vehicle and $R = B \cap W_k$, when R is added to G_B, we get a connected graph. Moreover, $T_R \leq Z^*_{K-ARCP}$ where T_R shows the optimal objective value of* R-MIP *and Z^*_{K-ARCP} denotes the optimal objective value of* K-ARCP*.*

Proof. The first part follows by the structure of the mathematical model and problem definition. Constraint 21 enforces to open at least a blocked edge between two separated components for every choice of subcomponents. This will result in finding $G_R = (V, A/B \cap R)$

Table 1 Order of edges visited by vehicles (1 to k).

Walk	Path of the walks	Finishing Time of the walks
W_1	$w_{1,1} \Rightarrow w_{1,2} \Rightarrow w_{1,3} \Rightarrow \ldots \Rightarrow w_{1,n(1)}$	$t_{1,1} \Rightarrow t_{1,2} \Rightarrow t_{1,3} \Rightarrow \ldots \Rightarrow t_{1,n(1)}$
W_2	$w_{2,1} \Rightarrow w_{2,2} \Rightarrow w_{2,3} \Rightarrow \ldots \Rightarrow w_{2,n(2)}$	$t_{2,1} \Rightarrow t_{2,2} \Rightarrow t_{2,3} \Rightarrow \ldots \Rightarrow t_{2,n(2)}$
.	.	.
.	.	.
.	.	.
W_k	$w_{k,1} \Rightarrow w_{k,2} \Rightarrow w_{k,3} \Rightarrow \ldots \Rightarrow w_{k,n(k)}$	$t_{k,1} \Rightarrow t_{k,2} \Rightarrow t_{k,3} \Rightarrow \ldots \Rightarrow t_{k,n(k)}$

Table 2 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	$B_{k,1} \Rightarrow B_{k,2} \Rightarrow B_{k,3} \Rightarrow \ldots \Rightarrow B_{k,m(k)}$	$\tau_{k,1} \Rightarrow \tau_{k,2} \Rightarrow \tau_{k,3} \Rightarrow \ldots \Rightarrow \tau_{k,m(k)}$

such that G_R is connected. The second part follows by the fact that we are ignoring timing conflicts in *R-MIP*. In *R-MIP* we get a wider feasible region by ignoring timing variables and constraints, which will result in better optimal values. Disregarding time related elements will result in non feasible solutions in some cases. This infeasible cases occur when a vehicle is crossing a blocked edge since it is going to be unblocked by another vehicle. Since we are relaxing time constraints, the opener vehicle might unblock the blocked edge after the other vehicle crosses it in sense of time, which is not acceptable in *K-ARCP*. ◀

5 Feasibility Algorithm

Since *R-MIP* does not have time-related elements, our solution might not be feasible for *K-ARCP*. We can derive a feasible solution by modifying the solution obtained from *R-MIP*. We determine those edges with timing conflict and then we shift the time of the second vehicle that is crossing the blocked edge to derive a feasible solution.

As explained in the *Walk Extraction Algorithm* we can derive walks of each vehicle from the output of *R-MIP*. With the result obtained from the *Walk Extraction Algorithm* we can form Table 1. It includes the path of every vehicle and the corresponding time for crossing every edge in every path. $w_{i,j}$ shows j^{th} edge crossed by vehicle i and $t_{i,j}$ is the corresponding time with it. With the information in Table 1, $T_R = \max_k (t_{k,n(k)})$.

In Table 2, we only consider the edges in B that are traversed with different vehicles and the corresponding time of crossing them. Here $B_{i,j}$ shows j^{th} edge in B that has been crossed by vehicle i and $\tau_{i,j}$ is the time when traversing the edges is completed.

Note that an edge of set B may be repeated in the walk of a vehicle, or it may appears in the walks of more than one vehicles. During the algorithm, as we process the edges given in Table 2, we shift the finishing time of those edges that are effected and remove the processed edges from Table 2.

With Table 2 in hand we can write our feasibility algorithm as follows.

Feasibility Algorithm

- **Step 1:** Choose $\psi = min\{\tau_{i,j}\}$ over all $i = 1, ..., k, j = 1, ..., n(k)$.
  If $\psi = min\{\tau_{i,j}\} = max\{\tau_{i,j}\}$ paths are synchronized. Otherwise go to step 2.
- **Step 2:** Determine vehicle number $(= \kappa)$ and blocked edge $= (i, j)$ associated with ψ. (Go to one of the following cases considering z^{κ}_{ij} and z^{κ}_{ji} values:)
- **Case 1: If $z^{\kappa}_{ij} = 1$: "No Change Case"**
  "Apply following change:"
  Update Table 2 by removing all $B_{i,j}$s and $B_{j,i}$s and their corresponding time from it and go back to step 1.
- **Case 2: If $z^{\kappa}_{ij} = 0$ but $z^{\kappa}_{ji} = 1$: "Backward Opener Case"**
  "Apply following changes:"

  1. For $\forall \tau^{\kappa}_{\alpha,\beta} : \tau^{\kappa}_{i,j} \leq \tau^{\kappa}_{\alpha,\beta} \leq \tau^{\kappa}_{j,i}$, shift $\tau^{\kappa}_{\alpha,\beta}$ values as follows: $\tau^{\kappa}_{\alpha,\beta(new)} = \tau^{\kappa}_{\alpha,\beta} + b_{ij}$ where b_{ij} is the extra cost associated with opening edges (i, j).
  2. If (i, j) or (j, i) is in the walk of vehicle η such that $\eta \neq \kappa$ and $\tau^{\eta}_{i,j} \leq \tau^{\kappa}_{i,j}$ or $\tau^{\eta}_{j,i} \leq \tau^{\kappa}_{i,j}$ for $\forall \tau^{\eta}_{\alpha,\beta} : \tau^{\kappa}_{i,j} \leq \tau^{\eta}_{\alpha,\beta}$, shift $\tau^{\eta}_{\alpha,\beta}$ values as following: $\tau^{\eta}_{\alpha,\beta(new)} = \tau^{\eta}_{\alpha,\beta} + \tau^{\kappa}_{i,j} - \tau^{\eta}_{i,j} + c_{ij}$, where c_{ij} is the time or cost of crossing edge (i, j) after fixing it.
  3. Update Table 2 by removing all $B_{i,j}$s and $B_{j,i}$s and their corresponding time from it and go back to Step 1.

- **Case 3: If $z^{\kappa}_{ij} = 0$ and $z^{\kappa}_{ji} = 0$: "Swap opener vehicle Case"**
  Determine vehicle ρ such that $z^{\rho}_{ij} + z^{\rho}_{ji} = 1$ meaning vehicle ρ is opening edge (i, j). (For simplicity of notation let us assume vehicle ρ is opening edge (i, j) at time τ_{ρ}.)
  "Apply following changes:"

  1. (Making changes in the walk of vehicle ρ) $\forall \tau^{\rho}_{\alpha,\beta} : \tau_{\rho} \leq \tau^{\rho}_{\alpha,\beta}$, shift $\tau^{\rho}_{\alpha,\beta}$ values as following: $\tau^{\rho}_{\alpha,\beta(new)} = \tau^{\rho}_{\alpha,\beta} - \tau_{\rho} + max\{\tau^{\kappa}_{i,j} + c_{ij}, \tau_{\rho} - b_{ij}\}$.
  2. (Making changes in the walk of vehicle κ) vehicle κ is now the opener of this blocked edge. So $\forall \tau^{\kappa}_{\alpha,\beta} : \tau^{\kappa}_{i,j} \leq \tau^{\kappa}_{\alpha,\beta}$, shift $\tau^{\kappa}_{\alpha,\beta}$ values as following : $\tau^{\kappa}_{\alpha,\beta(new)} = \tau^{\kappa}_{\alpha,\beta} + b_{ij}$.
  3. If (i, j) or (j, i) is in the walk of vehicle η such that $\eta \neq \kappa \neq \rho$ and $\tau^{\eta}_{i,j} \leq \tau^{\kappa}_{i,j}$ or $\tau^{\eta}_{j,i} \leq \tau^{\kappa}_{i,j}$ for $\forall \tau^{\eta}_{\alpha,\beta} : \tau^{\kappa}_{i,j} \leq \tau^{\eta}_{\alpha,\beta}$, shift $\tau^{\eta}_{\alpha,\beta}$ values as following: $\tau^{\eta}_{\alpha,\beta(new)} = \tau^{\eta}_{\alpha,\beta} + \tau^{\kappa}_{i,j} - \tau^{\eta}_{i,j} + c_{ij}$.
  4. Update Table 2 by removing all $B_{i,j}$s and $B_{j,i}$s and their corresponding time from it and go back to Step 1.

"No change" case: Since vehicle κ is opening edge (i, j) in its first visit there is no timing conflict in this case.

"Backward opener" case: In this case, same vehicle opens edge (i, j) in the (j, i) direction in its second pass over it. In the 2nd change of the "Backward opener" case we are considering that vehicle η could finish his walk on edge (i, j) at any possible time and particularly $\tau^{\eta}_{i,j}$ or $\tau^{\eta}_{j,i}$, but now the earliest time it can finish its walk across edge (i, j) is $\tau^{\kappa}_{i,j} + c_{ij}$.

"Swap opener vehicle" case: In this case, the vehicle κ is crossing edge (i, j) without unblocking it, since in the solution of *R-MIP*, ρ is the opener vehicle of edge (i, j), i.e, $z^{\rho}_{ij} + z^{\rho}_{ji} = 1$. Since ρ is opening this edge after vehicle κ crosses it, timing conflict happens. For the shifting procedure we should consider vehicle κ, ρ and all other vehicles k, such that $x^{k}_{ij} + x^{k}_{ji} \geq 1$ separately.

Table 3 Distance limit for graph generation.

Number of nodes	100	75	50	25
Distance	10	15	20	25

▶ **Lemma 3** (Feasibility Algorithm). *Given k walks through $G = (V, A)$; $w_{i,1} \Rightarrow w_{i,2} \Rightarrow w_{i,3} \Rightarrow \ldots \Rightarrow w_{i,n(i)}$ where $i \in \{1, 2, \ldots, k\}$, and their corresponding time ; $t_{i,1} \Rightarrow t_{i,2} \Rightarrow t_{i,3} \Rightarrow \ldots \Rightarrow t_{i,n(i)}$, by applying feasibility algorithm, we can modify the non-feasible walks to a feasible solution for* K-ARCP.

Proof. The proof follows by the properties of the algorithm. We keep track of the walks like a simulation system and assign required waiting times whenever it is necessary. With this procedure we prevent timing conflicts among vehicles on blocked edges. ◀

6 Results

In order to test the performance of *R-MIP* and the feasibility algorithm, we generated two types of data.

6.1 Randomly generated data set

We generated Euclidean random graphs with 25, 50, 75 and 100 nodes. First, we assigned random coordinates in a 100 × 100 plane to every node. Costs $(= c_{ij})$ on edges are equal to the Euclidean distances. The extra cost $(= b_{ij})$ on edge (i, j) is generated according to: $(b_{ij} = c_{ij} \times U(10, 30))$ where $U(10, 30)$ is uniform distribution between 10 and 30. In each case with different number of nodes, an edge (i, j) exists if distance between nodes i and j is lower than the distance limits given in Table 3. But, as we know, one of our primary assumptions is that our graph is connected in the beginning. In case the generated graph is not connected, we add some random edges between disconnected components to make it connected. Depots are also chosen randomly among all the nodes in the graph and we assumed every node has the potential to be a depot. According to the problem definition $G_B = (V, A/B)$ is a disconnected graph consisting of $|Q|$ separated components. We also impose $|Q| \geq K + 1$ in our instances, to increase the possibility of assigning at least one unblocking task to every vehicle. Edges are randomly added to the set B one by one. This procedure stops when $|Q| \geq K + 1$.

We had instances with 100, 75, 50 and 25 nodes. With 100 and 75 nodes, we tried 1, 2, 3 and 4 vehicles and for the cases with 50 and 25 nodes, we tried 1 and 2 vehicles. For every case, we had 5 different instances. For example, we generated 5 instances with the 75 nodes and 2 vehicles, and so on. In order to check the capabilities of our model, we tested 5 instances with 4 vehicles and 1000 nodes positioned in an $10,000 \times 10,000$ plane with edge existence distance of 10 units.

As we see in Table 4, in almost 70% of the instances we derived the optimal solution of *K-ARCP* by solving *R-MIP*. In the other cases we found good lower and upper bounds for the optimal solution of *K-ARCP*. When the number of vehicles gets higher, the possibility of occurrence of a timing conflict is higher and we get wider boundaries for our optimal solution. In the case with one vehicle, we can always get the optimal solution of *K-ARCP* with solving *R-MIP*, since timing conflict requires at least two vehicles.

In Table 4 and 5, the maximal optimality gap shows the maximum possible gap from optimal solution to either of the lower bound or upper bound solutions.

Table 4 Results with randomly generated data according to method 1.

Number of Nodes	Number of Vehicles	Maximal Optimality Gap	Exact Optimal Solution
1000	4	4.1 %	4 out of 5
100	1	0 %	5 out of 5
	2	2.6 %	4 out of 5
	3	3.2 %	3 out of 5
	4	8.4 %	2 out of 5
75	1	0%	5 out of 5
	2	5.2 %	2 out of 5
	3	5.8 %	2 out of 5
	4	5.0 %	2 out of 5
50	1	0%	4 out of 5
	2	2.4 %	5 out of 5
25	1	0 %	5 out of 5
	2	12.4 %	5 out of 5

We should mention that all of these instances were tested on an Intel® Xeon® E5-2643 0 CPU @ 3.3 GHz (two processors) with 32 GB of RAM device and except for the cases with 1000 nodes, all instances were solved in less than 1 minute. However, for the cases with 1000 nodes, all the instances were solved in at most 13 minutes. More than 80% of these run times were from solving *R-MIP*. Even with very large instances the *feasibility algorithm* takes less than a minute.

6.2 Istanbul data set

We used the network of Istanbul city given in [3]. Istanbul's network is modelled by 74 nodes and 316 edges. The edges are categorized into 3 different groups due to their proximity to the epicentre of the earthquake scenario: as high, medium and low risk edges. The probability to lose an edge after an earthquake, from low risk class is 0.3, and this probability is equal to 0.4 and 0.5 for medium and high risk edges, respectively. After each run we verified if the number of disconnected components, $|Q|$, is higher than the number of vehicles, K, or not. If $|Q| \geq K$ it means that our current number of vehicles is suitable for our problem, otherwise, we are keeping too many vehicles for our network. In the latter case, we do not solve the problem and we refer to them as NS in Table 5. In Table 5, exact optimal solution for every instance shows if the optimal solution of *R-MIP* and *K-ARC* are equal to each other or not. The fifth column in Table 5 gives the optimal objective value for every instance. This optimal value is the length of the longest walk among all the vehicles in hours. For Istanbul's network, we tested cases with 1 up to 4 vehicles. In each case, we generated 5 random problems. The results showed that keeping more than 3 vehicles for the probabilities assigned to different categories of edge risks is not logical, since all the 5 problems generated for this case had less than 5 components. Meanwhile, the probabilities that we assigned to losing edges after a disaster is quite pessimistic, which means 3 vehicles would be enough to support Istanbul's network in an expected earthquake.

For the case of 3 vehicles, in 2 out of 5 problems generated, we had 4 or more disconnected components and with the case of 2 vehicles, in 4 out of 5, our network was separated to more than 3 components. Table 5 shows all the results related to our instances. Table 5, shows that we obtained the optimal solution to *K-ARCP* by solving *R-MIP* in almost 60% of the

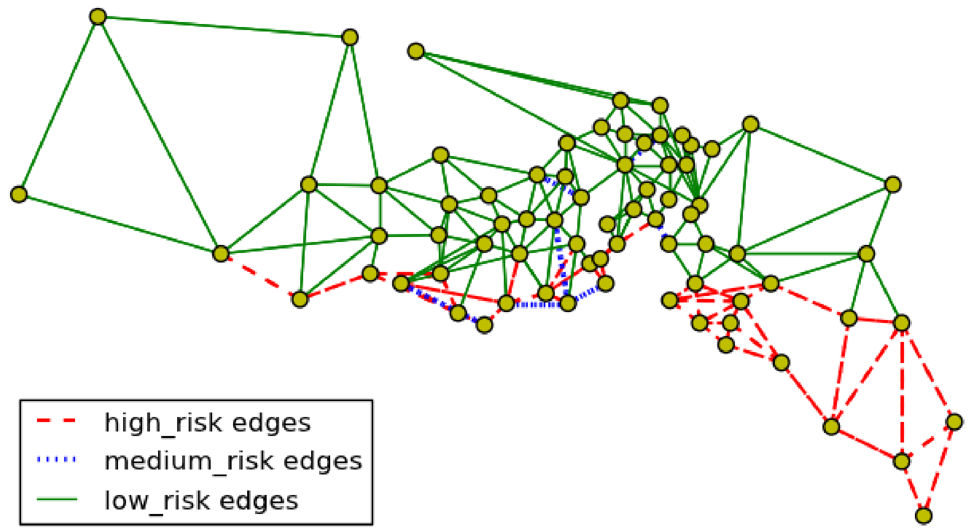

Figure 3 Categorization of Istanbul's roads.

solved instances. As we see when the number of vehicles gets higher the timing conflict cases gets more as we expected. All these instances were solved in less than 12 minutes. For the cases with one vehicle these run times were less than a minute but as the number of vehicles gets higher, it affects the run time accordingly.

7 Conclusion

We defined a new arc routing problem with the motivation of planning road clearance operations after a disaster. In this problem, we optimize the routes of K vehicles that traverse arcs and open a subset of blocked ones to reconnect the post disaster road network. We find which edges to unblock and walks of k vehicles, such that the longest walk is completed in minimum time. We call this problem *min max K-arc routing for connectivity problem(K-ARCP)*. We developed a heuristic approach that converts the solution of a relaxed MIP model to a feasible solution.

In spite of the difficulty of *K-ARCP*, we could derive the optimal solution in more than 60% of the tested cases by our approach. In fact for the case of one vehicle, our mathematical model solves *K-ARCP* exactly in all the instances. When the number of vehicles is small (at most 4), and number of nodes is large (≥ 100), the possibility of timing conflicts gets lower and results in better bounds for the optimal solution. On the other hand, when we put too many vehicles in a network with modest number of nodes, we may not use all the vehicles. In reality, there would be a very high cost associated with providing vehicles, since these vehicles in our problem are teams of machinery with required personnel and equipment. Therefore, fewer number of vehicles would be used for relatively smaller networks which will lower the possibility of time conflicts in the relaxed model's solution.

Exact methods to solve *K-ARCP* for small instances can be explored as future work. Other objectives that aim to connect the network partially with a given time limit could also be considered.

Table 5 Results of Istanbul's Network.

Number of Vehicles	Instance Number	Maximal Optimality Gap	Exact Optimal Solution	Optimal Value of *R-MIP* (in hours)
1	1	0 %	Yes	11.72
	2	0 %	Yes	6.99
	3	0 %	Yes	10.26
	4	0 %	Yes	12.15
	5	0 %	Yes	5.47
2	1	0%	Yes	5.83
	2	NS	–	–
	3	8 %	No	12.383
	4	10 %	No	17.05
	5	0 %	Yes	19.23
3	1	NS	–	–
	2	22 %	No	5.39
	3	NS	–	–
	4	25 %	No	4.62
	5	NS	–	-
4	1	NS	–	–
	2	NS	–	–
	3	NS	–	–
	4	NS	–	–
	5	NS	–	–

References

1 D. T. Aksu and L. Ozdamar. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61(0):56–67, 2014.

2 A. N. Asaly. Logistics planning for restoration of network connectivity after a disaster. Master's thesis, Koc University, 2013.

3 A. N. Asaly and F. S. Salman. *Global Logistics, New Directions in Logistics Management*, chapter Arc Selection and Routing for Restoration of Network Connectivity after a Disaster. Taylor and Francis, 2014.

4 A. Assad, W. Pearn, and B. Golden. The capacitated Chinese postman problem: Lower bounds and solvable cases. *American Journal of Mathematical and Management Science*, 7:63–88, 1987.

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

6 A. Corberán, I. Plana, and J. M. Sanchis. A branch and cut algorithm for the windy general routing problem and special cases. *Networks*, 49(4):245–257, 2007.

7 J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman problems. *Mathematical Programming*, 5:88–124, 1973.

8 H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, Part II: The rural postman problem. *Operations Research*, 43(3):399–414, 1995.

9 C. S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

10 W. L. Pearn. Solvable cases of the k-person Chinese postman problem. *Operations Research Letters*, 16:241–244, 1994.

11 Z. Win. *Contributions to Routing Problems.* PhD thesis, University of Augsburg, 1987.

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