A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration

Citation for published version:
Moreno, A, Munari, P & Alem, D 2019, 'A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration', *European Journal of Operational Research*, vol. 257, no. 1, pp. 16-34. https://doi.org/10.1016/j.ejor.2018.11.004

Digital Object Identifier (DOI):
10.1016/j.ejor.2018.11.004

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
European Journal of Operational Research

Publisher Rights Statement:
javascript:void(0);

A Branch-and-Benders-Cut Algorithm for the Crew Scheduling and Routing Problem in Road Restoration

Alfredo Moreno[a], Pedro Munari[a,*], Douglas Alem[b]

[a] *Production Engineering Department, Federal University of São Carlos, Rod. Washington Luís Km 235, CEP 13565-905, São Carlos, Brazil*
[b] *University of Edinburgh Business School, Management Science and Business Economics Group, 29 Buccleuch Place, EH89JS, Edinburgh, UK*

Abstract

Extreme events such as disasters cause partial or total disruption of basic services such as water, energy, communication and transportation. In particular, roads can be damaged or blocked by debris, thereby obstructing access to certain affected areas. Thus, restoration of the damaged roads is necessary to evacuate victims and distribute emergency commodities to relief centers or affected areas. The Crew Scheduling and Routing Problem (CSRP) addresses decisions in post-disaster situations with the aim of minimizing the time that affected areas remain inaccessible. The integration of crew scheduling and routing decisions makes this problem too complicated to be effectively solved for practical instances using mixed integer programming (MIP) formulations recently proposed in the literature. Therefore, we propose a branch-and-Benders-cut (BBC) algorithm that decomposes the integrated problem into a master problem (MP) with scheduling decisions and subproblems with routing decisions. Computational tests based on instances from the literature show that the proposed exact method improves the results of MIP formulations and other exact and metaheuristic methods proposed in literature. The BBC algorithm provides feasible solutions and optimality gaps for instances that thus far have not been possible to solve by exact methods in the literature.

Keywords: Combinatorial optimization, Benders decomposition, Branch-and-cut, Crew scheduling and routing, Road restoration.

1. Introduction

Infrastructure systems that provide essential public services such as water, energy, telecommunications, and transportation are commonly disrupted after extreme events. Floods, landslides, and earthquakes are examples of natural hazards that might damage the overall network composed by roads, bridges, and tunnels, thereby contributing to the interruption of services and logistics activities. In this context, restoring transportation infrastructure is crucial to carry out an effective short-term response, which includes the evacuation of victims from affected areas

*Corresponding author
Email addresses: `alfredmorenoarteaga@gmail.com` (Alfredo Moreno), `munari@dep.ufscar.br` (Pedro Munari), `douglas.alem@ed.ac.uk` (Douglas Alem)

to temporary shelters and the distribution of relief aid. The restoration of transportation infrastructure after extreme events is referred to in the literature as the *road restoration problem* (Tuzun Aksu and Ozdamar, 2014).

Road restoration involves certain decisions that must be taken quickly, such as the selection of the roads to restore and the scheduling and routing of the crews that will perform the repair activities. In this paper, we are particularly interested in a variant studied in Maya-Duque et al. (2016) that explicitly considers the complex interdependence between scheduling and routing decisions for a single crew, hereafter called the Crew Scheduling and Routing Problem (CSRP) in road restoration. We consider that a damaged road can have one or more damaged points (damaged nodes), as may occur in real cases, especially on long highways. The scheduling decisions define the sequence in which the damaged nodes in the network will be visited by the crew. The routing decisions determine the paths/routes to be used by the crew to visit and repair the damaged nodes. In this variant, a path is usually a sequence of nodes and arcs used by the crew to travel from one damaged point to another, while a route is a sequence of paths that ends at the depot after repairing all the damaged nodes. The objective is to restore the damaged nodes in the network as soon as possible, because they are necessary to define paths connecting a source node to demand nodes that require humanitarian assistance.

The design of crew routes is challenging because damaged nodes can obstruct access to other nodes of the network and also damaged roads are not traversable unless they are completely repaired first. The traversable roads include those that were not damaged and the repaired ones. Then, the number of paths that are feasible at a specific moment depends on which nodes are damaged at that moment, which in turn depends on the scheduling decisions. In addition, without considering routing decisions simultaneously, the damaged nodes that are not accessible at a given moment might be selected first in the schedule, making the schedule infeasible in practice. Furthermore, the shortest paths between damaged nodes, if they exist, change dynamically during the restoration according to the schedule.

The integration of the main decisions that emerge in road restoration has been addressed by other authors in the literature (Çelik, 2016). Particularly, the CSRP has been tackled recently via the proposition of MIP and dynamic programming models (Maya-Duque et al., 2016). However, such models have proven to be intractable and failed to solve even small instances. Hence, the authors have devised heuristic methods (Maya-Duque et al., 2016) to obtain feasible solutions for the instances of the CSRP. As usual, the main drawback of heuristic approaches is that they do not provide optimality guarantees or any information on the quality of the solutions. Furthermore, a heuristic can stagnate in locally sub-optimal solutions.

In this paper, we develop an exact algorithm based on Benders decomposition for the CSRP. The algorithm exploits the fact that when the scheduling decisions are fixed, the routing decisions become a set of shortest-path subproblems. To solve the subproblems, we propose specialized algorithms based on Dijkstra's shortest-path algorithm. Hence, we consider a master problem (MP) with scheduling decisions and subproblems with the remaining routing decisions. The resulting MP obtained from the Benders decomposition is solved by a single search tree, exploring the generation of cuts inside the tree. This strategy has been recently referred to as Branch-

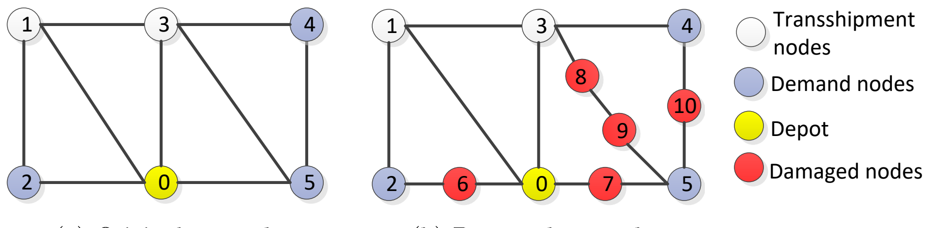
and-Benders-Cut (BBC) (Gendron et al., 2016; Errico et al., 2017) and has been shown to be more effective than the standard Benders approach, which solves a mixed-integer MP at each iteration. We are not aware of any other decomposition-based exact algorithm proposed for the CSRP or related variants.

Due to the discrete subproblems, standard duality theory cannot be applied to derive cuts; therefore, we propose different types of lower-bounding functions and combinatorial Benders cuts (Laporte and Louveaux, 1993) based on particular characteristics of the CSRP. Combinatorial Benders cuts cut off infeasible solutions in the MP, while lower-bounding functions set lower bounds for the feasible solutions in the MP. We empirically compare different BBC approaches based on combinations of feasibility and optimality cuts. In addition, we add valid inequalities to the MP, which helps to transfer information from the subproblems that is lost due to the decomposition. Construction and local search heuristics are also used to provide good initial solutions for the BBC.

The remainder of this paper is organized as follows. In Section 2, we describe the CSRP. Section 3 shows the related literature. In Section 4, we present the BBC algorithm. We discuss the computational results in Section 5. Finally, Section 6 presents final remarks and areas of future research.

2. Problem description

The CSRP is defined on an undirected and connected graph $G = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges (arcs). There are demand nodes ($\mathcal{V}^d \subset \mathcal{V}$) representing the affected cities and damaged nodes ($\mathcal{V}^r \subset \mathcal{V}$) representing the damaged points in the network. Demand nodes $i \in \mathcal{V}^d$ correspond to locations where some demand d_i for humanitarian assistance exists. Furthermore, there may be transshipment (intersection) nodes, which represent the intersection of two or more edges. Figure 1(a) shows an example of a network before being affected by extreme events (original network), while Figure 1(b) shows the corresponding network considering damaged nodes in the points where the edges (roads) were damaged. Notice that some edges can be damaged in more than one point. There is one depot (node 0) that is a supply node to be connected with the demand nodes and from which the repair crew initially departs to repair the damaged nodes. For each node $i \in \mathcal{V}$, there is a set $\mathcal{E}_i \subseteq \mathcal{E}$ representing the edges incident to node i. A damaged node $j \in \mathcal{V}^r$ has a repair time δ_j that represents the time the crew spends to repair the node j. A travel time τ_e and a length (distance) ℓ_e are defined for each edge $e \in \mathcal{E}$.



(a) Original network. (b) Damaged network.

Figure 1: Example of a graph representing the CSRP.

We consider a single crew available to perform the restoration activities. The problem consists of determining (i) the optimal crew scheduling to repair the damaged nodes, (ii) the paths that must be followed by the crew between two successive damaged nodes in the schedule, and (iii) the paths between the depot and the demand nodes. The damaged nodes must be repaired the first time they are visited by the crew, incurring in the repair time. In subsequent visits, the crew can use the already repaired damaged nodes without incurring in a repair time. Some damaged nodes cannot be repaired before the restoration of other damaged nodes. For instance, node 9 in Figure 1(b) cannot be repaired directly from the depot without the restoration of other damaged node (7, 8 or 10).

The objective of the CSRP consists of minimizing the time that the demand nodes remain inaccessible from the depot weighted by their corresponding demands. The accessibility of the demand nodes influences the delivery of commodities and the evacuation of affected people, and hence, it must be restored as soon as possible. A demand node $i \in \mathcal{V}^d$ is called accessible if there exists a path that connects this node to the depot using only undamaged and/or repaired nodes and that is not longer than a maximum distance l_i. The maximum distance l_i is based on pre-disaster conditions and has to be greater than or equal to the shortest distance between the depot and the demand node i. In Figure 1(a), for example, assuming a distance $\ell = 1$ in all the edges of the graph, the shortest distance from the depot to demand node 2 is 1. Then, $l_2 \geq 1$. If $l_2 = 1$, only path 0-2 can be used to connect the depot with demand node 2. On the other hand, if $l_2 = 3$, paths 0-2 and 0-3-1-2 can be used to connect the depot with demand node 2. Paths connecting the depot with the demand nodes can require the restoration of damaged nodes. In Figure 1(b), for example, if path 0-3-4 is defined for connecting node 4 with the depot, no damaged node must be repaired to make node 4 accessible. In this case, the time that demand node 4 remain inaccessible from the depot is equal to zero, as no damaged node is used in the path 0-3-4. On the other hand, if paths 0-2 and 0-5 are defined for connecting nodes 2 and 5 with the depot, respectively, then damaged node 6 must be repaired for demand node 2 to become accessible and damaged node 7 must be repaired for demand node 5 to become accessible. In this case, the time that the demand nodes 2 and 5 remain inaccessible from the depot is equal to the exact time at which nodes 6 and 7 are repaired, respectively. Furthermore, repairing damaged nodes 8, 9 and 10 is not necessary for connecting the depot with the demand nodes. However, these nodes also need to be repaired in the long term. To minimize the time that demand nodes remain inaccessible from the depot, it is expected that the optimal solution to the problem in Figure 1(b) considers first the restoration of damaged nodes 6 and 7 and then the restoration of nodes 8, 9 and 10.

3. Literature review

The CSRP has been tackled recently in the literature using exact methods and heuristics. Maya-Duque et al. (2016) developed a dynamic programming (DP) algorithm to optimally solve the CSRP. This approach is based on the gradual addition of damaged nodes to a schedule that starts in the depot, keeping a list of states with information about the elapsed time and current location of the crew, the unrepaired damaged nodes, and the inaccessible demand nodes.

However, the DP algorithm was able to solve to optimality only few (small) instances of the problem. An MIP formulation was also developed by the same authors, but they claimed that a direct implementation of the model in a commercial solver resulted in an intractable solution method even for small instances. Hence, they did not report computational results of using the model to solve the problem. Finally, because of the limitations regarding their exact approaches, the authors developed a metaheuristic based on GRASP to solve medium and large instances. Due to the heuristic nature of the method and the lack of lower bounds, the analysis of the quality of the solutions is compromised.

Variants of the CSRP that integrate scheduling and routing decisions have also been studied in the literature. Feng and Wang (2003) were the first authors to develop a mathematical model integrating scheduling and routing decisions towards road restoration, focusing on highway emergency rehabilitation after earthquakes. Different from the CSRP, they presented a multi-objective model to maximize both the total kilometers of roads repaired and the total number of saved lives, while minimizing the risk of the restoration operations. This model does not include the definition of paths between depots and the demand nodes. Furthermore, the dynamic changes in the accessibility of the nodes along the network are not taken in account. This means that the crew cannot visit some damaged nodes before the restoration of other damaged nodes. To incorporate the network dynamics, Yan and Shih (2007) devised a time-space network MIP model. This formulation considers copies i' of an original node i to represent the state of this node over the time horizon. The model minimizes the completion time of the restoration. They did not consider the design of paths to reach the demand nodes. To find feasible solutions for the problem, the authors proposed a heuristic algorithm that divides the originally damaged network into several smaller networks. Each subnetwork was then solved using a commercial solver. However, even the subproblems remain unsolvable in practical time. Therefore, in a subsequent study (Yan and Shih, 2012), the same authors implemented an ant colony system-based metaheuristic to solve practical instances of the problem.

Tang et al. (2009) used the same idea of time-space networks to model a stochastic version of the problem presented in Yan and Shih (2007). They incorporated both stochastic travel and repair times into the problem using a two-stage stochastic programming model. The first-stage refers to the scheduling and routing decisions, whereas the second-stage considers alternative routing decisions for each scenario. The model aims at minimizing the travel and repair times plus an expected penalty value for the modification of the routes. Small instances of the problem were solved by a commercial optimization solver.

Yan and Shih (2009) integrated crew scheduling and routing with relief distribution in a bi-objective model to minimize the completion time of the restoration, and the time due to the relief distribution to all demand nodes. The bi-objective model was reduced to a single objective via the evaluation of a weighted objective function, and thus was solved by a heuristic analogously to Yan and Shih (2007). Similarly, Yan et al. (2014) incorporated rescheduling repair decisions into the problem proposed by Yan and Shih (2007). Basically, they considered that backup repair crews can be dispatched to support the regular crews when subsequent events after the primary extreme event cause new damage nodes over the time horizon. An ant colony system-based metaheuristic was used to solve practical instances of the problem. Xu and Song (2015) also proposed optimizing crew scheduling and routing with relief distribution but focused on minimizing the time in which relief goods arrive at the demand nodes. The resulting problem was solved by an ant colony system-based metaheuristic.

Pramudita et al. (2012) and Pramudita and Taniguchi (2014) integrated location decisions with crew scheduling and routing decisions. They considered the problem as a variant of the undirected capacitated arc routing problem (CARP), in which there exists a set of blocked arcs that need to be unblocked. Additional constraints were added to the classical CARP to limit access to some section of the network as a result of debris-blocked arcs. Different from the CSRP, the objective is to minimize the cost of collecting the debris in all the damaged arcs. Furthermore, the definition of paths to reach the demand nodes was not considered by the authors. Pramudita and Taniguchi (2014) studied the same problem by transforming the CARP into the capacitated vehicle routing problem (CVRP). The transformation associates blocked arcs with two nodes that must be visited in sequence. Pramudita et al. (2012) and Pramudita and Taniguchi (2014) used a tabu search metaheuristic to solve practical instances of the problem.

Özdamar et al. (2014) proposed a multi-objective non-linear recursive model to minimize both the network inaccessibility and the completion time of the restoration operations. In their model, schedule decisions are generated for a fleet of dozers that perform the task of debris cleanup from blocked arcs. The authors developed heuristics based on priority selection rules to solve the problem. Akbari and Salman (2017b) introduced the multi-vehicle synchronized arc routing problem. The model optimally determines the set of debris-blocked roads that need to be repaired and the synchronized routes for the crews (vehicles) to clear these roads in the shortest completion time. They proposed an MIP formulation and a relaxation-based heuristic in which the routes of the crew might not be synchronized. Additionally, they developed a constructive heuristic to obtain a feasible solution from the unsynchronized solution and a neighborhood search algorithm to improve the feasible solutions. Finally, the same problem and its solution method were addressed in Akbari and Salman (2017a) with a different objective function consisting of maximizing the network components connected to the depot node. This problem differs from the CSRP in the sense that it considers different objective functions and multiple crews. Furthermore, it does not involve the design of paths to connect demand nodes with a central depot. Such paths are, in practice, used to perform the distribution of supplies or the evacuation of victims.

Notice that several studies devise heuristic/metaheuristic algorithms to address the CSRP or related problems. On the other hand, the literature on exact methods is still scarce. Despite the fact some authors rely on mathematical formulations and commercial solvers to solve small instances, practical instances have not been addressed. The dynamic programming algorithm proposed by Maya-Duque et al. (2016) is the only specialized exact method proposed for the CSRP, but it fails to solve even small instances of the problem. We are not aware of any decomposition-based exact algorithm proposed for the CSRP or its variants.

In this paper, we contribute to the literature on the CSRP by proposing a state-of-the-art exact approach based on the Benders decomposition and the branch-and-cut algorithm, referred to as the branch-and-Benders-cut (BBC) method. This is a challenging task, since the CSRP integrates two non-trivial combinatorial optimization problems. The components of the proposed method are specialized and sharpened to take advantage of the mathematical structure of the CSRP. For example, the method relies on combinatorial cuts and lower bound functions that are able to cut off and set lower bounds for multiple solutions simultaneously. We develop efficient specialized algorithms to solve the subproblems that emerge from the decomposition, instead of solving their corresponding classic MIP formulations via commercial solvers. We also propose valid inequalities that effectively accelerate the convergence of the BBC approach. Furthermore, construction and local search heuristics are also used to find good initial solutions for the method. As the proposed BBC approach is an exact algorithm, the solution quality can be assessed, which is relevant not only from the theoretical perspective, but also in practice, as it can help decision-makers to rely on solutions that are known to be optimal or near-optimal. We could verify experimentally that our algorithm is efficient to solve many practical-size instances. Moreover, we show that reasonable solutions are obtained even for very large-scale instances that have never been tackled before by exact methods.

4. Solution approach

In this section, we present a mathematical formulation and propose the BBC algorithm. Basically, this algorithm has three main components: an MIP master problem defined in Subsection 4.2, optimality and feasibility cuts defined in Subsection 4.3 and separation routines defined in Subsection 4.4. The MP considers only scheduling decisions for the crew, while subproblems determine the paths between pairs of damaged nodes and between the depot and the demand nodes. The solutions of a MP are used to generate feasibility and optimality cuts that cut off solutions corresponding to infeasible schedules. A flowchart showing the interaction between the main components of the proposed BBC algorithm is presented in Subsection 4.5. Additionally, in Subsection 4.6, we derive valid inequalities to have stronger LP relaxations, and in Subsection 4.8, we develop construction and local search heuristics to find good feasible solutions.

4.1. Mathematical modeling

To formulate the CSRP, we closely follow the mathematical model mentioned in Maya-Duque et al. (2016). The notation used to describe the model is as follows.

<u>Sets</u>

$\mathcal{V}$	Set of nodes.
$\mathcal{V}^d \subset \mathcal{V}$	Set of demand nodes.
$\mathcal{V}^r \subset \mathcal{V}$	Set of damaged nodes.
$\mathcal{E}$	Set of arcs.
$\mathcal{E}_i \subseteq \mathcal{E}$	Set of arcs incident to node $i \in \mathcal{V}$.

Parameters

d_i	Demand of node $i \in \mathcal{V}^d$.
δ_i	Repair time of node $i \in \mathcal{V}^r$.
τ_e	Travel time on arc $e \in \mathcal{E}$.
ℓ_e	Length (distance) of arc $e \in \mathcal{E}$.
l_i	Maximum distance allowed between the depot and the demand node $i \in \mathcal{V}^d$.
M	A sufficiently large number.

Decision variables

$$X_{ij} = \begin{cases} 1, \text{ if node } j \in \mathcal{V}^r \cup \{0\} \text{ is repaired immediately after node } i \in \mathcal{V}^r \cup \{0\}. \\ 0, \text{ otherwise.} \end{cases}$$

$$P_{eij} = \begin{cases} 1, \text{ if arc } e \in \mathcal{E} \text{ is used on the path from node } i \in \mathcal{V}^r \cup \{0\} \text{ to node } j \in \mathcal{V}^r \cup \{0\}. \\ 0, \text{ otherwise.} \end{cases}$$

$$N_{kij} = \begin{cases} 1, \text{ if node } k \in \mathcal{V} \text{ is used on the path from node } i \in \mathcal{V}^r \cup \{0\} \text{ to node } j \in \mathcal{V}^r \cup \{0\}. \\ 0, \text{ otherwise.} \end{cases}$$

$$Y_{ej} = \begin{cases} 1, \text{ if arc } e \in \mathcal{E} \text{ is used on the path from supply node 0 to node } j \in \mathcal{V}^d. \\ 0, \text{ otherwise.} \end{cases}$$

$$V_{kj} = \begin{cases} 1, \text{ if node } k \in \mathcal{V} \text{ is used on the path from supply node 0 to node } j \in \mathcal{V}^d. \\ 0, \text{ otherwise.} \end{cases}$$

Z_i^r	Exact time at which the damaged node $i \in \mathcal{V}^r$ is repaired.
Z_i^d	Exact time at which the demand node $i \in \mathcal{V}^d$ becomes accessible. If node i is accessible at time zero, this variable takes value zero.

Note that the variables X_{ij} define the schedule of the crew, i.e., the sequence of damaged nodes to be repaired. They do not provide the route of the crew, as they are defined for damaged nodes only. The full route is obtained from variables P_{eij} and N_{kij}, which determine the arcs and nodes, respectively, to be visited in a path between each two consecutive damaged nodes $i - j$ in the schedule of the crew. On the other hand, variables Y_{ej} and V_{kj} define the arcs and nodes, respectively, to be visited in the paths between the depot and each demand node j. These two types of variables are not related to the crew.

The model is formulated as follows:

$$\min \sum_{i \in \mathcal{V}^d} d_i \cdot Z_i^d. \tag{1}$$

$$\text{s.t.} \sum_{j \in \mathcal{V}^r \cup \{0\}} X_{ij} = 1, \ \forall\, i \in \mathcal{V}^r \cup \{0\}, \tag{2}$$

$$\sum_{i \in \mathcal{V}^r \cup \{0\}} X_{ij} = 1, \ \forall\, j \in \mathcal{V}^r \cup \{0\}, \tag{3}$$

$$\sum_{e \in \mathcal{E}_i} P_{eij} = X_{ij}, \ \forall\, i \in \mathcal{V}^r \cup \{0\}, \ j \in \mathcal{V}^r, \tag{4}$$

$$\sum_{e \in \mathcal{E}_j} P_{eij} = X_{ij}, \ \forall\, i \in \mathcal{V}^r \cup \{0\}, \ j \in \mathcal{V}^r, \tag{5}$$

$$\sum_{e \in \mathcal{E}_k} P_{eij} = 2N_{kij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r,\ k \in \mathcal{V} \setminus \{i,\ j\}, \tag{6}$$

$$\sum_{e \in \mathcal{E}_0} Y_{ej} = 1, \forall\, j \in \mathcal{V}^d, \tag{7}$$

$$\sum_{e \in \mathcal{E}_j} Y_{ej} = 1, \forall\, j \in \mathcal{V}^d, \tag{8}$$

$$\sum_{e \in \mathcal{E}_k} Y_{ej} = 2V_{kj}, \forall\, j \in \mathcal{V}^d,\ k \in \mathcal{V} \setminus \{0,\ j\}, \tag{9}$$

$$\sum_{e \in \mathcal{E}} Y_{ej} \cdot \ell_e \leq l_j, \forall\, j \in \mathcal{V}^d, \tag{10}$$

$$Z_j^r \geq Z_i^r + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + \delta_j - (1 - X_{ij}) \cdot M, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{11}$$

$$Z_j^r \geq Z_k^r + (N_{kij} - 1) \cdot M, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r,\ k \in \mathcal{V}^r, \tag{12}$$

$$Z_i^d \geq Z_j^r + (V_{ji} - 1) \cdot M, \forall\, i \in \mathcal{V}^d,\ j \in \mathcal{V}^r, \tag{13}$$

$$X_{ij} \in \{0,\, 1\}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r \cup \{0\}, \tag{14}$$

$$P_{eij}, N_{kij} \in \{0,\, 1\}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r,\ k \in \mathcal{V},\ e \in \mathcal{E}, \tag{15}$$

$$Y_{ei}, V_{ki} \in \{0,\, 1\}, \forall\, i \in \mathcal{V}^d,\ k \in \mathcal{V},\ e \in \mathcal{E}, \tag{16}$$

$$Z_i^r \geq 0, \forall\, i \in \mathcal{V}^r \cup \{0\}, \tag{17}$$

$$Z_i^d \geq 0, \forall\, i \in \mathcal{V}^d. \tag{18}$$

The objective function (1) consists of minimizing the time that the demand nodes remain inaccessible from the depot, weighted by their corresponding demands. A demand node $j \in \mathcal{V}^d$ is called accessible if there exists a path that connects this node to the depot using only undamaged and/or repaired nodes and that is not longer than a maximum distance l_j – see constraints (10). Thus, the accessibility time of a demand node depends on the damaged nodes in its path from the depot and is computed in constraints (13). In Figure 1(b), for example, paths 0-6-2 and 0-7-5 can be defined for connecting nodes 2 and 5 with the depot, respectively. Thus, the times that the demand nodes 2 and 5 remain inaccessible from the depot are equal to the exact times at which nodes 6 and 7 are repaired, respectively. Constraints (2) and (3) specify that each damaged node must be visited once during the schedule of the crew. Constraints (4), (5) and (6) ensure the flow conservation in the path of the crew between damaged nodes i and j. If there is a path between damaged nodes i and j ($X_{ij} = 1$), constraints (4) force the use of an arc incident to node i in the path, while constraints (5) force the use of an arc incident to node j in the path. Furthermore, for each node k in the path from i to j ($N_{kij} = 1$), there is one arc leaving and one arc arriving at node k considered in the path, as imposed by constraints (6). Similarly, constraints (7), (8) and (9) ensure the flow conservation in the paths from the depot to the demand nodes. Constraints (10) prohibit the use of paths with a distance greater than the maximum distance allowed between the depot and the demand nodes. Notice that l_j considers the distances only, not travel or repair times. Constraints (11) define the exact time at which the damaged nodes are repaired. For a given node j, this is the result of adding the time at which the predecessor node i is repaired plus the travel time of the path from node i to node j plus the time it takes to repair node j. These constraints also act as subtour elimination constraints and are based on the Miller-Tucker-Zemlin (MTZ) formulation of the traveling salesman problem (TSP) (Miller et al., 1960), which has a number of constraints that depends polynomially on the number of nodes. They are different from the subtour elimination constraints originally used in the model cited by Maya-Duque et al. (2016), which are based on the Dantzig-Fulkerson-Johnson (DFJ) formulation of the TSP (Dantzig et al., 1954) and lead to a number of constraints that is exponential in terms of the number of nodes. To keep the model polynomial-sized, we decided to use the MTZ-based constraints. Constraints (12) ensure that a node k in the path from node i to node j must be repaired before node j; i.e., damaged unrepaired nodes cannot be used in a path from node i to node j. Constraints (13) define the exact time at which each demand node i become accessible, which is based on the time when damaged nodes in the path connecting i to the depot are repaired. Finally, constraints (14)-(18) impose the domain of the decision variables. It is worth mentioning that variables P_{eij} and Y_{ej} do not need to be defined as binary variables in the computational implementation because they naturally assume binary values if variables N_{kij} and V_{kj} are defined as binaries, respectively.

4.2. Benders decomposition

Benders decomposition is a variable partitioning technique whose goal is to tackle problems with complicating variables (Benders, 1962; Costa, 2005; de Sá et al., 2013). Usually, a master problem considering only the complicating variables is solved, then the complicating variables are temporarily fixed, and one or more subproblems are solved. For the CSRP, we identified as complicating variables the X_{ij} variables, which define the schedule of the crew. When the scheduling decisions (X_{ij}) are fixed, the remaining problem becomes a set of shortest-path problems, which can be efficiently solved by using specialized algorithms based on the well-known Dijkstra's shortest-path algorithm (Dijkstra, 1959). The master problem is defined as follows:

$$(MP) \qquad \min \quad \Theta, \tag{19}$$

$$\text{s.t.} \quad \text{Constraints } (2), (3), (14), \tag{20}$$

$$R_j \geq R_i + 1 - |\mathcal{V}^r \cup \{0\}| \cdot (1 - X_{ij}),\ \forall\ i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{21}$$

$$\Theta \geq \sum_{i \in \mathcal{V}^d} d_i \cdot \theta_i, \tag{22}$$

$$\Theta, \theta_i, R_j\ \geq 0,\ \forall\ i \in \mathcal{V}^d,\ j \in \mathcal{V}^r \cup \{0\}. \tag{23}$$

Model (19)-(23) still lacks the feasibility and optimality cuts to be defined in Subsection 4.3. Notice that constraints (11), which act also as subtour elimination constraints in model (1)-(18), do not remain in the MP (they go to the subproblems because of variables Z_j^r and P_{eij}). Thus, we add the new subtour elimination constraints (21) to the MP, together with the auxiliary variables R_j. Variable θ_i computes the exact time at which the demand node $i \in \mathcal{V}^d$ becomes accessible, and Θ computes the value of the objective function. Initially, the lower bound for the Θ and θ_i variables is zero. When a solution is found for the MP, feasibility or optimality cuts are added, and they are likely to increase the lower bound of the Θ and/or θ_i variables. We can set a lower bound for variable Θ directly or by using the θ_i variables. Constraint (22) guarantees that the addition of optimality cuts setting a lower bound for the variables θ_i also sets a lower bound for the variable Θ.

The MP determines a schedule for the crew. The feasibility of this schedule for the original model (1)-(18) is verified in subproblem SP1, which obtains a set of shortest paths between consecutive nodes in the schedule of the crew:

$$
\begin{aligned}
(SP1)\ \min \quad & \sum_{i \in V^r} Z_i^r, && (24)\\
\text{s.t.} \quad & \text{Constraints } (6), (12), (15), (17), && (25)\\
& \sum_{e \in \mathcal{E}_i} P_{eij} = \widehat{X}_{ij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, && (26)\\
& \sum_{e \in \mathcal{E}_j} P_{eij} = \widehat{X}_{ij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, && (27)\\
& Z_j^r \geq Z_i^r + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + (\widehat{X}_{ij} - 1) \cdot M + \delta_j, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, && (28)
\end{aligned}
$$

in which $\widehat{X}_{ij}$ is a solution for the MP. For each pair of consecutive nodes $i - j$ with $\widehat{X}_{ij} = 1$ in the schedule defined by the MP, SP1 determines the shortest path with arcs and nodes defined by variables P_{eij} and N_{kij}, respectively. Indeed, for this pair $i - j$, constraints (28) become

$$Z_j^r \geq Z_i^r + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + \delta_j$$

and hence, the objective function becomes a summation of the repair times and travel times on the traversed arcs.

SP1 may be infeasible if there is no path between two nodes $i - j$ that uses only undamaged and/or repaired nodes. In such a case, the schedule $\widehat{X}_{ij}$ provided by the MP is infeasible in the original problem (1)-(18), and feasibility cuts must be added to the MP (see Subsection 4.3). Otherwise, the values of the variables Z_i^r are used to calculate the total cost of the schedule in subproblem SP2, which determines the shortest paths between the depot and the demand nodes. It can be defined as follows:

$$
\begin{aligned}
(SP2) \qquad \min \quad & \sum_{i \in \mathcal{V}^d} d_i \cdot Z_i^d, && (29)\\
\text{s.t.} \quad & \text{Constraints } (7), (8), (9), (10), (16), (18), && (30)\\
& Z_i^d \geq \widehat{Z}_k^r + (V_{ki} - 1) \cdot M, \forall\, i \in \mathcal{V}^d,\ k \in \mathcal{V}^r, && (31)
\end{aligned}
$$

where parameter $\widehat{Z}_k^r$ is obtained from a solution of subproblem SP1. Subproblem SP2 determines the shortest paths from the depot to each demand node $i \in \mathcal{V}^d$ with a distance length less than or equal to the maximum distance l_i. Each path is composed of arcs and nodes defined by variables Y_{ej} and V_{kj}, respectively. The exact time at which the demand node $i \in \mathcal{V}^d$ becomes accessible is used to generate optimality cuts for the MP, as defined in the next section. From subproblem SP2, we derive only optimality cuts. If subproblem SP2 is infeasible, then the original problem (1)-(18) is also infeasible because there is no path between the depot and some demand node

i with a distance length less than or equal to the maximum distance l_i (it considers only the distance of each arc, not the travel times or the repair times).

Therefore, we have decomposed the decisions of the CSRP into three parts: the MP, which determines the crew schedule; SP1, which checks whether this schedule is feasible and, if it is, obtains crew paths between each pair of damaged nodes; and SP2, which determines the paths between the depot and each demand node and the corresponding objective costs. Note that we could have defined a single subproblem by gathering subproblems SP1 and SP2 and hence evaluated both the feasibility and cost of the MP solutions simultaneously. However, having separate subproblems allows us to design efficient specialized algorithms, as presented in Subsection 4.4.

4.3. Combinatorial Benders cuts and lower-bounding functions

Every time an integer solution is found by the BBC algorithm, separation procedures based on specialized solution methods for subproblems SP1 and SP2 seek violated feasibility or optimality cuts, and the corresponding combinatorial Benders cuts (feasibility cuts) or lower-bounding functions (optimality cuts) are added to the MP. We rely on feasibility and optimality cuts based on particular characteristics of the problem and on inequalities proposed for related problems in the literature (Hjorring and Holt, 1999; Laporte et al., 2014). Proposition 1 states feasibility cuts for the MP.

Proposition 1. *Let $K = (v_0, v_1, ..., v_{(h-1)}, v_h, ..., v_p, ..., v_{|\mathcal{V}^r|})$ be an infeasible schedule for the crew, where v_i is the ith damaged node to be repaired and $v_0 = 0$. Assume that K is obtained by solving the MP and corresponds to the solution $\widehat{X}_{v_{(i-1)}v_i} = 1$, $\forall i = 1, ..., |\mathcal{V}^r|$. For a given index $h > 0$, let $S_h = \{v_0, v_1, ..., v_{(h-1)}, v_h\}$, and assume that K is infeasible because there exists no path from node $v_{(h-1)}$ to node v_h without using at least one damaged node not yet repaired v_p, with $p > h$. Hence, the following feasibility cuts are violated and can be added to the MP:*

$$\sum_{i \in S_h \setminus \{v_h\}} \sum_{\substack{j \in S_h \setminus \{v_0\}: \\ i \neq j}} X_{ij} \leq |S_h| - 2, \tag{32}$$

$$\sum_{i \in S_h} \sum_{\substack{j \in S_h: \\ \widehat{X}_{ij}=1}} X_{ij} \leq |S_h| - 2. \tag{33}$$

Proof. Assume that there is no feasible path from node $v_{(h-1)}$ to node v_h. Hence, there is at least one damaged node v_p, with $p > h$, that must be repaired before node v_h (otherwise, v_h cannot be reached). Let $\bar{S}_h$ be any permutation of elements of set $S_h \setminus \{v_0, v_h\}$. Every schedule containing any partial sequence $\bar{K} = (v_0, \bar{S}_h, v_h)$ is infeasible because the node v_p is not repaired before node v_h. Then, all the schedules that contain any partial sequence $\bar{K}$ must be avoided. Every partial sequence $\bar{K}$ can be represented in the MP by binary variables in the left-hand side of (32), where $|S_h| - 1$ of them takes a value of 1. Therefore, to avoid any sequence $\bar{K}$, it is necessary to restrict the left-hand side of (32) to be strictly smaller than $|S_h| - 1$. The cut defined in (33) is a particular case of cut (32) to avoid any schedule with the sequence $\bar{K} = S_h$.□

To illustrate Proposition 1, consider the schedule $K = \{v_0, v_1, v_2, v_3, v_4, v_5\} = \{0, 3, 1, 2, 4, 5\}$ that is assumed to be infeasible because there is no path from node 1 to node 2 without using node 5. Then, $v_h = v_3 = 2$ and $S_h = S_3 = \{0, 3, 1, 2\}$. The possible permutations of set $S_h \setminus \{v_0, v_h\}$ are $\bar{S}_h^1 = \{3, 1\}$ and $\bar{S}_h^2 = \{1, 3\}$. Thus, all the schedules that contain the partial sequences $\bar{K}^1 = \{0, 3, 1, 2\}$ and $\bar{K}^2 = \{0, 1, 3, 2\}$ must be avoided. The feasibility cut (32) is $X_{03} + X_{01} + X_{02} + X_{13} + X_{12} + X_{23} + X_{21} + X_{31} + X_{32} \leq 2$, where sequence $\bar{K}^1$ is represented by variables X_{03}, X_{31}, and X_{12} and sequence $\bar{K}^2$ is represented by variables X_{01}, X_{13}, and X_{12}. Each sequence is represented by three binary variables taking a value of 1, so to avoid the schedules with the infeasible sequences $\bar{K}^1$ and $\bar{K}^2$, we need to force these variables to sum to less than 3. The feasibility cut (33) considering only the sequence S_h is $X_{03} + X_{31} + X_{12} \leq 2$.

Note that the cut defined in (32) avoids all schedules with a partial sequence starting at node v_0, ending at node v_h, and containing nodes from set $S_h \setminus \{v_0, v_h\}$ (in any order). Thus, it cuts off every schedule with any partial sequence $\bar{K} = (v_0, \bar{S}_h, v_h)$. Equation (33) is a cut to avoid every schedule with a partial sequence starting at node v_0, ending at node v_h, and containing nodes from set S_h (in the original order), cutting off every schedule with a partial sequence $\bar{K} = S_h$. Only one of them, (32) or (33), is necessary to cut off the solution corresponding to K. However, the number of solutions cut off by (32) is greater than or equal to the number of solutions cut off by (33).

When a solution of the MP is feasible for the original model (1)-(18), optimality cuts must be added to properly set the corresponding cost. Proposition 2 defines optimality cuts for the variable Θ of the MP.

Proposition 2. *Let $L = (v_0, v_1, ...v_{(h-1)}, v_h)$ be a feasible partial sequence of damaged nodes repaired by the crew corresponding to the MP solution $\widehat{X}_{v_{(i-1)}v_i} = 1$, $\forall i = 1, ..., h$, where $v_0 = 0$ and v_h is the last node to be repaired to make all the demand nodes in the set $\mathcal{V}^d$ accessible. An optimality cut to be added to the MP is:*

$$\Theta \geq \widehat{\Theta} \cdot (\sum_{i=1}^{h} X_{v_{(i-1)}v_i} - (h-1)), \tag{34}$$

where $\widehat{\Theta}$ is the total cost computed in subproblem SP2.

Proof. All the demand nodes become accessible when node v_h in the partial sequence L is repaired, with a corresponding total cost $\widehat{\Theta}$. Hence, every schedule containing the sequence L must have a cost $\widehat{\Theta}$. The sequence L is represented by binary variables in the right-hand side of (34) when those h binary variables take value 1. Then, if the partial sequence L is considered in the schedule, the summation is equal to h, and we have the lower bound $\widehat{\Theta}$ for variable Θ activated in the MP, as $\Theta \geq \widehat{\Theta} \cdot (h - (h-1))$. Otherwise, if the partial sequence L is not considered in the schedule, there are $p < h$ variables taking a value of 1 in the right-hand side of (34), and the lower bound $\widehat{\Theta}$ cannot be activated in the MP, as we have $\Theta \geq \widehat{\Theta}(p - (h-1))$ with $p < h$. $\square$

Cut (34) sets a lower bound for variable Θ only. Proposition 3 defines optimality cuts based on variables θ_i, $\forall i \in \mathcal{V}^d$.

Proposition 3. *Let $L^k = (v_0^k, v_1^k, ..., v_{(h-1)}^k, v_h^k)$ be a feasible partial sequence of damaged nodes repaired by the crew corresponding to the MP solution $\widehat{X}_{v_{(i-1)}^k v_i^k} = 1$, $\forall i = 1, \ldots, h$, where $v_0^k =$*

0 and v_h^k is the last node repaired to make the demand node $k \in \mathcal{V}^d$ accessible. Let $P_h^k = \{v_1^k, ..., v_{(h-1)}^k\}$. Let $\bar{P}_h^k$ be any permutation of elements of set P_h^k and $\bar{L}^k = (v_0^k, \bar{P}_h^k, v_h^k)$. Then, the following optimality multi-cuts can be added to the MP:

$$\theta_k \geq \widetilde{\theta}_k \cdot \left(\sum_{i \in P_h^k} (X_{0i} + X_{i(v_h^k)}) + \sum_{i \in P_h^k} \sum_{\substack{j \in P_h^k: \\ i \neq j}} X_{ij} - |P_h^k| \right), \forall\ k \in \mathcal{V}^d, \tag{35}$$

$$\theta_k \geq \widehat{\theta}_k \cdot \left(X_{0v_1} + X_{(v_{h-1}^k)(v_h^k)} + \sum_{i \in P_h^k} \sum_{\substack{j \in P_h^k: \\ \widehat{X}_{ij}=1}} X_{ij} - |P_h^k| \right), \forall\ k \in \mathcal{V}^d, \tag{36}$$

where $\widehat{\theta}_k = \widehat{Z}_k^d$, $\forall\ k \in \mathcal{V}^d$, is computed in subproblem SP2 and $\widetilde{\theta}_k$ is a lower bound for variable θ_k when any partial sequence $\bar{L}^k$ is considered in the schedule. $\widetilde{\theta}_k$ can be computed as:

$$\widetilde{\theta}_k = \sum_{j \in P_h^k \cup \{v_h^k\}} \delta_j + \sum_{j \in P_h^k \cup \{v_h^k\}} t_j^*, \tag{37}$$

$$t_j^* = \min_{\substack{i \in P_h^k \cup \{v_0\}: \\ i \neq j}} \{t_{ij}\},\ \forall\ j \in P_h^k \cup \{v_h^k\}, \tag{38}$$

where δ_j is the repair time of node j and t_{ij} is the cost of the shortest path from node i to node j considering that all nodes are repaired.

Proof. Cut (36) is a particular case of (35) to set the cost $\widehat{\theta}_k$ for variables θ_k corresponding to the original schedule L^k. For every partial sequence $\bar{L}^k = (v_0^k, \bar{P}_h^k, v_h^k)$, the demand node k becomes accessible when node v_h^k is repaired. For a given $\bar{L}^k$, we do not have the actual cost for variables θ_k. Instead, we have a valid lower bound $\widetilde{\theta}_k$. In the calculation of $\widetilde{\theta}_k$, we consider that in any sequence $\bar{L}^k$, all the nodes of set P_h^k must be repaired, and then the total repair time $(\sum_{j \in P_h^k \cup \{v_h^k\}} \delta_j)$ must be computed. Additionally, the crew must arrive at all the damaged nodes in the sequence $\bar{L}^k$, and then we compute a lower bound using the minimum travel time to arrive at each node $j \in \bar{L}^k$ from any other node $i \in \bar{L}^k$ $(\sum_{j \in P_h^k \cup \{v_h^k\}} t_j^*)$. As a result, $\widetilde{\theta}_k$ must be less than or equal to the actual accessibility time $\widehat{\theta}_k$. □

The optimality cut (34) sets a lower bound (actual cost) for the total cost Θ for any schedule with the partial sequence L. Cut (36) is similar to (34) but sets a lower bound (actual cost) for variables θ_k, $\forall k \in \mathcal{V}^d$, which in turn sets a lower bound for the total cost Θ. Only one of them, either (34) or (36), is necessary to set the actual total cost for every schedule with partial sequence L. Cut (35) sets a valid (underestimated) lower bound for any schedule with any partial sequences $\bar{L}^k$, $\forall k \in \mathcal{V}^d$, so it sets a lower bound for more solutions in the MP than cuts (34) and (36).

4.4. Separation procedures

Both subproblems SP1 and SP2 can be efficiently solved via specialized methods based on Dijkstra's shortest-path algorithm (Dijkstra, 1959) instead of using the models (24)-(28) and (29)-(31), respectively. A pseudo-code of the method proposed to solve SP1 is outlined in Algorithm 1. The graph $G = (\mathcal{V}, \mathcal{E})$, a schedule $K = (v_0, v_1, ..., v_i, ..., v_{|\mathcal{V}^r|})$, the repair times δ_j, $\forall j \in \mathcal{V}^r$, and the travel times τ_e, $\forall e \in \mathcal{E}$ are used as the input of the algorithm. If SP1 is feasible for the schedule K, then the output of the algorithm is given by the optimal values for variables Z_i^r, $\forall i \in \mathcal{V}^r$. Otherwise, the algorithm indicates that the subproblem is infeasible.

Algorithm 1 Algorithm for solving SP1.

```
Input:
Graph $G = (\mathcal{V}, \mathcal{E})$;
Scheduling solution $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$;
Parameters $\delta_j$, $\forall j \in \mathcal{V}^r$, and $\tau_e$, $\forall e \in \mathcal{E}$;
Output:
If SP1 is feasible, return "Feasible SP1" and save optimal values $\widehat{Z}_j^r$, $\forall j \in \mathcal{V}^r$;
If SP1 is infeasible, return "Infeasible SP1";
1:  $C_e := \tau_e$, $\forall e \in \mathcal{E}$;
2:  $C_e := \infty$, $\forall e \in \mathcal{E}_j, j \in \mathcal{V}^r$;
3:  $\widehat{Z}_j^r := 0$, $\forall j \in \mathcal{V}^r$;
4:  for $j = 1$ to $|\mathcal{V}^r|$ do
5:      $C_e := \tau_e + \delta_{v_j}$, $\forall e \in \mathcal{E}_{v_j}$;
6:      Find the cost $\mathcal{C}$ of the shortest path from node $v_{j-1}$ to $v_j$;
7:      if $\mathcal{C} < \infty$ then
8:          $\widehat{Z}_{v_j}^r := \widehat{Z}_{v_{j-1}}^r + \mathcal{C}$;
9:          $C_e := \tau_e$, $\forall e \in \mathcal{E}_{v_j} : e \notin \bigcup_{i=j+1}^{|\mathcal{V}^r|} \mathcal{E}_{v_i}$ ;
10:         $C_e := \infty$, $\forall e \in \mathcal{E}_{v_j} : e \in \bigcup_{i=j+1}^{|\mathcal{V}^r|} \mathcal{E}_{v_i}$ ;
11:     else
12:         return "Infeasible SP1";
13:     end if
14: end for
15: return "Feasible SP1";
```

Algorithm 1 starts by setting the cost C_e of each arc in the network as ∞ if the arc e is incident to a damaged node; otherwise, this cost is set as τ_e (lines 1 and 2). Then, iteratively and for each damaged node $v_j \in K \setminus \{v_0\}$, the cost C_e of each arc $e \in \mathcal{E}_{v_j}$ (i.e., incident to v_j) is reset as $\tau_e + \delta_{v_j}$ (line 5), and Dijkstra's algorithm is used to find the shortest path between nodes v_{j-1} and v_j (line 6). If a path between nodes v_{j-1} and v_j exists without using a damaged node (that was not repaired yet), the cost $\mathcal{C}$ of the path must be less than ∞, and the value of variable $Z_{v_j}^r$ is updated (line 8). The cost C_e of each arc incident to the damaged node v_j is also updated, as this node has been repaired (line 9).

It is important to emphasize that the arcs incident to damaged nodes not yet repaired have cost ∞. Thus, if an arc incident to node v_j is also incident to another damaged node not yet repaired, then that arc must continue with cost ∞ (line 10). If there is no path between nodes v_{j-1} and v_j without using a not yet repaired damaged node (i.e. $\mathcal{C} = \infty$), then the algorithm terminates and returns that SP1 is infeasible (line 12).

Figure 2 shows an example of the variation in the cost C_e in Algorithm 1 for a network with two damaged nodes, crew schedule $K = (v_0, v_1, v_2)$, $\tau_e = 1$ and $\delta_j = 2$, for all $e \in \mathcal{E}$ and $i \in \mathcal{V}^r$.

Initially, the cost C_e of each arc incident to damaged nodes v_1 and v_2 is equal to ∞. In the first iteration, the cost of each arc incident to node v_1 is then reset to $C_e = \tau_e + \delta_{v_1} = 3$ and the Dijkstra's algorithm finds the shortest path between nodes v_0 and v_1. Once node v_1 is repaired, the cost of each arc incident to node v_1 is updated to $C_e = \tau_e = 1$. In the second iteration, the cost of each arc incident to node v_2 becomes $C_e = \tau_e + \delta_{v_j} = 3$ and Dijkstra's algorithm is now used to find the shortest path between nodes v_1 and v_2. Finally, after all nodes have been repaired, the cost of each arc becomes $C_e = \tau_e = 1$.

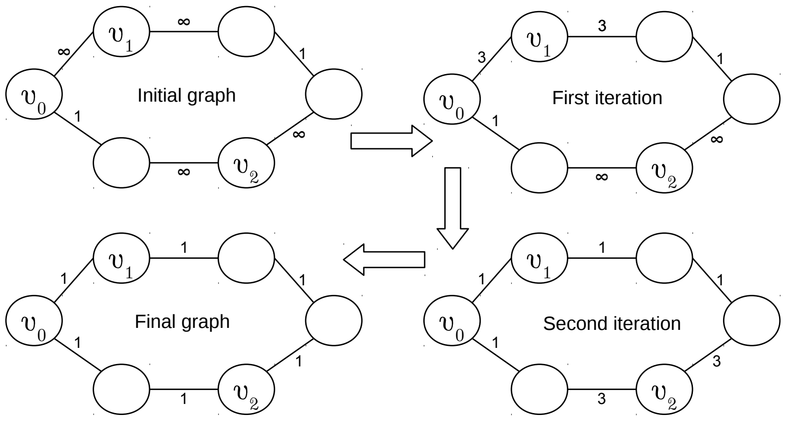


Figure 2: Example of the variation in arc costs in Algorithm 1.

An iterative solution approach based on solving a sequence of shortest-path problems can be used for solving subproblem SP2 as well. Recall that the goal of this subproblem is to determine the time at which affected areas become accessible. A node is accessible if there is a path from the depot to this node using only undamaged and/or repaired nodes and if the length of this path is no longer than a maximum distance l_i. If it is possible to access a demand node $i \in \mathcal{V}^d$ without using only undamaged nodes, then it becomes accessible at time $\widehat{Z}^d_i = 0$. Otherwise, let $j \in \mathcal{V}^r$ be the last damaged node that was repaired before i becomes accessible at exact time $\widehat{Z}^r_j$. Then, $\widehat{Z}^d_i = \widehat{Z}^r_j$. Notice that in subproblem SP2, given any two demand nodes i_1, i_2, the shortest path determined from the depot to i_1 is independent of the path determined from the depot to i_2. Hence, SP2 can be decomposed into $|\mathcal{V}^d|$ independent subproblems.

Algorithm 2 presents the pseudo-code of the proposed approach. The graph $G = (\mathcal{V}, \mathcal{E})$, a schedule $K = (v_0, v_1, ..., v_i, ..., v_{|\mathcal{V}^r|})$, the corresponding values of variables Z^r_i provided by the SP1, and the parameters ℓ_e, $\forall e \in \mathcal{E}$; l_i, $\forall i \in \mathcal{V}^d$; and d_i, $\forall i \in \mathcal{V}^d$ are considered the input of the algorithm. Initially, the cost C_e of each arc in the network is set as ℓ_e (line 1), the actual length (distance) of the arc. Iteratively, for each damaged node $v_j \in K \setminus \{v_0\}$, the algorithm sets the cost of each arc incident to v_j as ∞ (line 4) starting with the last damaged node ($v_{|\mathcal{V}^r|}$) in the schedule K. Then, for each demand node $i \in \mathcal{V}^d$ (line 5), Dijkstra's algorithm is used to find the shortest path from the depot to this node (line 6). If the cost $\mathcal{C}_i$ to reach this demand node is larger than the maximum allowed distance l_i (line 7), then the node v_j is necessary to find a path with cost smaller than the maximum distance l_i, and hence, the time instant $\widehat{Z}^d_i$ in which the demand node i becomes accessible is set as $\widehat{Z}^r_{v_j}$ (line 8). Note that we update $\widehat{Z}^d_i$ only if it was not updated in previous iterations; thus, $\widehat{Z}^d_i$ is equal to the largest repair time of the damaged nodes visited in the path from the depot to node i. Finally, the total cost $\widehat{\Theta}$ is computed (line

12). Recall that subproblem SP2 is always feasible if the original problem (1)-(18) is feasible.

Algorithm 2 Algorithm for solving the SP2.

```
Input:
Graph $G = (\mathcal{V}, \mathcal{E})$;
Scheduling solution $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$;
Time $\widehat{Z}_i^r$ at which damaged node $i \in \mathcal{V}^r$ is repaired;
Parameters $\ell_e$, $\forall e \in \mathcal{E}$, $l_i$, $\forall i \in \mathcal{V}^d$ and $d_i, \forall i \in \mathcal{V}^d$;
Output:
Time $\widehat{Z}_i^d$ at which the demand node $i \in \mathcal{V}^d$ becomes accessible;
Total cost $\widehat{\Theta}$;
 1: $C_e := \ell_e$, $\forall e \in \mathcal{E}$;
 2: $\widehat{Z}_i^d := 0$, $\forall i \in \mathcal{V}^d$;
 3: for $j = |\mathcal{V}^r|$ to 1 do
 4:     $C_e := \infty$, $\forall e \in \mathcal{E}_{v_j}$;
 5:     for $i = 1$ to $|\mathcal{V}^d|$ do
 6:         Find the cost $\mathcal{C}_i$ of the shortest path from the depot to the demand node $i$;
 7:         if $\mathcal{C}_i > l_i$ and $\widehat{Z}_i^d = 0$ then
 8:             $\widehat{Z}_i^d := \widehat{Z}_{v_j}^r$;
 9:         end if
10:     end for
11: end for
12: Compute total cost $\widehat{\Theta} := \sum_{i \in \mathcal{V}^d} d_i \cdot \widehat{Z}_i^d$;
```

Figure 3 shows an example of the variation in cost C_e in Algorithm 2 for a network with two damaged nodes, $K = (v_0, v_1, v_2)$ and $\ell_e = 1$, $\forall e \in \mathcal{E}$. In the first iteration, the cost of each arc incident to the last damaged node v_2 in the schedule is set to $C_e = \infty$, and Dijkstra's algorithm finds the shortest paths between node v_0 and each demand node $i \in \mathcal{V}^d$. If the cost $\mathcal{C}_i$ of the path between v_0 and the demand node i is larger than l_i, then the node v_2 is necessary to find a path with length smaller than l_i, and $\widehat{Z}_i^d = \widehat{Z}_{v_2}^r$. In the second iteration, the cost of each arc incident to the damaged node v_1 is updated with $C_e = \infty$, and the shortest paths between node v_0 and demand nodes $i \in \mathcal{V}^d$ are found again. In this case, if it is not possible to find a path for a demand node i with a cost $\mathcal{C}_i$ less than l_i, it needs either the node v_1 or v_2 in the path. Then, the time of accessibility $\widehat{Z}_i^d$ is equal to $\widehat{Z}_{v_1}^r$ if this was not updated in the past iteration with $\widehat{Z}_{v_2}^r$.

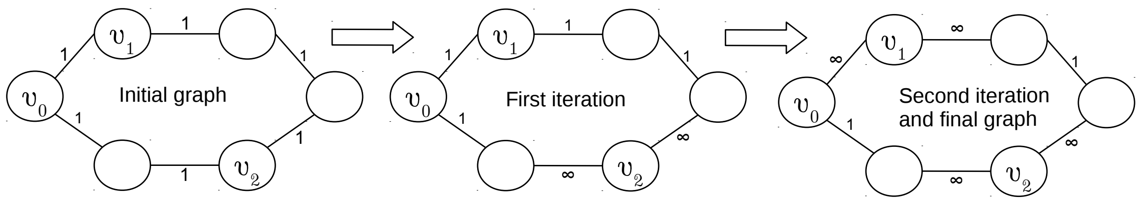
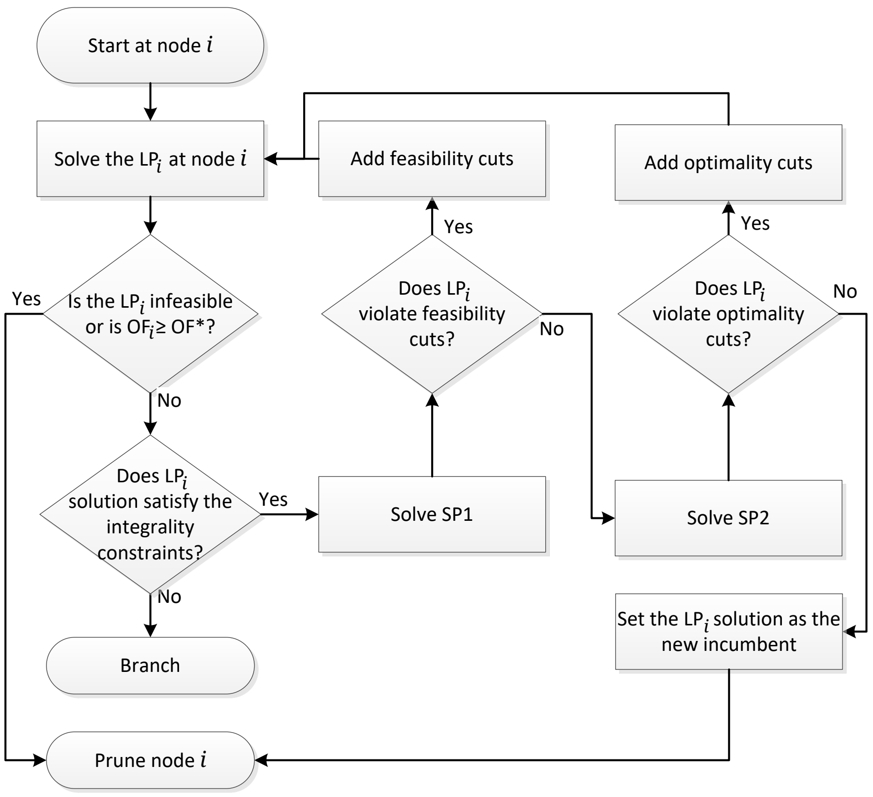


Figure 3: Example of the variation in arc costs in Algorithm 2.

4.5. *Branch-and-Benders-cut*

In the classical Benders decomposition, the MP and the subproblems are solved iteratively in an alternating sequence. At each iteration, the MP is solved to optimality by an MIP solver, and a considerable time may be spent revisiting candidate solutions that have been eliminated in previous iterations (Rahmaniani et al., 2017). On the other hand, in the BBC algorithm, a single search tree is built instead, and the cuts are generated inside the tree using separation routines that seek violated feasibility or optimality cuts (Errico et al., 2017).

Figure 4 shows a flowchart of the BBC method focusing on how the separation routines are used at each node of the branch-and-bound tree. At each node i, we solve the linear relaxation of the current MP, denoted by LP_i. If the LP_i is infeasible or the objective value of the LP_i solution (OF_i) is higher than or equal to the objective value of the current incumbent solution, then node i is pruned. Otherwise, integrality constraints are checked, and if the LP_i solution is not integer feasible, then branching is performed. Every time the LP_i solution is integer feasible, we call the separation routines of the subproblem. First, we solve SP1 and, if SP1 is infeasible, add new feasibility cuts to the MP. If no feasibility cut is obtained, then we solve SP2 to obtain an optimality cut for the MP. If no feasibility or optimality cuts are obtained, then the LP_i solution is feasible for the original problem (1)-(18) and is set as the new incumbent solution. Otherwise, the MP has been modified, LP_i must be resolved, and the described steps are applied again. It is worth mentioning that automatized cuts (for example, Gomory's cuts) and/or heuristics (for example, the relaxation induced neighborhood search (RINS) heuristic) available in commercial solvers can be used at each node of the branch-and-bound tree as well, although they are not included in Figure 4.


Start at node i
Solve the LP_i at node i
Add feasibility cuts
Add optimality cuts
Yes
Yes
Yes
Is the LP_i infeasible or is $\mathrm{OF}_i \geq \mathrm{OF}^*$?
Does LP_i violate feasibility cuts?
No
Does LP_i violate optimality cuts?
No
No
Does LP_i solution satisfy the integrality constraints?
Yes
Solve SP1
Solve SP2
No
Set the LP_i solution as the new incumbent
Branch
Prune node i

Note: OF_i is the objective value of the LP_i solution. OF* is the objective value of the current incumbent.

Figure 4: Flowchart illustrating how the separation routines are used in a given node i of the BBC method.

4.6. *Valid inequalities*

The proposed BBC method also relies on valid inequalities, which are added to the MP and help improve the lower bounds provided by its linear relaxation. The first set of inequalities partially adds information regarding the relation between variables θ_i and R_j. To define the valid inequalities, we first determine the shortest path between the depot and each demand node $i \in \mathcal{V}^d$. For each demand node i, we identify the damaged nodes that are used in a shortest path from the depot to this node. Then, we forbid the use of such damaged nodes in the paths and look for a new shortest path from the depot to the same node i again. If a damaged node j is forbidden and a path with distance less than l_i cannot be found for node i, then we have identified that the damaged node j is necessary to connect the demand node i with the depot. In such case, the accessibility time of node i depends on the repair time of node j and hence we obtain the following valid inequality:

$$\theta_i \geq R_j, \forall\ i \in \mathcal{V}^d,\ j \in \mathcal{Q}_i, \tag{39}$$

in which $\mathcal{Q}_i$ is the set of damaged nodes that must be used to access the demand node i with a distance less than l_i. Recall that R_j was defined as a decision variable of the MP and denotes a lower bound for the exact time at which the node j is repaired in the schedule defined by the variables X_{ij}. Hence, we replace constraints (21) with:

$$R_j \geq R_i + t_{ij} + \delta_j - M \cdot (1 - X_{ij}), \forall\ i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{40}$$

where t_{ij} is the minimum time to travel from node i to node j when no nodes are damaged, which is easily computed using Dijkstra's algorithm.

Let $\mathcal{P} \subset \mathcal{V}^r$ be the subset of damaged nodes that cannot be repaired directly from the depot because they are not accessible without the restoration of other nodes. Using this set, we can define the following valid inequality:

$$\sum_{j \in \mathcal{P}} X_{0j} \leq 0. \tag{41}$$

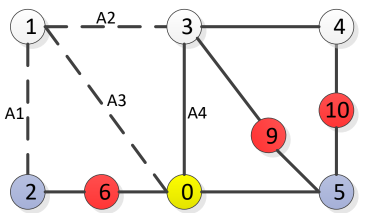
It is also possible to identify the demand nodes that need the repair of at least one damaged node to become accessible. For each demand node, the lower bound for the time instant that it becomes accessible is the travel time plus the repair time of the first node repaired by the crew from the depot. Then, the valid inequality is given by:

$$\theta_i \geq \sum_{j \in \mathcal{V}^r} (t_{0j} + \delta_j) \cdot X_{0j}, \forall\ i \in \mathcal{S}, \tag{42}$$

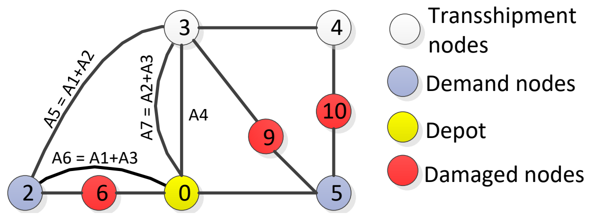
where $\mathcal{S}$ is the subset of demand nodes that require the restoration of at least one damaged node to guarantee they become accessible.

We also propose valid inequalities based on the reduction of the original damaged network of the problem. Let $L \subseteq \mathcal{V}^r$ be a subset of the damaged nodes and $F \subseteq \mathcal{V}^d$ be a subset of the demand nodes in the original graph G. We define G^{LF} as the subgraph obtained from G by deleting all the damaged nodes that are not in L and transforming all the demand nodes that do not belong to F into transshipment nodes . For instance, consider the graph G represented in Figure 1(b) with $\mathcal{V}^r = \{6, 7, 8, 9, 10\}$ and $\mathcal{V}^d = \{2, 4, 5\}$. For $L = \{6, 9, 10\}$ and $F = \{2, 5\}$, the graph G^{LF} is represented in Figure 5(a). To obtain G^{LF}, we removed all the damaged nodes in $\mathcal{V}^r \setminus L$ from G and transformed the demand nodes in $\mathcal{V}^d \setminus F$ into transshipment nodes.

We can further reduce the number of nodes in G^{LF} by removing transshipment nodes that are not directly connected to damaged nodes. For each node i removed from G^{LF}, we delete the arcs adjacent to this node and create new arcs connecting each pair of nodes j and k that were neighbors of i in G^{LF}, such that $j \neq k$. The cost c_{jk} of the new arc $j-k$ is set as $c_{jk} = c_{ji} + c_{ik}$. The resulting graph, denoted by $\bar{G}^{LF}$, is hereafter called as the LF-reduction of G. Figure 5(b) illustrates the graph $\bar{G}^{LF}$ obtained from the LF-reduction of graph G given in Figure 1(b). After obtaining the subgraph G^{LF} presented in Figure 5(a), we obtain $\bar{G}^{LF}$ by deleting node 1 from G^{LF}, as it was not directly connected to any damaged node. Then, we deleted arcs A1, A2 and A3, as they were adjacent to node 1, and created arcs A5, A6, and A7. Notice that either arc A4 or A7 is redundant and hence we can delete the one with the largest cost.



(a) Graph G^{LF}.



(b) Reduced graph $\bar{G}^{LF}$.

Figure 5: Example of a reduction of a damaged network.

From a feasible solution of the CSRP defined using $\bar{G}^{LF}$, we can derive valid inequalities for the original problem, as pointed out in Proposition 4.

Proposition 4. *Given $L \subseteq \mathcal{V}^r$ and $F \subseteq \mathcal{V}^d$, let $K^{\bar{G}^{LF}}$ be an optimal solution of the CSRP defined using the LF-reduction $\bar{G}^{LF}$ of the original graph G. Let $\widehat{\Theta}^{\bar{G}^{LF}}$ be the optimal value and $\widehat{\theta}_i^{\bar{G}^{LF}}$ be the value of the variable Z_i^d in the optimal solution $K^{\bar{G}^{LF}}$, for all $i \in F$. Then, the following inequalities are valid for the MP of the original CSRP defined using the graph G:*

$$\sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} > 0} d_i \cdot \theta_i \geq \widehat{\Theta}^{\bar{G}^{LF}}, \tag{43}$$

$$\Theta \geq \widehat{\Theta}^{\bar{G}^{LF}} + \sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} = 0} d_i \cdot \theta_i + \sum_{i \in \mathcal{V}^d \setminus F} d_i \cdot \theta_i. \tag{44}$$

Proof. Valid inequality (43) is proved by contradiction. Assume that there is a solution K^G of the original CSRP (*i.e.*, using the original graph G) such that

$$\sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} > 0} d_i \cdot \widehat{\theta}_i^G < \widehat{\Theta}^{\bar{G}^{LF}},$$

where $\widehat{\theta}_i^G$ is the value of variable Z_i^d in the solution K^G. Since $L \subseteq \mathcal{V}^r$, graph $\bar{G}^{LF}$ have the same or less damaged nodes than graph G. Hence, a solution for the CSRP defined using $\bar{G}^{LF}$

exists with $\widehat{\theta}_i^{\bar{G}^{LF}} \leq \widehat{\theta}_i^{G}, \forall\, i \in F$. Then,

$$\sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} > 0} d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} \leq \sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} > 0} d_i \cdot \widehat{\theta}_i^{G} < \widehat{\Theta}^{\bar{G}^{LF}},$$

which is a contradiction because $\widehat{\Theta}^{\bar{G}^{LF}}$ is the value of an optimal solution of the CSRP defined using $\bar{G}^{LF}$. Notice that valid inequality (43) remains valid if $\widehat{\Theta}^{\bar{G}^{LF}}$ is a lower bound for the value of the optimal solution related to graph $\bar{G}^{LF}$.

For inequality (44), notice that from constraint (22) of the MP, we have

$$\Theta \geq \sum_{i \in \mathcal{V}^d} d_i \cdot \theta_i = \sum_{i \in F} d_i \cdot \theta_i + \sum_{i \in \mathcal{V}^d \setminus F} d_i \cdot \theta_i = \sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{G^{LF}} > 0} d_i \cdot \theta_i + \sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{G^{LF}} = 0} d_i \cdot \theta_i + \sum_{i \in \mathcal{V}^d \setminus F} d_i \cdot \theta_i.$$

Then, using valid inequality (43), we obtain

$$\Theta \geq \sum_{i \in \mathcal{V}^d} d_i \cdot \theta_i \geq \widehat{\Theta}^{G^{LF}} + \sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{G^{LF}} = 0} d_i \cdot \theta_i + \sum_{i \in \mathcal{V}^d \setminus F} d_i \cdot \theta_i,$$

which proves the valid inequality (44). □

Valid inequalities (43) and (44) can be useful when the value (or a lower bound for the value) of the optimal solution of the CSRP defined using the LF-reduction $\bar{G}^{LF}$ is trivial or can be easily derived. The separation procedure of these valid inequalities is detailed in the following subsection.

4.7. Graph reduction (GR) strategy

Based on Proposition 4, we propose a Graph Reduction (GR) strategy to obtain the LF-reduction of a graph G, as outlined in Algorithm 3. Basically, we create subgraphs $\bar{G}^{LF}$ using a feasible solution of the original CSRP (*i.e.*, using graph G). This feasible solution can be quickly obtained using a heuristic, for example (see Section 4.8). Then, we determine the LF-reduction $\bar{G}^{LF}$, solve the CSRP defined using this subgraph and check if there are violated valid inequalities of type (43) and (44) to be added to the MP of the original CSRP. The idea is to generate sufficiently small subgraphs $\bar{G}^{LF}$, so that the corresponding (reduced) CSRP can be quickly solved.

Let $L = (v_0, \ldots, v_i, \ldots, v_h)$ be a feasible partial sequence of damaged nodes repaired by the crew, where v_i is the ith damaged node repaired by the crew and v_h is the last damaged node repaired in order to make all the demand nodes in the set $\mathcal{V}^d$ accessible. For a given positive integer number $n_1 \leq h$, we partition the set of nodes in the partial schedule L into $P_1 = 1 + \left\lfloor \frac{h-1}{n_1} \right\rfloor$ sets. The sets are labeled from 0 to $P_1 - 1$, where $L_p = \{v_{(1+n_1 \cdot p)}, \ldots, v_{(n_1+n_1 \cdot p)}\}, \forall\, p = 0, \ldots, P_1 - 2$, and $L_{P_1-1} = \{v_{(1+n_1\cdot(P_1-1))}, \ldots, v_h\}$. Similarly, let $F = (u_0, \ldots, u_i, \ldots, u_{|\mathcal{V}^d|})$ be the sequence of demand nodes connected to the depot when the damaged nodes are repaired according to sequence L, where u_i is the ith demand node that becomes accessible. Given a positive integer number $n_2 \leq |\mathcal{V}^d|$, we create a partition of the nodes in F given by $P_2 = 1 + \left\lfloor \frac{|\mathcal{V}^d|-1}{n_2} \right\rfloor$ sets. The sets are labeled from 0 to $P_2 - 1$, where $F_f = \{u_{(1+n_2 \cdot f)}, \ldots, u_{(n_2+n_2 \cdot f)}\}, \forall\, f =$

$0, \dots, P_2 - 2$, and $F_{P_2-1} = \{v_{(1+n_2 \cdot (P_2-1))}, \dots, v_{|\mathcal{V}^d|}\}$. Then, for each $p = 0, \dots, P_1 - 1$ and $f = 0, \dots, P_2 - 1$ we use sets L_p and L_f to obtain the LF-reduction $\bar{G}^{L_p F_f}$ and solve the corresponding (reduced) CSRP to generate valid inequalities (43) and (44).

Algorithm 3 Graph reduction strategy to derive valid inequalities (43) and (44).

```
Input:
Graph $G = (\mathcal{V}, \mathcal{E})$;
Sequences $L = (v_0, \dots, v_i, \dots, v_h)$ and $F = (u_0, \dots, u_i, \dots, u_{|\mathcal{V}^d|})$;
Positive integer numbers $n_1 \leq h$ and $n_2 \leq |\mathcal{V}^d|$;
Output:
Valid inequalities of type (43), (44);
1: for $p = 0$ to $P_1 - 1$ do
2:     for $f = 0$ to $P_2 - 1$ do
3:         Generate subgraph $\bar{G}^{L_p F_f}$;
4:         Solve the CSRP defined using subgraph $\bar{G}^{L_p F_f}$;
5:         Derive the valid inequalities (43), (44) from the solution obtained using $\bar{G}^{L_p F_f}$;
6:     end for
7: end for
8: Add the valid inequalities (43),(44) to the MP.
```

Notice that we are not using an arbitrary selection of damaged nodes to generate the subgraphs $\bar{G}^{L_p F_f}$, but a feasible sequence L. This sequence can be obtained, for example, by using a heuristic able to quickly define a good sequence of damaged nodes to be repaired. This way, we group in a same subgraph, the damaged nodes that are likely to be repaired sequentially in the solution of the original problem (associated with graph G). Also, notice that we do not consider all the damaged nodes but only those enough to make the demand nodes accessible. Similarly, we group in a same subgraph, the demand nodes that are likely to require the restoration of common damaged nodes to become connected to the depot.

4.8. Construction and local search heuristics

In this section, we use a construction heuristic and two local search heuristics with the aim of finding good feasible solutions of the CSRP. The feasible solutions are used as initial incumbent solutions in the BBC algorithm.

4.8.1. Construction heuristic

The crew scheduling decision can be modeled as a traveling salesman problem (TSP) in which the cities to be visited are the damaged nodes. A simple construction heuristic for this problem is a greedy algorithm that makes a locally optimal choice at each iteration in an attempt to find a global optimum. The proposed method starts at the depot and, at each iteration, inserts at the end of the schedule a node that is not in the schedule yet and has the minimum travel time (when no nodes are damaged) to the last inserted node. A node insertion is feasible if this node can be visited without using a node that was not already repaired. Only feasible insertions can be selected at each iteration, and as a consequence, it always generates a feasible schedule. The construction heuristic can also generate feasible random solutions if we insert at the end of the schedule a randomly selected node and not the one with the minimum travel time to the last inserted node.

4.8.2. Local search heuristics

We propose two local search operators with the aim of improving a feasible schedule generated by the construction heuristic. The first local search operator (*swap*) exchanges the positions of two damaged nodes in the schedule. The second local search operator is a pairwise exchange (2-*opt*) that involves removing two edges and replacing them with two different edges that reconnect the fragments created. Let W_K^n be the set of all possible solutions (neighbors) obtained by applying the operator n in the schedule K, where $n \in \{swap, 2\text{-}opt\}$. Let Θ^K be the cost of the schedule K. Let $\widehat{K}_i$ be the ith element of set W_K^n. The local search heuristic based on the two operators is outlined in Algorithm 4. We have a feasible schedule as the input and a locally optimal solution provided by the heuristic as the output. We use subproblems SP1 and SP2 to evaluate the feasibility (line 10) and cost (line 12) of the schedule created when the operators are applied. When a solution better than the current one is found, the local search process is restarted. Furthermore, when a set W_K^n is fully explored, we restart the algorithm from a random solution (line 28). The algorithm terminates when no improvement is found for the last randomly generated solution.

5. Computational experiments

In this section, we evaluate the performance of the proposed solution approach using instances from the literature. All the algorithms were implemented in C++ programming language. The BBC method was implemented on top of the IBM CPLEX Optimization Solver 12.7 using the Concert Technology library. We implemented the specialized algorithms to solve subproblems SP1 and SP2 and the heuristics according to their descriptions in Sections 4.4 and 4.8. All cuts and valid inequalities are added to problem using the Callback procedures available in the Concert Technology library. The experiments were run on a Linux PC with a CPU Intel Core i7 3.4 GHz and 16.0 GB of memory using a single thread. The stopping criteria was either the elapsed time exceeding the time limit of 3,600 seconds or the optimality gap being smaller than 10^{-4}. All the remaining parameters of CPLEX were kept at their default values.

5.1. Instance description

We carried out computational experiments using two types of theoretical instances: S1, which is composed of small instances, and S2, which is composed of medium and large instances, as presented by Maya-Duque et al. (2016). As described by the authors, they generated networks with different numbers of nodes and arcs based on the instance generator proposed by Klingman et al. (1974). Table 1 shows the characteristics of the set of instances. The type (S1 or S2), network name (class), number of demand nodes, and the total number of nodes and arcs in the original network can be seen in columns 1 to 5 of Table 1, respectively. For each original network, one class of instances was generated by varying two parameters, namely, α and β. Parameter α defines the percentage of damaged arcs in the network. Parameter β specifies the maximum tolerable percentage by which the paths connecting demand nodes to the depot can increase in relation to the shortest paths in the network when no damaged node exists. For example, $\alpha = 50\%$ indicates that half of the arcs of the original network were damaged, and $\beta = 50\%$

Algorithm 4 Local search heuristic using the operator $n \in \{swap, 2\text{-}opt\}$.

```
Input:
Schedule $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$; Cost $\Theta^K$ of scheduling $K$;
Output:
Schedule $K^* = (v_0^*, v_1^*, ..., v_j^*, ..., v_{|\mathcal{V}^r|}^*)$;
 1: $\Theta^{K^*} := \Theta^{\widehat{K}_i}$; $K^* := K$;
 2: Determine set $W_K^n$;
 3: improvement_global := 1;
 4: while improvement_global = 1 do
 5:     improvement_global := 0; improvement_local := 1;
 6:     while improvement_local = 1 do
 7:         $i := 1$;
 8:         improvement_local := 0;
 9:         while $i \leq |W_K^n|$ do
10:             Evaluate feasibility of schedule $\widehat{K}_i$ by solving subproblem SP1;
11:             if schedule $\widehat{K}_i$ is feasible then
12:                 Calculate cost $\Theta^{\widehat{K}_i}$ of schedule $\widehat{K}_i$ by solving subproblem SP2;
13:                 if $\Theta^{\widehat{K}_i} < \Theta^K$ then
14:                     $\Theta^K := \Theta^{\widehat{K}_i}$; $K := \widehat{K}_i$;
15:                     $i := |W_K^n| + 1$;
16:                     Determine new set $W_K^n$;
17:                     improvement_local := 1;
18:                     if $\Theta^{\widehat{K}} < \Theta^{K^*}$ then
19:                         $\Theta^{K^*} := \Theta^{\widehat{K}}$; $K^* := K$;
20:                         improvement_global := 1;
21:                     end if
22:                 end if
23:             end if
24:             $i := i + 1$;
25:         end while
26:     end while
27:     if improvement_global = 1 then
28:         Find a new random solution $K$ with the construction heuristic;
29:         Determine the new set $W_K^n$;
30:     end if
31: end while
```

indicates that the maximum distance l_i for the paths between the depot and the demand node i is 1.5 times the length of the shortest path between the depot and the node i when no damaged node exists. Columns 6 and 7 of Table 1 show the values of α and β, respectively.

For each damaged arc in the original network, one or more damaged nodes are added in the middle of the arc. Therefore, the total numbers of nodes and arcs in the instance depend on the parameter α. For example, the original network in Figure 1(a) with 6 nodes and 9 arcs is transformed into the damaged network in Figure 1(b) with 10 nodes and 14 arcs. In the table, original network 1 with 25 nodes and 40 arcs is transformed into a damaged network with 27 nodes and 42 arcs when $\alpha = 5\%$ (the 2 damaged arcs are converted into 2 damaged nodes) and into a damaged network with 45 nodes and 60 arcs when $\alpha = 50\%$ (20 new damaged nodes). Thus, damaged networks generated from original network 1 have 27 to 45 nodes and 40 to 60 arcs. Columns 8 and 9 show the total number of nodes and arcs in the damaged networks. By combining the values of α and β for original network 1, for example, 20 instances were generated. For original network 16, the values of $\alpha = 5\%, 25\%, 50\%$ were combined with $\beta = 5\%, 10\%$ to form 6 instances, while the values of $\alpha = 10\%, 30\%$ were combined with $\beta = 25\%, 50\%$ to form 4 instances. For networks 1-15, the number of instances generated is 20. For networks 16-39, the number of instances generated is 10.

It is worth mentioning that some of the large instances in group S2 are actually much larger than the practical instances we typically find in real-world situations. Feng and Wang (2003), for example, considered a real network with 10 damaged points and less than 100 total nodes. Yan and Shih (2007) and Yan and Shih (2009) also considered real networks with less than 100 nodes but with 24 damaged points. Similarly, Xu and Song (2015) considered a real case with 36 damaged nodes and not more than 100 total nodes. Pramudita and Taniguchi (2014) considered a larger real damaged network with 98 damaged points (blocked arcs) and 198 total nodes. Finally, Akbari and Salman (2017a) considered one of the largest practical cases in the literature, involving networks with 240 damaged points, 349 nodes and 689 arcs. Note that we are considering instances with up to 312 damaged nodes, 712 total nodes and 937 total arcs.

Table 1: Set of instances.

	Network	Demand	Original network						Damaged network		Total	
Type	(class)	nodes	nodes	arcs	Values for α (%)		Values for β (%)		nodes	arcs	instances	
S1	1	19	25	40	5, 10, 25, 30, 50		5, 10, 25, 50		27 to 45	42 to 60	20	
S1	2	19	25	37	5, 10, 25, 30, 50		5, 10, 25, 50		26 to 43	38 to 55	20	
S1	3	19	25	39	5, 10, 25, 30, 50		5, 10, 25, 50		26 to 44	40 to 58	20	
S1	4	24	30	83	5, 10, 25, 30, 50		5, 10, 25, 50		34 to 71	87 to 124	20	
S1	5	24	30	89	5, 10, 25, 30, 50		5, 10, 25, 50		34 to 74	93 to 133	20	
S1	6	24	30	84	5, 10, 25, 30, 50		5, 10, 25, 50		34 to 72	88 to 126	20	
S1	7	28	35	118	5, 10, 25, 30, 50		5, 10, 25, 50		40 to 94	123 to 177	20	
S1	8	28	35	115	5, 10, 25, 30, 50		5, 10, 25, 50		40 to 92	120 to 172	20	
S1	9	28	35	113	5, 10, 25, 30, 50		5, 10, 25, 50		40 to 91	118 to 169	20	
S1	10	15	20	39	5, 10, 25, 30, 50		5, 10, 25, 50		21 to 39	40 to 58	20	
S1	11	15	20	37	5, 10, 25, 30, 50		5, 10, 25, 50		21 to 38	38 to 55	20	
S1	12	15	20	37	5, 10, 25, 30, 50		5, 10, 25, 50		21 to 38	38 to 55	20	
S1	13	35	40	146	5, 10, 25, 30, 50		5, 10, 25, 50		47 to 113	153 to 219	20	
S1	14	35	40	143	5, 10, 25, 30, 50		5, 10, 25, 50		47 to 111	150 to 214	20	
S1	15	35	40	143	5, 10, 25, 30, 50		5, 10, 25, 50		47 to 111	150 to 214	20	
S2	16	50	60	191	5, 25, 50	10, 30	05, 10	25, 50	69 to 155	200 to 286	6	4
S2	17	50	60	197	5, 25, 50	10, 30	25, 50	05, 10	69 to 158	206 to 295	6	4
S2	18	50	60	196	5, 25, 50	10, 30	05, 10	25, 50	69 to 158	205 to 294	6	4
S2	19	70	80	247	5, 25, 50	10, 30	25, 50	05, 10	92 to 203	259 to 370	6	4
S2	20	70	80	245	5, 25, 50	10, 30	05, 10	25, 50	92 to 202	257 to 367	6	4
S2	21	70	80	248	5, 25, 50	10, 30	25, 50	05, 10	92 to 204	260 to 372	6	4
S2	22	90	100	274	5, 25, 50	10, 30	05, 10	25, 50	113 to 237	287 to 411	6	4
S2	23	90	100	271	5, 25, 50	10, 30	25, 50	05, 10	113 to 235	284 to 406	6	4
S2	24	90	100	273	5, 25, 50	10, 30	05, 10	25, 50	113 to 236	286 to 409	6	4
S2	25	125	140	324	5, 25, 50	10, 30	25, 50	05, 10	156 to 302	340 to 486	6	4
S2	26	125	140	323	5, 25, 50	10, 30	05, 10	25, 50	156 to 301	339 to 484	6	4
S2	27	125	140	322	5, 25, 50	10, 30	25, 50	05, 10	156 to 301	338 to 483	6	4
S2	28	140	170	398	5, 25, 50	10, 30	05, 10	25, 50	189 to 369	417 to 597	6	4
S2	29	140	170	399	5, 25, 50	10, 30	25, 50	05, 10	189 to 369	418 to 598	6	4
S2	30	140	170	396	5, 25, 50	10, 30	05, 10	25, 50	189 to 368	415 to 594	6	4
S2	31	200	200	447	5, 25, 50	10, 30	25, 50	05, 10	222 to 423	469 to 670	6	4
S2	32	200	200	449	5, 25, 50	10, 30	05, 10	25, 50	222 to 424	471 to 673	6	4
S2	33	200	200	449	5, 25, 50	10, 30	25, 50	05, 10	222 to 424	471 to 673	6	4
S2	34	300	300	524	5, 25, 50	10, 30	05, 10	25, 50	326 to 562	550 to 786	6	4
S2	35	300	300	525	5, 25, 50	10, 30	25, 50	05, 10	326 to 562	551 to 787	6	4
S2	36	300	300	525	5, 25, 50	10, 30	05, 10	25, 50	326 to 562	551 to 787	6	4
S2	37	400	400	625	5, 25, 50	10, 30	25, 50	05, 10	431 to 712	656 to 937	6	4
S2	38	400	400	625	5, 25, 50	10, 30	05, 10	25, 50	431 to 712	656 to 937	6	4
S2	39	400	400	625	5, 25, 50	10, 30	25, 50	05, 10	431 to 712	656 to 937	6	4
Total											540	

5.2. Description of experiments

In this section, we present a description of the computational experiments. Table 2 presents the combination and stopping criteria of eleven proposed solution strategies, in which ET refers to elapsed time and TL to the total time limit. For instance, H3 is the heuristic strategy that uses first the construction heuristic, then the local search $2opt$, and finally the local search *swap*, either with stopping criteria given by time limit or a locally optimal solution found. The first four solution strategies (H1-H4) use only the construction and local search heuristics presented in Section 4.8. The following five strategies (BBC1-BBC5) are variants of the Branch-and-Benders-Cut (BBC) algorithm that use different combinations of the cuts presented in Section 4.3 and some of the valid inequalities presented in Section 4.6. The same separation algorithms are used in all the BBC strategies to identify the feasibility and cost of a scheduling solution of the MP. Then, we enumerate and add inequalities to the MP according to the type of cuts used in each BBC strategy. The algorithm BBC6 relies on the best heuristic method to provide a good feasible initial solution to the best BBC method. The best solution found with the heuristic is set as the incumbent solution of the MP. In the GR-BBC6 method, the algorithm BBC6 is combined with the Graph Reduction (GR) strategy presented in Section 4.7 to derive valid inequalities (39)-(44). BBC6 is used to solve the reduced CSRP defined using the subgraphs $\bar{G}^{L_pF_f}$, which are generated using solutions provided by the heuristic H3, considering sets of $n_1 = 20$ damaged nodes and $n_2 = |\mathcal{V}^d|$ demand nodes (see Algorithm 4). The time limit to solve the reduced CSRP is 60 seconds. If the reduced CSRP for a given subgraph $\bar{G}^{L_pF_f}$ is not solved to optimality within this time limit, we reduce this subgraph by removing only the first five nodes of sets L_p and F_f and then we solve the corresponding reduced CSRP again. Finally, the MIP model presented in Section 4.1 is used to solve the problem.

Table 2: Characteristic of the solution methods.

Solution strategy	Combination (stopping criteria)[1]
H1	Construction heuristic + *2opt* ($ET > TL$ or locally optimal).
H2	Construction heuristic + *swap* ($ET > TL$ or locally optimal).
H3	Construction heuristic + *2opt* ($ET > \frac{1}{2}TL$ or locally optimal) + *swap* ($ET > TL$ or locally optimal).
H4	Construction heuristic + *swap* ($ET > \frac{1}{2}TL$ or locally optimal) + *2opt* ($ET > TL$ or locally optimal).
BBC1	BBC algorithm with valid inequalities (39) - (42), feasibility cut (33) and optimality multi-cuts (35) and (36) ($ET > TL$ or gap $= 0$).
BBC2	BBC algorithm with valid inequalities (39) - (42), feasibility cut (32) and optimality cut (34) ($ET > TL$ or gap $= 0$).
BBC3	BBC algorithm with valid inequalities (39) - (42), feasibility cut (32) and optimality multi-cuts (36) ($ET > TL$ or gap $= 0$).
BBC4	BBC algorithm with valid inequalities (39) - (42), feasibility cut (32) and optimality multi-cuts (35) and (36) ($ET > TL$ or gap $= 0$).
BBC5	BBC algorithm without valid inequalities (39) - (42), feasibility cut (32) and optimality multi-cuts (35) and (36) ($ET > TL$ or gap $= 0$).
BBC6	H3 ($ET > \frac{1}{6}TL$ or locally optimal) + BBC4 ($ET > TL$ or gap $= 0$).
GR-BBC6	H3 ($ET > \frac{1}{6}TL$ or locally optimal) + GR ($ET > \frac{1}{60}TL$ each subproblem or optimality of the subproblems) + BBC4 with additional valid inequalities (43) - (44) ($ET > TL$ or gap $= 0$).
MIP model	Model (1)-(18) ($ET > TL$ or gap $= 0$)

[1] Let TL be the total time limit and ET be the elapsed time.

The solution methods were evaluated using performance profiles as proposed by Dolan and Moré (2002). Given a set $\mathcal{P}$ of instances and a set $\mathcal{F}$ of solution methods, performance profiles are based on the cumulative distribution function $P(f, q)$, which indicates the probability of a strategy f with a $\log_2$ performance ratio being within a factor $q \in R$ of the best possible ratio. The function $P(f, q)$ is defined as:

$$P(f,q) = \frac{|\{p \in \mathcal{P} : \log_2(v(p,f)) \leq q\}|}{|\mathcal{P}|}\ ,\ q \geq 0, \tag{45}$$

$$\text{with } v(p,f) = \frac{TC_{pf}}{min\{TC_{pf} : f \in F\}}, \tag{46}$$

where $|\mathcal{P}|$ is the total number of instances and TC_{pf} is the performance measure (objective function cost, gap or elapsed time) of problem p when solved by method f. Values of $P(f,q)$ when $q = 0$ indicate the fraction of instances for which the strategy reached the best solution. For $q > 0$, $P(f,q)$ is the fraction of instances for which strategy f obtained solutions with a quality within a factor of 2^q of the best solutions. Values of q when $P(f,q) = 1$ indicate that quality of the solutions obtained by strategy f for all instances are within a factor of 2^q of the best solutions.

5.3. Computational performance of the proposed approaches

To evaluate the performance of the heuristic approaches, we use the objective value of the solutions found within a time limit of 3,600 seconds. The heuristic algorithms do not provide a lower bound for the objective value, so we cannot calculate the optimality gap. On the other hand, the BBC approaches provide upper- and lower-bound values, so we use the optimality gap provided by the algorithms within a time limit of 3,600 seconds as well to compare the BBC approaches. The optimality gap is computed as:

$$gap = \frac{Z^U - Z^L}{Z^U}, \tag{47}$$

in which Z^U is the upper bound or best integer solution and Z^L is the lower bound. The optimality gap is a good indicator of the quality of the methods because it considers simultaneously the upper and lower bound of the solutions. However, we also compared the upper bounds of the BBC strategies, and the overall results were similar to those obtained with the optimality gaps.

Figure 6 shows the performance profiles for the heuristic strategies (H1-H4) using the objective value. The results indicate that the two strategies that combine the local search heuristics *swap* and 2*opt*, H3 and H4, yield a more stable performance than the others. Strategy H3 (H4) found the smallest objective function cost for 80.18% (74.44%) of the instances, and in the remaining instances, H3 (H4) provides a solution with cost within a factor of $2^{0.61} \approx 1.53$ ($2^{0.78} \approx 1.72$) of the lowest cost found. Due to this behavior, H3 was selected as the best heuristic strategy.

Figure 7 presents the performance profiles for the BBC algorithms (BBC1-BBC5) based on the optimality gap. Table 3 shows the extreme values of the performance profiles for the BBC strategies. The performance profiles reveal that the majority of the strategies have similar results in 90% ($P(f,q) = 0.9$) of the instances. As expected, strategy BBC5, which does not use any valid inequalities, presents the worst performance. In most instances, the valid inequalities improved the lower bound, thus accelerating the convergence of the algorithms. In fact, while BBC5 found the best gap only for 35.5% of the instances, BBC4 found the best gap for 64.8% of the instances, which represents a substantial improvement of 82%. Furthermore, for instances in which the best gap is not achieved, BBC4 provides solutions with a gap within a factor of $2^{3.45} \approx 11$ of the best gap, while for the BBC5 algorithm, the factor is $2^{7.66} \approx 202$.

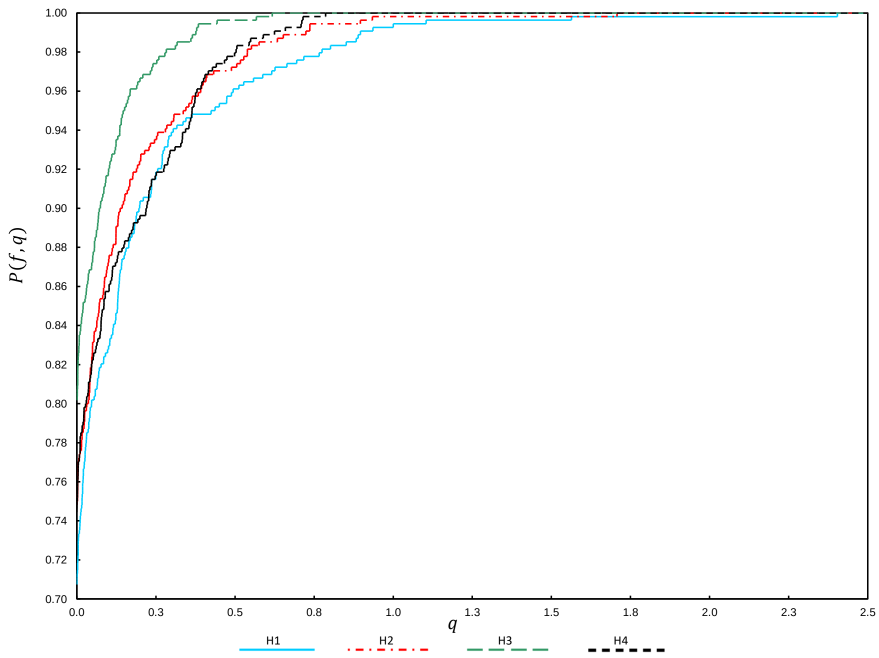


Figure 6: Performance profiles of the heuristic methods based on the objective value.

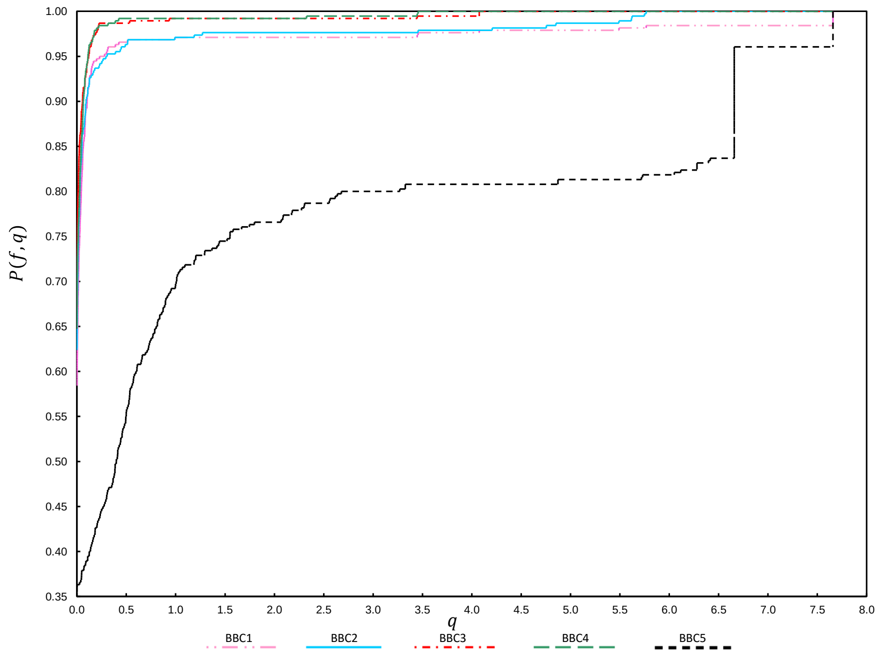


Figure 7: Performance profiles of the BBC algorithms based on the optimality gap.

Table 3: Extreme values of the performance profiles for the BBC strategies.

BBC strategy	$P(f,q)$[1]	q[2]
BBC1	0.5842	7.6582
BBC2	0.6237	5.7664
BBC3	0.6411	4.0752
BBC4	0.6474	3.4558
BBC5	0.3553	7.6582

[1] Values of $P(f,q)$ when $q = 0$.
[2] Values of q when $P(f,q) = 1$.

By comparing the performance profiles of algorithms BBC1 and BBC4, it is possible to see that using feasibility cut (32) is better than using feasibility cut (33). This result was also expected because equation (32) cuts off a larger number of infeasible solutions when it is used. Using multiple lower bound functions as optimality cuts appears to be more efficient than using single cuts, which can be deduced from the comparison among algorithms BBC2, BBC3 and BBC4. Notice that the optimality multi-cut approaches (BBC3 and BBC4) are faster and more stable than the optimality single-cut approach (BBC2). Finally, from the comparison of algorithms BBC3 and BBC4, we can conclude that the use of cut (35) improves the convergence of the method. Optimality multi-cut (35) helps set a lower bound for a greater number of solutions than multi-cut (36) individually. Therefore, the algorithm BBC4 is selected as the best strategy, which provides the smallest gap for 64.73% of the instances and, in the remaining instances, provides solutions with a gap within a factor of $2^{3.45} \approx 11$ of the best gap obtained.

Approach BBC6 combines the best heuristic and BBC strategies, H3 and BBC4, respectively. GR-BBC6 combines BBC6 with the GR strategy. We build performance profiles based on the gap provided by the algorithms within the time limit of 3,600 seconds to compare BBC4, BBC6, GR-BBC6, and the MIP model. As we can see in Figure 8, not surprisingly, the BBC algorithms outperform the MIP model. In fact, the mathematical model found feasible solutions for only 45.8% of the instances. BBC6 shows a more stable performance than the BBC4 algorithm. Thus, starting the BBC with an initial solution provided by heuristic H3 improves the performance of the BBC algorithm. By comparing GR-BBC6 and BBC6, we can infer that the valid inequalities (43)−(44) derived by the GR strategy are effective to improve the convergence of the method. GR-BBC6 (BBC6) achieved the best gap in 96.1% (50.78%) of the instances and, for the instances it was not achieved, the solution gap was within a factor of $2^{4.68} \approx 25.63(2^{6.12} \approx 69.55)$ of the best gap obtained.

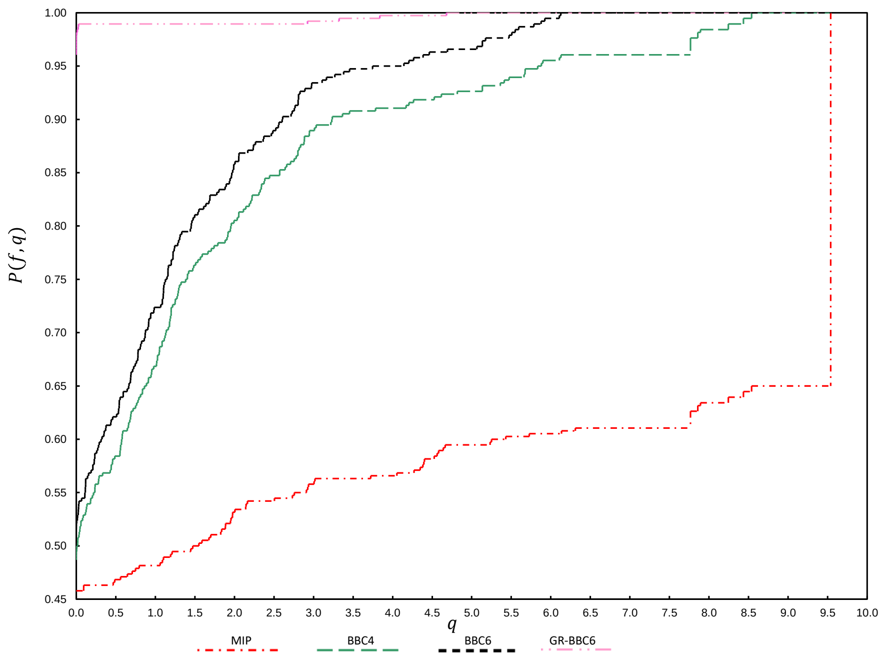


Figure 8: Performance profiles of the best BBC strategies and the MIP model based on the optimality gap.

5.4. Performance of the best strategy

Table 4 shows the average upper bound, gap and elapsed time of the best strategy proposed in this paper for solving the instances of group S1 and S2. For all instances, the GR-BBC6 method provided feasible solutions within 3,600 seconds. The average total elapsed time was 2,361 seconds, and the average time that GR-BBC6 spent to find the best upper bound was 797.3 seconds, 66.24% smaller than the elapsed time. Thus, GR-BBC6 finds feasible solutions relatively quickly, and most of the elapsed time is consumed to improve the lower-bound values. The average gap considering all instances was 38.82%. For instances S1, the average gap was 9.32%, while for instances S2, the average gap was 57.26%. As expected, worse gaps are obtained for instances with a large number of nodes and arcs. If we consider only practical size instances (according to most of the real-world cases presented in the literature) we could limit our experiments to the first 24 class of instances, obtaining an average gap of 18.17%. Therefore, the GR-BBC6 algorithm is effective to solve practical size instances with good quality solutions.

Table 4: Average results of the GR-BBC6 strategy.

Type	Network (Class)	Avg. upper bound	Avg. gap (%)	Avg. elapsed time (sec.)	Avg. best[1] time (sec.)
S1	1	9,744.98	2.36	720.02	0.06
S1	2	34,088.95	5.04	1,039.29	0.05
S1	3	49,862.45	2.47	844.10	427.62
S1	4	18,037.43	8.24	1,440.08	2.51
S1	5	18,484.94	8.76	1,440.36	518.32
S1	6	20,917.01	12.51	1,618.59	260.78
S1	7	36,511.03	6.71	2,177.40	14.25
S1	8	26,048.79	13.94	1,956.36	66.38
S1	9	33,953.25	21.84	1,580.71	21.32
S1	10	48,459.97	2.42	725.84	487.55
S1	11	38,538.07	3.61	798.72	0.09
S1	12	28,036.81	1.76	723.84	6.21
S1	13	23,566.08	9.64	2,139.00	19.63
S1	14	81,031.99	20.28	2,119.10	351.12
S1	15	52,200.77	20.21	2,138.85	8.85
S2	16	38,737.05	18.09	2,074.49	628.29
S2	17	30,448.01	16.56	2,259.61	391.54
S2	18	97,476.41	21.39	2,092.61	569.09
S2	19	65,092.84	33.02	2,160.52	1,186.22
S2	20	71,172.82	38.95	2,550.34	697.30
S2	21	75,602.30	40.20	2,520.80	1,190.12
S2	22	211,486.51	41.05	2,883.30	1,522.30
S2	23	98,827.21	41.49	2,880.07	1,129.43
S2	24	209,434.98	45.56	2,880.88	1,975.86
S2	25	155,240.79	49.76	2,880.24	1,127.12
S2	26	274,847.22	62.94	2,961.89	1,756.10
S2	27	163,429.50	53.77	2,880.10	1,267.61
S2	28	435,199.83	58.23	2,881.21	1,495.73
S2	29	247,954.95	65.94	2,880.62	1,311.75
S2	30	468,494.13	57.76	3,252.19	1,643.45
S2	31	314,909.80	70.55	2,880.80	1,232.92
S2	32	337,827.97	61.12	2,881.92	1,224.98
S2	33	275,998.35	65.94	3,240.33	912.47
S2	34	481,122.55	88.49	3,600.03	1,097.02
S2	35	472,126.95	84.17	3,600.05	1,704.58
S2	36	497,125.76	86.16	3,600.03	1,311.96
S2	37	532,181.82	91.50	3,600.16	988.37
S2	38	697,781.81	91.87	3,600.05	1,433.22
S2	39	553,372.93	89.66	3,600.05	1,112.55
Avg. All		187,830.13	38.82	2,361.66	797.30
Avg. S1		34,632.17	9.32	1,430.82	145.65
Avg. S2		283,578.85	57.26	2,943.43	1,204.58

[1] Time that GR-BBC6 spent to find the best upper bound.

We also performed additional experiments increasing the time limit of the GR-BBC6 algorithm to 24 hours. We considered the classes of instances 12 and 38, which present the smallest and the largest optimality gaps according to Table 4. The results reveal that the gap is reduced from 1.76% to 1.21% (31.25%) in class 12, and from 91.87% to 86.89% (5.42%) in class 38. Thus, our approach can be more effective for longer computational times, although the gap improvement can be rather negligible from the practical point of view.

Table 5 shows the average gap of the GR-BBC6 method according to different values of α and β. For example, the value 16.65 in bold in the table indicates the average gap for all instances with $\alpha = 10\%$ and $\beta = 5\%$. Note that the instances become more challenging when the percentage of damage (α) increases, as expected. In fact, more damaged nodes lead to (possibly) more crew schedules to be evaluated in the MP, slowing down the convergence of the method. More nodes in the network also makes the resolution of the subproblems even harder. The GR-BBC6 strategy found solutions with an average gap of 7.76% for the instances with $\alpha = 5\%$, and an average gap of 55.55% for instances with $\alpha = 50\%$. Similarly, the difficulty of the instances decreases (on average) when the maximum tolerable percentage (β) increases. Higher values of β make it easier for subproblem SP2 to find a feasible path between the depot and the demand nodes. The average gap for instances with $\beta = 5\%$ is 31.92%, while the average gap for instances with $\beta = 50\%$ is 29.10%. It is worth mentioning that, in most of the practical situations, no more than $\alpha = 30\%$ of the roads are considered as damaged roads. For instance, Akbari and Salman (2017a), which addressed one of the largest practical cases in the literature, considered 33% of the roads as damaged.

Table 5: Average gap for each value of α and β.

		α (%)					
		5	10	25	30	50	Avg.
β (%)	5	8.51	**16.65**	38.10	39.72	56.64	31.92
	10	7.59	16.33	34.11	40.85	53.85	30.55
	25	7.58	15.63	35.92	39.44	55.84	30.88
	50	7.34	12.32	32.64	37.36	55.83	29.10
	Avg.	7.76	15.23	35.19	39.34	55.55	30.61

5.5. *Comparison with other results from the literature*

This section compares the results obtained by the GR-BBC6 strategy with the results of other approaches available in the literature, namely, the dynamic programming (DP) algorithm and the iterated greedy-randomized constructive procedure (IGRCP) metaheuristic, both of which were proposed by Maya-Duque et al. (2016). While the DP approach is also an exact method analogous to our GR-BBC6 method, the IGCRP is a metaheuristic and hence has no guarantee of optimality. This metaheuristic is based on the greedy randomized adaptive search procedure (GRASP) and consists of two phases: the construction of a feasible solution and an improvement in the constructed solution, including multiple runs of the construction phase after the improvement routine. Our BBC algorithm is the first exact method proposed in literature able to find a lower bound for all the considered instances. Thus, it is not possible to perform any comparison of lower bounds using other approaches from the literature. This way, we only compare the solution costs (upper bounds) provided by the BBC with the costs delivered by the other approaches. We emphasize that the purpose is not to compare the methods, but to verify the quality of the solutions provided by the GR-BBC6 method. We show the results only for instances in group S1 (small instances) because the DP strategy proposed in Maya-Duque et al. (2016) is not able to solve medium and large instances. The IGRCP metaheuristic, on the other hand, was used to solve S2 instances in Maya-Duque et al. (2016), but we did not have access to those solutions until the submission of the paper.

Table 6 shows the average upper bound and elapsed time of the three approaches for instances in group S1. The character "–" indicates that no solution was obtained for one or more instances of the class. The last column "ratio" shows the ratio of the upper bound of IGRCP in relation to the upper bound of GR-BBC6. Ratios smaller than 1 indicate that the GR-BBC6 strategy improves the upper bound found by the IGRCP metaheuristic. The columns "# optimal" show the number of optimal solutions found by each exact method. The DP algorithm solved all the instances to optimality for classes of instances corresponding to networks 1, 2 and 10. For the other classes, the DP algorithm did not solve some of the instances within a time limit of 24 hours, especially those with $\alpha = 50$. For instances of classes corresponding to networks 1, 2 and 10, the solutions of the GR-BBC6 method were equal to the solutions of the DP strategy, indicating that these solutions are optimal, although there is a nonzero gap related to the lower bound computed with the GR-BBC6 method.

The BBC is able to prove optimality in (181) 60.33% of the small instances, while the best exact approach available so far in the literature proved optimality for (160) 53.33% of the small instances. Furthermore, our BBC algorithm solves to optimality 18.33% of the medium and large instances, while medium and large instances were not solved with the DP. In terms of computational times, the DP strategy was slower than the GR-BBC6 strategy for instances in classes 1, 2 and 10.

Table 6: Average result of the solution methods for small instances.

	Avg. upper bound			# optimal		Avg. elapsed time (sec.)			Avg. best[4]	
Class	IGRCP[1]	DP[2]	GR-BBC6[3]	DP	GR-BBC6	IGRCP	DP	GR-BBC6	time (sec.)	Ratio
1	9,744.98	9,744.98	9,744.98	20	16	1.09	2,945.31	720.02	0.06	1.000
2	34,088.95	34,092.54	34,088.95	20	16	1.43	8,733.50	1,039.29	0.05	1.000
3	49,987.07	–	49,862.45	17	16	1.60	–	844.10	427.62	0.998
4	18,246.59	–	18,037.43	16	11	7.04	–	1,440.08	2.51	0.989
5	18,151.81	–	18,484.94	12	11	14.10	–	1,440.36	518.32	1.018
6	21,253.39	–	20,917.01	1	11	17.98	–	1,618.59	260.78	0.984
7	36,873.30	–	36,511.03	0	8	11.77	–	2,177.40	14.25	0.990
8	26,382.27	–	26,048.79	0	9	38.37	–	1,956.36	66.38	0.987
9	35,223.52	–	33,953.25	0	11	20.26	–	1,580.71	21.32	0.964
10	48,545.84	48,460.28	48,459.97	20	16	0.91	16,663.44	725.84	487.55	0.998
11	39,212.63	–	38,538.07	17	16	1.59	–	798.72	0.09	0.983
12	28,876.04	–	28,036.81	16	16	0.87	–	723.84	6.21	0.971
13	23,535.63	–	23,566.08	8	8	7.83	–	2,139.00	19.63	1.001
14	87,163.33	–	81,031.99	8	8	91.47	–	2,119.10	351.12	0.930
15	52,085.29	–	52,200.77	5	8	53.51	–	2,138.85	8.85	1.002
Average	35,291.38	–	34,632.17	10.67	12.07	17.99	–	1,430.82	145.65	0.988

[1] Metaheuristic based on GRASP proposed in Maya-Duque et al. (2016)
[2] Exact dynamic programming algorithm proposed in Maya-Duque et al. (2016)
[3] Best BBC strategy (1 hour time limit).
[4] Time that GR-BBC6 spent to find the best upper bound.

On average, the solutions provided by the BBC approach GR-BBC6 are better than the solutions provided by the IGRCP heuristic. Additionally, the BBC6 method provided a lower bound and an optimality gap for all the solutions within a time limit of 3,600 seconds. Thus, the BBC can obtain a valid lower bound for all the instances without deteriorate the cost (upper bound) of the solutions or even at improving the upper bound of the solutions. As expected, the IGRCP metaheuristic was the fastest method but gave no guarantee of optimality, as it corresponds to a heuristic method. Note that most of the time spent by the BBC strategy is to improve the lower bound. In fact, the average time spent by GGR-BBC6 to find the best upper bound is 145.65 seconds, 10 times smaller than the average elapsed time.

6. Conclusions and future research

This paper explored branch-and-Benders-cut (BBC) approaches to solve the crew scheduling and routing problem (CSRP), in the context of road restoration. As a key contribution, it developed the first exact solution approach that is able to obtain feasible solutions and lower bounds for all instances from the literature, including very large-scale instances. The addressed problem is typically found in post-disaster situations where the damaged network must be repaired as quickly as possible to promote an effective response. The joint presence of scheduling and routing decisions explains the complexity of solving such problems, for which commercial solvers cannot be efficiently used. Thus, we have devised approaches based on the Benders decomposition, applied to a MIP formulation that determines a fair and efficient road restoration plan. We employed feasibility cuts, multiple optimality cuts, and specialized valid inequalities, which have enhanced the performance of the BBC approaches. The use of simple heuristics to provide initial incumbent solutions for the master problem was also an important strategy to accelerate the convergence of the methods. The proposed BBC strategies have improved the results of exact and heuristic methods proposed so far in the literature. In fact, our best approach has proven the optimality of 41.67% of the instances, and for 100% of the instances, it obtained valid lower bounds for the first time. It is worth noting that we have not found any other computational study that considers so many nodes and arcs for any variant of the CSRP in road restoration. The major remaining obstacle, though, is to provide the optimality certificate for some large-scale instances. In this sense, we would like to investigate particular properties and characteristics of the problem to derive new valid inequalities and different ways for decomposing the MIP formulation. Finally, it would be of interest to improve the lower bounds of the current solutions and to develop hybrid methods combining exact and metaheuristic strategies to obtain tighter solutions.

Acknowledgements

This work was supported by the São Paulo Research Foundation (FAPESP) [grant numbers 2015/26453-7, 2016/15966-6 and 2016/23366-9]; and the National Council for Scientific and Technological Development (CNPq) [grant numbers 141973/2016-1 and 304601/2017-9]. The third author thanks the support of both the British Academy via BA/Leverhulme Small Research Grant SRG18R1\180939 and the UEBS First Grant Venture Fund. The authors are grateful to the two anonymous referees for their valuable suggestions that helped to improve the manuscript.

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

Akbari, V., Salman, F. S., 2017b. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. European Joural of Operational Research 257 (2), 625–640.

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.

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

Costa, A. M., 2005. A survey on Benders decomposition applied to fixed-charge network design problems. Computers & Operations Research 32 (6), 1429–1450.

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operational Research Society of America 2 (4), 393–410.

de Sá, E. M., de Camargo, R. S., de Miranda, G., 2013. An improved Benders decomposition algorithm for the tree of hubs location problem. European Journal of Operational Research 226 (2), 185–202.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1 (1), 269–271.

Dolan, E. D., Moré, J. J., 2002. Benchmarking optimization software with performance profiles. Mathematical Programming 91 (2), 201–213.

Errico, F., Crainic, T. G., Malucelli, F., Nonato, M., 2017. A Benders Decomposition Approach for the Symmetric TSP with Generalized Latency Arising in the Design of Semiflexible Transit Systems. Transportation Science 51 (2), 706–722.

Feng, C.-m., Wang, T.-c., 2003. Highway Emergency Rehabilitation Scheduling in Post-Earthquake 72 Hours. Journal of the Eastern Asia Society for Transportation Studies 5, 3276–3285.

Gendron, B., Scutellà, M. G., Garroppo, R. G., Nencioni, G., Tavanti, L., 2016. A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks. European Journal of Operational Research 255 (1), 151–162.

Hjorring, C., Holt, J., 1999. New optimality cuts for a single-vehicle stochastic routing problem. Annals of Operations Research 86, 569–584.

Klingman, D., Napier, A., Stutz, J., 1974. Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. Management Science 20 (5), 814–821.

Laporte, G., Louveaux, F. V., 1993. The integer L-shaped method for stochastic integer programs with complete recourse. Operations Research Letters 13 (3), 133–142.

Laporte, G., Louveaux, F. V. F., Hamme, L. V., van Hamme, L., 2014. An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. Operations Research 50 (3), 415–423.

Maya-Duque, P. A., Dolinskaya, I. S., Sörensen, K., 2016. Network repair crew scheduling and routing for emergency relief distribution problem. European Journal of Operational Research 248 (1), 272–285.

Miller, C. E., Tucker, A. W., Zemlin, R. A., 1960. Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM 7 (4), 326–329.

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

Pramudita, A., Taniguchi, E., 2014. Model of debris collection operation after disasters and its application in urban area. International Journal of Urban Sciences 18 (2), 218–243.

Pramudita, A., Taniguchi, E., Qureshi, A. G., 2012. Undirected Capacitated Arc Routing Problems in Debris Collection Operation After Disaster. Infrastructure Planning and Management 68 (5), 805–813.

Rahmaniani, R., Crainic, T. G., Gendreau, M., Rei, W., 2017. The Benders decomposition algorithm: A literature review. European Journal of Operational Research 259 (3), 801–817.

Tang, C.-H., Yan, S., Chang, C.-W., 2009. Short-term work team scheduling models for effective road repair and management. Transportation Planning and Technology 32 (3), 289–311.

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

Xu, B., Song, Y., 2015. An Ant Colony-based Heuristic Algorithm for Joint Scheduling of Post-earthquake Road Repair and Relief Distribution. TELKOMNIKA (Telecommunication Computing Electronics and Control) 13 (2), 632.

Yan, S., Chu, J. C., Shih, Y.-L., 2014. Optimal scheduling for highway emergency repairs under large-scale supply-demand perturbations. IEEE Transactions on Intelligent Transportation Systems 15 (6), 2378–2393.

Yan, S., Shih, Y.-L., 2007. A time-space network model for work team scheduling after a major disaster. Journal of the Chinese Institute of Engineers 30 (1), 63–75.

Yan, S., Shih, Y.-L., 2009. Optimal scheduling of emergency roadway repair and subsequent relief distribution. Computers and Operations Research 36 (6), 2049–2065.

Yan, S., Shih, Y.-L., 2012. An ant colony system-based hybrid algorithm for an emergency roadway repair time-space network flow problem. Transportmetrica 8 (5), 361–386.