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

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] *Production Engineering Department, Federal University of São Carlos, Rod. João Leme dos Santos km 110, CEP 18052-780, Sorocaba, Brazil*

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 road restoration crew scheduling and routing problem (RRCSRP) 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@ufscar.br (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. We are in particular interested in a variant that explicitly considers the complex interdependence between scheduling and routing decisions, hereafter called the road restoration crew scheduling and routing problem (RRCSRP). The scheduling decisions define the sequence in which the damaged points in the network will be visited by the crews. The routing decisions determine the paths/routes to be used by the crews to visit and repair the damaged points. In this variant, a path is usually a sequence of nodes and arcs used by the crews 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 points.

The design of crew routes is challenging because damaged points can obstruct access to other points of the network and because 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 points are damaged at that moment, which in turn depends on the scheduling decisions. In addition, without considering routing decisions simultaneously, the damaged points 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 points, if they exist, change dynamically during the restoration according to the schedule.

The RRCSRP has been tackled recently via the proposition of integrated mathematical programming models (Maya-Duque et al., 2016; Akbari and Salman, 2017b). However, such models have proven to be intractable by general-purpose mixed integer programming (MIP) solvers even for small problem instances. An exact method based on dynamic programming has been proposed by Maya-Duque et al. (2016), but it fails to solve most instances of the problem. Hence, most authors in this field have devised heuristic and metaheuristic methods to solve practical instances of the integrated problem without an optimality guarantee or any information on the quality of the solutions.

In this paper, we develop an exact algorithm based on Benders decomposition for the RRCSRP. 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 RRCSRP.

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 RRCSRP. 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 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 provide a literature review about the RRCSRP. Section 3 describes the problem and presents a mathematical model. 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. Literature review

This literature review is mainly focused on variants of the road restoration problem with integrated scheduling and routing decisions. Readers are referred to Çelik (2016) for a comprehensive review of road restoration and recovery in humanitarian operations.

Feng and Wang (2003) were the first authors to propose a mathematical model integrating scheduling and routing decisions for road restoration, focusing on highway emergency rehabilitation after earthquakes. They developed a multi-objective model to maximize the total kilometers of roads repaired, to maximize the total number of lives saved, and to minimize the risk of the restoration operations in the first 72 hours post-disaster. Yan and Shih (2007) proposed an MIP formulation that minimizes the completion time of the restoration. To find feasible solutions of the model, they proposed a heuristic algorithm that divides the originally damaged network in several smaller networks. For each subnetwork, a smaller subproblem is solved using general-purpose optimization software. However, even the subproblems remain unsolvable due to the time limitations imposed in real-life instances.

In another study, Yan and Shih (2012) developed an ant colony system-based metaheuristic to solve practical instances of the problem. Yan et al. (2014) incorporated rescheduling repair decisions into the model proposed by Yan and Shih (2007) to include perturbations in the number of repair crews available and in the number of damaged roads. The authors also used an ant colony system-based metaheuristic to solve the problem. Tang et al. (2009) incorporated both stochastic travel and repair times into the road restoration problem using a two-stage stochastic programming model. In the first-stage, the scheduling and routing decisions are considered. In the second-stage, alternative routing decisions are considered in each scenario. The model aims at minimizing the travel and repair times plus an expected penalty value. Small instances of the problem were solved by a commercial optimization solver.

Özdamar et al. (2014) proposed a multi-objective non-linear recursive model to minimize the network inaccessibility and to minimize the completion time. In this 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 crews 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.

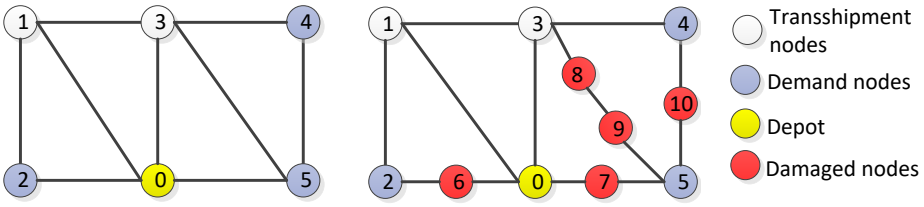
Some authors have considered additional decisions integrated with the crew scheduling and routing decisions. Yan and Shih (2009) integrated crew scheduling and routing with relief distribution in a bi-objective model to minimize the completion time of the restoration and the time to complete the distribution to all demand nodes. The bi-objective model was reduced to a single objective via evaluation of a weighted objective function and thus was solved by a heuristic analogous to Yan and Shih (2007). Pramudita et al. (2012) and Pramudita and Taniguchi (2014) integrated location decisions in the problem. 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. The objective is to minimize the cost of collecting the debris in all the damaged arcs. Pramudita and Taniguchi (2014) studied the same problem by transforming the CARP into the capacitated vehicle routing problem (CVRP). For the transformation, blocked arcs were associated with two nodes that must be visited in sequence. Pramudita et al. (2012) and Pramudita and Taniguchi (2014) used the tabu search metaheuristic to solve 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.

Finally, Maya-Duque et al. (2016) integrated additional decisions for defining the paths between the depot and the affected areas to optimize the accessibility of the demand nodes. They proposed a model in which the objective function is the sum of the time instants at which each demand node becomes accessible from the depot. Additionally, they developed two different approaches to solve the problem: a dynamic programming algorithm for small instances and a metaheuristic based on GRASP for medium and large instances.

Notice that most studies use heuristic/metaheuristic algorithms to address the problem. Some authors rely on mathematical formulations and general-purpose optimization software to solve small instances. A dynamic programming algorithm proposed by Maya-Duque et al. (2016) is the only specialized exact method proposed for the RRCSRP. All these exact approaches fail to solve even small instances of the problem, i.e., networks with up to 30 nodes. To the best of our knowledge, no paper has utilized decomposition techniques such as Benders decomposition to develop exact methods for the RRCSRP thus far.

3. Problem description and mathematical modeling

The RRCSRP 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 undirected 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 arcs, considered demand nodes with zero demand. 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 arcs (roads) were damaged. Notice that some arcs 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 arcs 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 arc $e \in \mathcal{E}$.



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

Figure 1: Example of a graph representing the RRCSRP.

In this paper, the objective function of the RRCSRP consists of minimizing the time that the demand nodes remain inaccessible from the depot weighted by their corresponding demands, as in Maya-Duque et al. (2016). 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. 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.

When damaged nodes are repaired, demand nodes can become accessible. 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 can be 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 arcs 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. 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, 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.

To formulate the RRCSRP, we closely follow the mathematical model proposed by Maya-Duque et al. (2014, 2016). The notation used to describe the model is as follows.

Sets

$\mathcal{V} = \mathcal{V}^r \cup \mathcal{V}^d \cup \{0\}$	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} { 1, if node $j \in \mathcal{V}^r \cup \{0\}$ is repaired immediately after node $i \in \mathcal{V}^r \cup \{0\}$.
0, otherwise.

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

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

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

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

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.

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) minimizes the weighted sum of the exact time at which each demand node becomes accessible. 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 proposed by Maya-Duque et al. (2014), 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. Solution approach

In this section, we propose a branch-and-cut algorithm based on Benders decomposition for the RRCSRP, referred to as a branch-and-Benders-cut (BBC) algorithm. Basically, this BBC has three main components: an MIP master problem defined in Subsection 4.1, optimality and feasibility cuts defined in Subsection 4.2 and separation routines defined in Subsection 4.3. 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 relaxed 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 4.4. Additionally, in Subsection 4.5, we derive valid inequalities to have stronger LP relaxations, and in Subsection 4.6, we develop construction and local search heuristics to find good feasible solutions.

4.1. Benders decomposition

Benders decomposition (Benders, 1962) is a variable partitioning technique whose goal is to tackle problems with complicating variables. Usually, a relaxed MP considering only the

complicating variables is solved, then the complicating variables are temporarily fixed, and one or more subproblems are solved. For the RRCSRP, 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 relaxed master problem (RMP) is defined as follows:

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

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

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

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

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

We use the term *relaxed* because the model (19)-(23) still lacks the feasibility and optimality cuts to be defined in Subsection 4.2. Notice that constraints (11), which act also as subtour elimination constraints in model (1)-(18), do not remain in the RMP (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 RMP, 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 RMP, 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. Constraints (22) guarantee 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 RMP 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:

$$(SP1) \ \min \quad \sum_{i \in V^r} Z_i^r, \tag{24}$$

$$\text{s.t.} \quad \text{Constraints (6), (12), (15), (17)}, \tag{25}$$

$$\sum_{e \in \mathcal{E}_i} P_{eij} = \widehat{X}_{ij}, \forall\ i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{26}$$

$$\sum_{e \in \mathcal{E}_j} P_{eij} = \widehat{X}_{ij}, \forall\ i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{27}$$

$$Z_j^r \geq Z_i^r + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + (\widehat{X}_{ij} - 1) \cdot M + \delta_j, \forall\ i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r, \tag{28}$$

in which $\widehat{X}_{ij}$ is a solution for the RMP. For each pair of consecutive nodes $i - j$ with $\widehat{X}_{ij} = 1$ in the schedule defined by the RMP, 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 RMP is infeasible in the original problem (1)-(18), and feasibility cuts must be added to the RMP (see Subsection 4.2). 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:

$$(SP2) \qquad \min \quad \sum_{i \in \mathcal{V}^d} d_i \cdot Z_i^d, \tag{29}$$

$$\text{s.t.} \quad \text{Constraints (7), (8), (9), (10), (16), (18),} \tag{30}$$

$$Z_i^d \geq \widehat{Z}_k^r + (V_{ki} - 1) \cdot M, \, \forall \, i \in \mathcal{V}^d, \; k \in \mathcal{V}^r, \tag{31}$$

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 RMP, 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 RRCSRP into three parts: the RMP, 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 RMP solutions simultaneously. However, having separate subproblems allows us to design efficient specialized algorithms, as presented in Subsection 4.3.

4.2. 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 RMP. 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 RMP.

Proposition 1. *Let* $K = (v_0, v_1, ..., v_{(h-1)}, v_h, ..., v_p, ..., v_{|\mathcal{V}^r|})$ *be a 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 RMP 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 RMP:*

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

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

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 RMP 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 RMP 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 RMP.

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 RMP 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 RMP 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 RMP, 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 RMP, as we have $\Theta \geq \widehat{\Theta}(p - (h-1))$ with $p < h$. □

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 RMP 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 RMP:*

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

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

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 RMP than cuts (34) and (36).

4.3. 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 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

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 δ_j, $\forall j \in \mathcal{V}^r$, and τ_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: $\quad C_e := \tau_e + \delta_{v_j}$, $\forall e \in \mathcal{E}_{v_j}$;
6: $\quad$ Find the cost $\mathcal{C}$ of the shortest path from node v_{j-1} to v_j;
7: $\quad$ **if** $\mathcal{C} < \infty$ **then**
8: $\quad\quad \widehat{Z}_{v_j}^r := \widehat{Z}_{v_{j-1}}^r + \mathcal{C}$;
9: $\quad\quad C_e := \tau_e$, $\forall e \in \mathcal{E}_{v_j} : e \notin \bigcup_{i=j+1}^{|\mathcal{V}^r|} \mathcal{E}_{v_i}$;
10: $\quad\quad C_e := \infty$, $\forall e \in \mathcal{E}_{v_j} : e \in \bigcup_{i=j+1}^{|\mathcal{V}^r|} \mathcal{E}_{v_i}$;
11: $\quad$ **else**
12: $\quad\quad$ return "Infeasible SP1";
13: $\quad$ **end if**
14: **end for**
15: return "Feasible SP1";

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 $C_e = \tau_e = 1$.

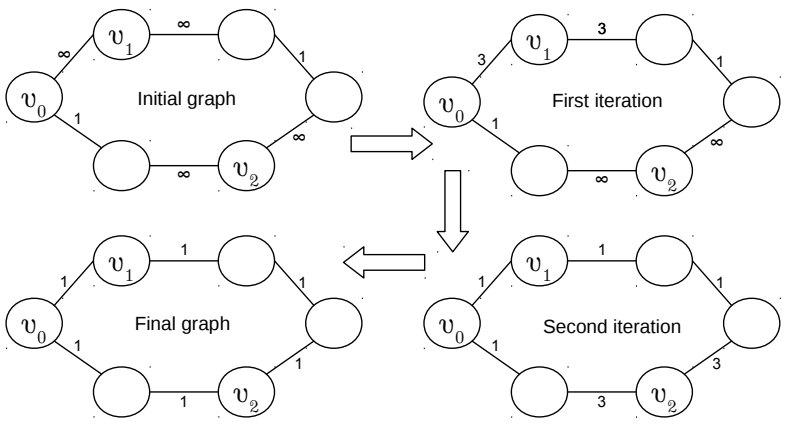


Figure 2: Example of the variation in arc costs in Algorithm 1.

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 $\widehat{Z}_i^d = 0$. Otherwise, let $j \in \mathcal{V}^r$ be the last damaged node that was repaired before i becomes accessible at exact time $\widehat{Z}_j^r$. Then, $\widehat{Z}_i^d = \widehat{Z}_j^r$. 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_i^r 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 $\widehat{Z}_i^d$ in which the demand node i becomes accessible is set as $\widehat{Z}_{v_j}^r$ (line 8). Note that we update $\widehat{Z}_i^d$ only if it was not updated in previous iterations; thus, $\widehat{Z}_i^d$ 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 ℓ_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: $\quad C_e := \infty$, $\forall e \in \mathcal{E}_{v_j}$;
5: $\quad$ **for** $i = 1$ **to** $|\mathcal{V}^d|$ **do**
6: $\quad\quad$ Find the cost $\mathcal{C}_i$ of the shortest path from the depot to the demand node i;
7: $\quad\quad$ **if** $\mathcal{C}_i > l_i$ and $\widehat{Z}_i^d = 0$ **then**
8: $\quad\quad\quad \widehat{Z}_i^d := \widehat{Z}_{v_j}^r$;
9: $\quad\quad$ **end if**
10: $\quad$ **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 $\widehat{Z}_{v_2}^r$.

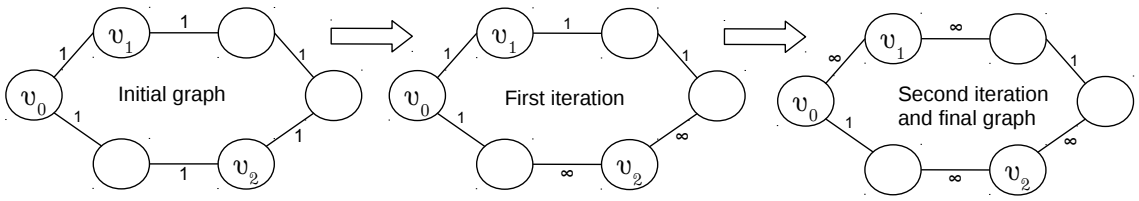


Figure 3: Example of the variation in arc costs in Algorithm 2.

4.4. Branch-and-Benders-cut

In the classical Benders decomposition, the RMP and the subproblems are solved iteratively in an alternating sequence. At each iteration, the RMP 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 RMP, 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 RMP. If no feasibility cut is obtained, then we solve SP2 to obtain an optimality cut for the RMP. 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 RMP 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 general-purpose optimization software can be used at each node of the branch-and-bound tree as well, although they are not included in Figure 4.

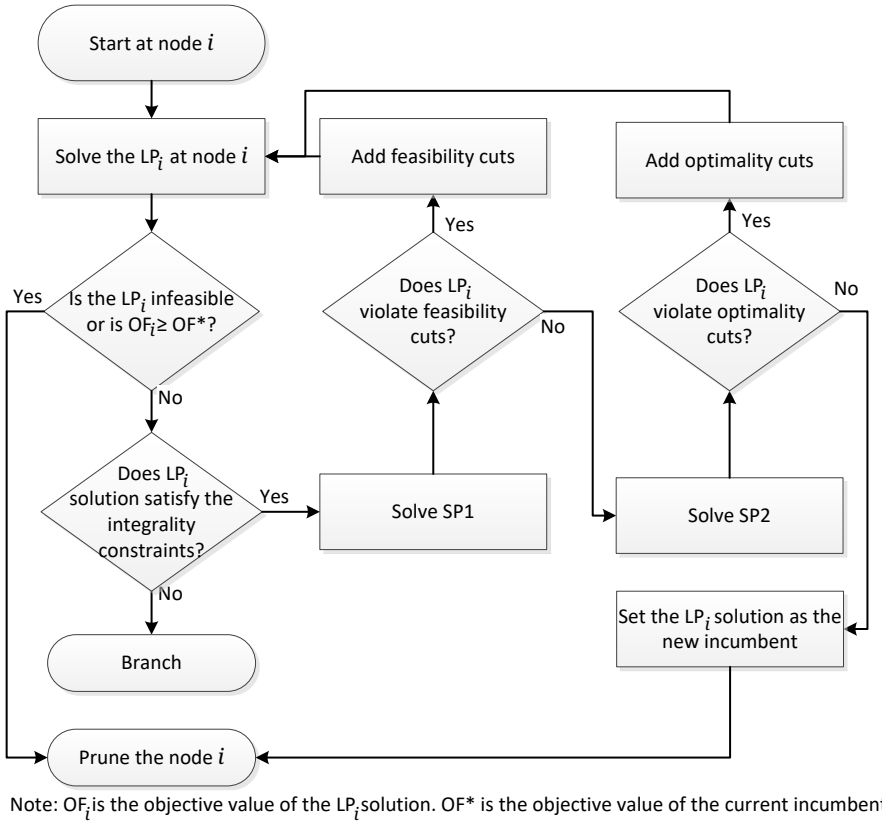


Figure 4: Flowchart illustrating how the separation routines are used in a given node i of the BBC method.

4.5. Valid inequalities

The proposed BBC method also relies on valid inequalities, which are added to the RMP and help improve the lower bounds provided by its linear relaxation. The first set of inequalities is based on the information of the network considering that no damaged node exists. To define them, we determine the shortest paths between the depot and each demand node $i \in \mathcal{V}^d$ with a distance less than l_i and identify the nodes that must be used in these paths. Then, the valid inequalities are given by:

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

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 RMP 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:

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

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.

4.6. 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 RRCSRP. The feasible solutions are used as initial incumbent solutions in the BBC algorithm.

4.6.1. 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.

4.6.2. 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 3. 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 8) and cost (line 10) 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.

Algorithm 3 Local search heuristic using the operator $n \in \{swap, 2\text{-}opt\}$.

```
Input:
Schedule K = (v_0, v_1, ..., v_j, ..., v_{|V^r|});
Cost Θ^K of scheduling K;
Output:
Schedule K* = (v*_0, v*_1, ..., v*_j, ..., v*_{|V^r|});
improvement := 1;
Determine set W^n_K;
K* := K;
while improvement = 1 do
    i := 1;
    improvement := 0;
    while i ≤ |W^n_K| do
        Evaluate feasibility of schedule K̂_i by solving subproblem SP1;
        if schedule K̂_i is feasible then
            Calculate cost Θ^{K̂_i} of schedule K̂_i by solving subproblem SP2;
            if Θ^{K̂_i} < Θ^K then
                Θ^K := Θ^{K̂_i};
                i := |W^n_K| + 1;
                K := K̂_i;
                K* := K;
                Determine new set W^n_K;
                improvement := 1;
            else
                i := i + 1;
            end if
        else
            i := i + 1;
        end if
    end while
end while
```

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 4.3 and 4.6. 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 groups 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). Table 1 shows the characteristics of the set of instances. 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). The type, name, and original number of nodes and arcs in the networks can be seen in columns 1, 2, 3 and 4 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. For each damaged arc, one or more damaged nodes were considered. 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 $\beta = 50\%$ 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. 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. The total numbers of nodes and arcs in the instance depend on the parameter α. Original network 1 with 25 nodes and 40 arcs, for example, is transformed into a damaged network with 27 nodes and 42 arcs when $\alpha = 5\%$ and into a damaged network with 45 nodes and 60 arcs when $\alpha = 50\%$.

Table 1: Set of instances.

Type	Original network	Number of original nodes	Number of original arcs	Values for α (%)		Values for β (%)		Total instances	
S1	1	25	40	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	2	25	37	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	3	25	39	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	4	30	83	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	5	30	89	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	6	30	84	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	7	35	118	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	8	35	115	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	9	35	113	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	10	20	39	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	11	20	37	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	12	20	37	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	13	40	146	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	14	40	143	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S1	15	40	143	5, 10, 25, 30, 50		5, 10, 25, 50		20	
S2	16	60	191	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	17	60	197	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	18	60	196	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	19	80	247	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	20	80	245	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	21	80	248	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	22	100	274	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	23	100	271	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	24	100	273	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	25	140	324	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	26	140	323	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	27	140	322	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	28	170	398	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	29	170	399	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	30	170	396	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	31	200	447	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	32	200	449	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	33	200	449	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	34	300	524	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	35	300	525	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	36	300	525	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	37	400	625	5, 25, 50	10, 30	25, 50	05, 10	6	4
S2	38	400	625	5, 25, 50	10, 30	05, 10	25, 50	6	4
S2	39	400	625	5, 25, 50	10, 30	25, 50	05, 10	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 ten 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 4.6. 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 4.2 and the valid inequalities presented in Section 4.5. The same separation algorithms are used in all the BBC strategies to identify the feasibility and cost of a scheduling solution of the RMP. Then, we enumerate and add inequalities to the RMP 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. Finally, the MIP model presented in Section 3 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, feasibility cut (33) and optimality multi-cuts (35) and (36) ($ET > TL$ or gap $= 0$).
BBC2	BBC algorithm with valid inequalities, feasibility cut (32) and optimality cut (34) ($ET > TL$ or gap $= 0$).
BBC3	BBC algorithm with valid inequalities, feasibility cut (32) and optimality multi-cuts (36) ($ET > TL$ or gap $= 0$).
BBC4	BBC algorithm with valid inequalities, feasibility cut (32) and optimality multi-cuts (35) and (36) ($ET > TL$ or gap $= 0$).
BBC5	BBC algorithm without valid inequalities, 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 $= 0$).
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:

$$P(f,q) = \frac{|\{p \in \mathcal{P} : \log_2(v(p,f)) \leq q\}|}{|\mathcal{P}|} \ , \ q \geq 0, \tag{43}$$

$$\text{with } v(p,f) = \frac{TC_{pf}}{min\{TC_{pf} : f \in F\}} \, , \tag{44}$$

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:

$$gap = \frac{Z^U - Z^L}{Z^U}, \tag{45}$$

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 5 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 94.47% (88.15%) of the instances, and in the remaining instances, H3 (H4) provides a solution with cost within a factor of $2^{0.24} \approx 1$ ($2^{0.36} \approx 1$) of the lowest cost found. Due to this behavior, H3 was selected as the best heuristic strategy.

Figure 6 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 most of the strategies presented similar results for a little more than 90% ($P(f,q) = 0.9$) of the instances. As expected, strategy BBC5 that does not use any valid inequalities presents the worst performance. In most instances, the valid inequalities help improve the lower bound and thus help improve the convergence of the algorithms. 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

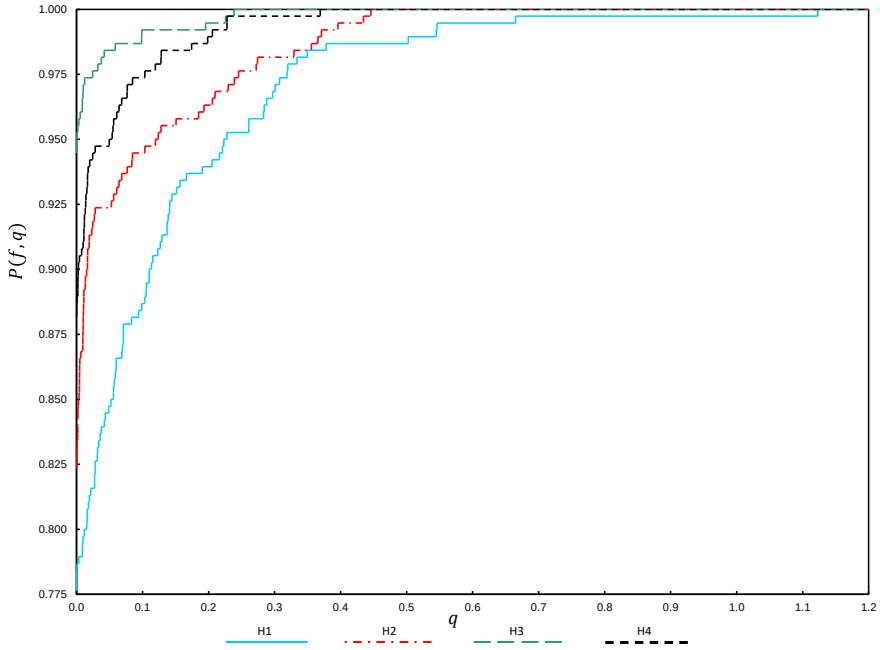


Figure 5: Performance profiles of the heuristic methods based on the objective values.

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.

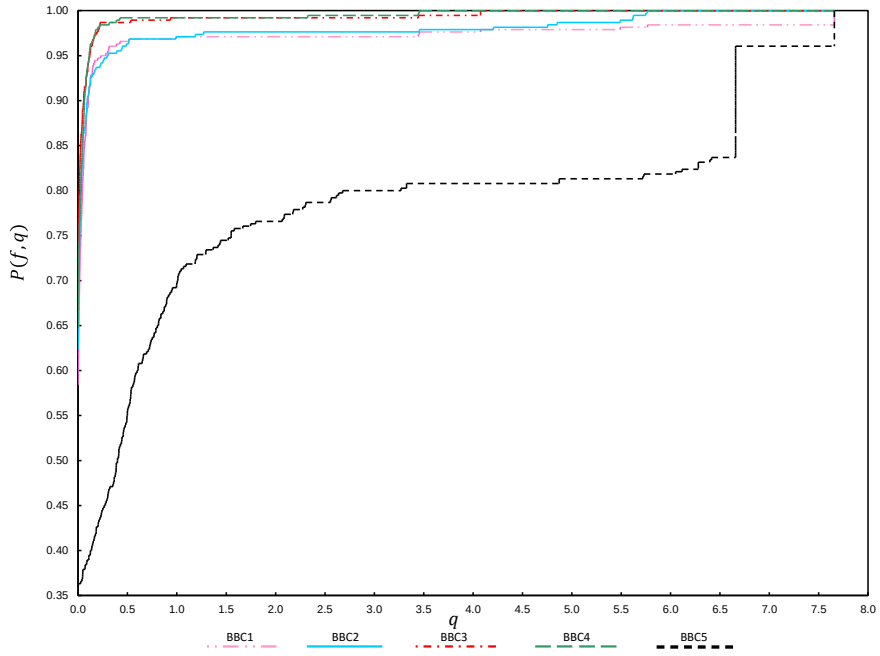


Figure 6: Performance profiles of the BBC algorithms based on the optimality gap.

Approach BBC6 combines the best heuristic and BBC strategies, H3 and BBC4, respectively. We build performance profiles based on the gap provided by the algorithms within the time limit of 3,600 seconds to compare BBC4, BBC6 and the MIP model. As we can see in Figure 7, not surprisingly, both BBC algorithms outperform the MIP model. In fact, the mathematical model

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 $P(f,q) = 1$.

found feasible solutions for only 65% 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.

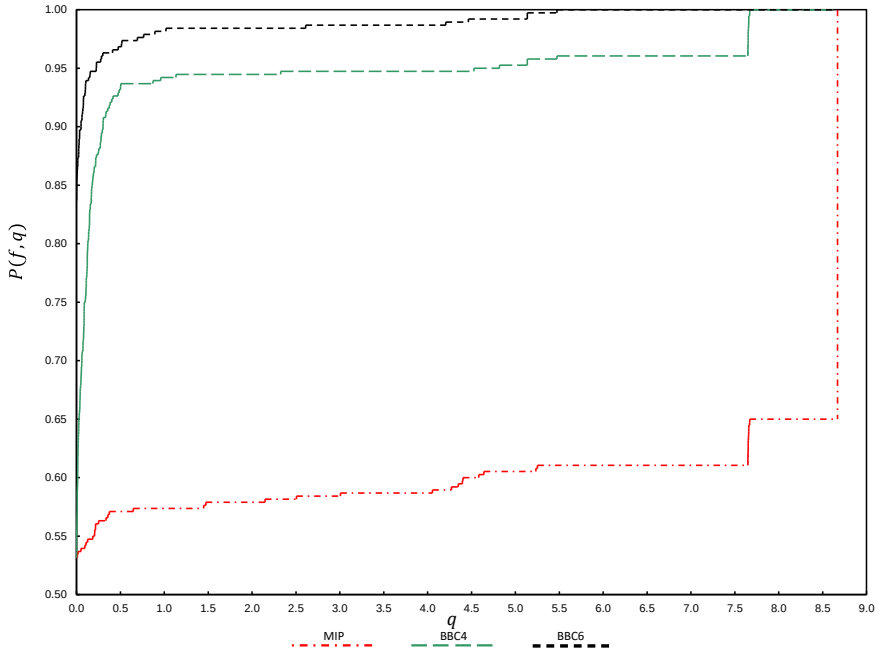


Figure 7: Performance profiles of the best BBC strategies and the MIP model based on the optimality gap.

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 BBC6 method provided feasible solutions within 3,600 seconds. The average gap considering all instances was 54.27%. The average total elapsed time was 2,617 seconds, and the average time that BBC6 spent to find the best upper bound was 1,122 seconds, 57% smaller than the elapsed time. Thus, BBC6 finds the solutions of the problem early in the optimization process, and most of the time consumed is to improve the lower-bound values. For instances S1, the average gap was 24.87%, while for instances S2, the average gap was 72.64%.

Table 5 shows the average gap of the BBC6 method according to different values of α and β. For example, the value 30.81 highlighted 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 in the network (α) increases, as expected. In fact, when there are more damaged nodes, there are possibly more crew schedules to be evaluated in the MP, slowing down the

convergence of the method. Additionally, there are more nodes in the network, making solving the subproblems harder. The BBC6 strategy found solutions with an average gap of 15.78% for the instances with $\alpha = 5\%$ and an average gap of 78.41% 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 49.56%, while the average gap for instances with $\beta = 50\%$ is 41.11%.

Table 4: Average results of the BBC6 strategy.

Type	Class (Network)	Avg. upper bound	Avg. gap (%)	Avg. elapsed time (sec.)	Avg. best[1] time (sec.)
S1	1	9,744.98	8.22	720.40	0.06
S1	2	34,088.95	17.85	1,284.93	162.46
S1	3	49,862.45	12.71	737.22	193.80
S1	4	18,132.98	20.64	1,800.02	1.36
S1	5	18,484.94	33.28	2,160.03	609.93
S1	6	20,921.92	25.97	1,651.64	337.32
S1	7	36,511.03	37.52	2,160.01	169.16
S1	8	26,048.79	27.22	2,160.04	6.25
S1	9	33,953.25	41.41	2,005.92	9.69
S1	10	48,459.97	10.26	744.63	46.80
S1	11	38,538.07	12.09	1,083.30	0.15
S1	12	28,036.81	8.43	720.58	163.22
S1	13	23,609.55	24.69	2,160.35	176.86
S1	14	81,113.72	45.49	2,303.79	33.87
S1	15	53,679.25	47.28	2,700.04	309.87
S2	16	39,496.70	31.61	2,159.99	550.00
S2	17	30,448.01	39.90	2,520.44	87.27
S2	18	112,769.97	46.69	2,520.19	286.04
S2	19	66,327.02	54.95	2,880.01	1,343.31
S2	20	74,659.97	52.52	2,521.87	2,065.80
S2	21	130,097.43	72.55	3,240.09	2,125.28
S2	22	233,237.97	59.52	3,240.28	1,933.92
S2	23	125,418.93	62.59	2,880.05	2,635.01
S2	24	225,489.76	76.67	3,600.00	2,104.00
S2	25	165,390.50	67.34	2,880.10	1,653.46
S2	26	275,426.20	70.18	3,520.16	2,776.84
S2	27	164,447.63	66.35	2,880.09	1,366.21
S2	28	435,199.83	76.82	3,600.00	1,830.31
S2	29	251,003.60	77.16	3,240.14	1,966.01
S2	30	478,406.37	77.90	3,600.00	1,300.81
S2	31	315,813.49	85.42	3,600.00	1,776.45
S2	32	337,827.97	83.83	3,600.00	1,694.58
S2	33	275,998.35	78.93	3,600.00	1,759.10
S2	34	481,424.07	93.88	3,600.00	1,761.70
S2	35	472,126.95	91.93	3,600.00	1,948.99
S2	36	497,333.03	92.73	3,600.00	1,999.92
S2	37	534,280.91	95.21	3,600.00	1,983.41
S2	38	697,782.38	95.09	3,600.00	2,374.04
S2	39	553,372.93	93.51	3,600.00	2,197.37
Avg. all		192,178.63	54.27	2,617.34	1,121.56
Avg. S1		34,745.78	24.87	1,626.19	148.05
Avg. S2		290,574.16	72.64	3,236.81	1,729.99

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

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

		α (%)					
		5	10	25	30	50	Avg.
β (%)	5	19.07	**30.81**	55.92	60.67	81.35	49.56
	10	16.34	30.52	55.20	61.18	80.15	48.68
	25	16.70	24.85	51.38	55.14	77.20	45.05
	50	11.00	20.66	46.09	52.81	74.97	41.11
	Avg.	15.78	26.71	52.15	57.45	78.41	46.10

5.5. Comparison with other methods in the literature

This section compares the BBC6 strategy with other methods proposed in literature, namely, the dynamic programming algorithm and the iterated greedy-randomized constructive procedure (IGRCP) metaheuristic, both of which were proposed by Maya-Duque et al. (2016). While the former approach is also an exact method analogous to our BBC6 approach, the latter approach is a heuristic and hence has no guarantee of optimality. Nevertheless, we include the results of this method in the analysis in order to verify how good the exact approaches are at providing upper bounds when they cannot reach optimality within the imposed time limit. The IGRCP metaheuristic is based on the greedy randomized adaptive search procedure (GRASP) metaheuristic 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. We show the results only for instances in group S1 (small instances) because the dynamic programming strategy proposed in Maya-Duque et al. (2016) is not effective in solving the proposed instances; thus, the authors did not use it to solve medium and large instances. The IGRCP metaheuristic, on the other hand, was used to solve instances of type S2 in Maya-Duque et al. (2016), but we did not have access to the solutions provided by this metaheuristic method.

Table 6 compares the average upper bound and elapsed time of BBC6 with the dynamic programming algorithm and the IGRCP metaheuristic for instances in group S1. The character "–" indicates that no solution was obtained for one or more instances of the class. The column ratio shows the ratio of the upper bound of IGRCP to the upper bound of BBC6. Ratios smaller than 1 indicate that the BBC6 strategy improves the upper bound found by the IGRCP metaheuristic. The dynamic programming algorithm solved all the instances to optimality for classes of instances corresponding to networks 1, 2 and 10. For the other classes, the dynamic programming 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 BBC6 method were equal to the solutions of the dynamic programming strategy, proving that these solutions are optimal, although the average gap related to the lower bound computed with the BBC6 method is higher than 0%. In terms of computational times, the dynamic programming strategy was slower than the BBC6 strategy for instances in classes 1, 2 and 10.

On average, the solutions provided by the BBC approach 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. As expected, the IGRCP metaheuristic was the fastest method but gave no guarantee of optimality. Note that most of the time spent by the BBC strategy is to improve the lower bound. In fact, the average time spent by BBC6 to find the best upper bound is 148.05 seconds, 11 times smaller than the average elapsed time.

6. Conclusions and future research

This paper explored branch-and-Benders-cut (BBC) approaches to solve the road restoration crew scheduling and routing problem. The results of the computational experiments suggested

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

	Avg. upper bound			Avg. elapsed time (sec.)			Avg. best[4]	
Class	IGRCP algorithm[1]	Dynamic programming[2]	BBC6[3]	IGRCP algorithm	Dynamic programming	BBC6	time (sec.) BBC6	Ratio
1	9,744.98	9,744.98	9,744.98	1.09	2,945.31	720.40	0.06	1.000
2	34,088.95	34,088.95	34,088.95	1.43	8,733.50	1,284.93	162.46	1.000
3	49,987.07	–	49,862.45	1.60	–	737.22	193.80	0.998
4	18,246.59	–	18,132.98	7.04	–	1,800.02	1.36	0.994
5	18,151.81	–	18,484.94	14.10	–	2,160.03	609.93	1.018
6	21,253.39	–	20,921.92	17.98	–	1,651.64	337.32	0.984
7	36,873.30	–	36,511.03	11.77	–	2,160.01	169.16	0.990
8	26,382.27	–	26,048.79	38.37	–	2,160.04	6.25	0.987
9	35,223.52	–	33,953.25	20.26	–	2,005.92	9.69	0.964
10	48,545.84	48,459.97	48,459.97	0.91	16,663.44	744.63	46.80	0.998
11	39,212.63	–	38,538.07	1.59	–	1,083.30	0.15	0.983
12	28,876.04	–	28,036.81	0.87	–	720.58	163.22	0.971
13	23,535.63	–	23,609.55	7.83	–	2,160.35	176.86	1.003
14	87,163.33	–	81,113.72	91.47	–	2,303.79	33.87	0.931
15	52,085.29	–	53,679.25	53.51	–	2,700.04	309.87	1.031
Average	35,291.38	–	34,745.78	17.99	–	1,626.19	148.05	0.990

[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 BBC6 spent to find the best upper bound.

the effectiveness of the proposed decomposition scheme and solution methods. In addition, the use of feasibility cuts, multiple optimality cuts and specialized valid inequalities has been proven to enhance the performance of the BBC approach, as they help transfer useful information from the subproblems to the master problem (MP). The use of simple heuristics to provide initial incumbent solutions for the MP was also an important strategy to accelerate the convergence of the method. The proposed BBC strategies improve the results of other exact and metaheuristic methods proposed in literature.

The next steps in this research include the development of alternative mixed integer programming formulations to consider multiple crews, prioritization, and humanitarian costs (Holguín-Veras et al., 2013). Another promising area of investigation is based on considering the uncertainties of disaster situations (Moreno et al., 2016; Alem et al., 2016). Finally, we intend to explore other efficient exact solution methods, e.g., branch-and-price based on interior point methods (Munari and Gondzio, 2013, 2015) in an attempt to handle larger instances and hybrid methods combining exact and metaheuristic strategies (Alvarez and Munari, 2017).

7. Acknowledgments

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 number 141973/2016-1].

References

Akbari, V., Salman, F. S., 2017a. Multi-vehicle prize collecting arc routing for connectivity problem. Computers & Operations Research 82, 52–68.
URL http://linkinghub.elsevier.com/retrieve/pii/S0305054817300072

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.

Alem, D., Clark, A., Moreno, A., 2016. Stochastic network models for logistics planning in disaster relief. European Journal of Operational Research 255 (1), 187–206.
URL http://www.sciencedirect.com/science/article/pii/S0377221716302788

Alvarez, A., Munari, P., 2017. An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. Computers & Operations Research 83, 1–12.
URL http://www.sciencedirect.com/science/article/pii/S0305054817300308

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.
URL http://dx.doi.org/10.1007/BF01386316

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.
URL http://linkinghub.elsevier.com/retrieve/pii/S1876735416300290

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.

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.
URL http://dx.doi.org/10.1007/s101070100263

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.
URL http://pubsonline.informs.org/doi/10.1287/trsc.2015.0636

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.
URL http://linkinghub.elsevier.com/retrieve/pii/S0377221716302958

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

Holguín-Veras, J., Pérez, N., Jaller, M., Van Wassenhove, L. N., Aros-Vera, F., 2013. On the appropriate objective function for post-disaster humanitarian logistics models. Journal of Operations Management 31 (5), 262–280.
URL http://dx.doi.org/10.1016/j.jom.2013.06.002

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.
URL http://dx.doi.org/10.1287/mnsc.20.5.814

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.
URL http://www.sciencedirect.com/science/article/pii/016763779390002X

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.
URL http://dx.doi.org/10.1287/opre.50.3.415.7751

Maya-Duque, P. A., Dolinskaya, I. S., Sörensen, K., 2014. Network repair crew scheduling and routing for emergency relief distribution problem, working paper 14-03, Northwestern University, Department of Industrial Engineering and Management Sciences.

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.
URL http://www.sciencedirect.com/science/article/pii/S0377221715005408

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.

Moreno, A., Alem, D., Ferreira, D., 2016. Heuristic approaches for the multiperiod location-transportation problem with reuse of vehicles in emergency logistics. Computers & Operations Research 69, 79–96.
URL http://dx.doi.org/10.1016/j.cor.2015.12.002

Munari, P., Gondzio, J., 2013. Using the primal-dual interior point algorithm within the branch-price-and-cut method. Computers & Operations Research 40 (8), 2026 – 2036.
URL http://www.sciencedirect.com/science/article/pii/S0305054813000762

Munari, P., Gondzio, J., 2015. Column generation and branch-and-price with interior point methods. Proceeding Series of the Brazilian Society of Applied and Computational Mathematics 3 (1), 7.
URL http://proceedings.sbmac.org.br/sbmac/article/view/617

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.
URL http://www.sciencedirect.com/science/article/pii/S0038012114000408

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.
URL http://www.tandfonline.com/doi/abs/10.1080/12265934.2014.929507

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 0, 1–17.
URL http://dx.doi.org/10.1016/j.ejor.2016.12.005

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.
URL http://www.tandfonline.com/doi/full/10.1080/03081060903017150

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.
URL http://dx.doi.org/10.1016/j.tre.2013.10.009http://linkinghub.elsevier.com/retrieve/pii/S1366554513001762

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.
URL http://journal.uad.ac.id/index.php/TELKOMNIKA/article/view/1437

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.
URL http://www.tandfonline.com/doi/abs/10.1080/02533839.2007.9671231

Yan, S., Shih, Y.-L., 2009. Optimal scheduling of emergency roadway repair and subsequent relief distribution. Computers and Operations Research 36 (6), 2049–2065.
URL http://linkinghub.elsevier.com/retrieve/pii/S0305054808001251

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.
URL http://www.tandfonline.com/doi/abs/10.1080/18128602.2010.515550