Decomposition-based algorithms 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 Luis Km 235, CEP 13565-905, São Carlos - SP, Brazil*
[b] *University of Edinburgh Business School, Management Science and Business Economics Group, 29 Buccleuch Place, EH89JS, Edinburgh, UK*

Abstract

The crew scheduling and routing problem (CSRP) consists of determining the best route and schedule for a single crew to repair damaged nodes in a network affected by extreme events. The problem also involves the design of paths to connect a depot to demand nodes that become accessible only after the damaged nodes in these paths are repaired. The objective is to minimize the total time that demand nodes remain inaccessible from the depot. The integrated scheduling and routing decisions make the problem too complicated to be effectively solved using mixed-integer programming (MIP) formulations. In this paper, we propose exact, heuristic and hybrid approaches for the CSRP. Specifically, we introduce (i) a branch-and-Benders-cut (BBC) method that enhances a previous approach by using a different variable partitioning scheme and new valid inequalities that strengthen the linear relaxation of the master problem; (ii) genetic algorithm and simulated annealing metaheuristics; and (iii) an exact hybrid BBC (HBBC) method that effectively combines the first two approaches. Computational experiments using benchmark instances show the benefits of the proposed enhancements for the BBC method and indicate that its performance is improved significantly by the use of metaheuristics within the hybridization scheme. The HBBC method obtained feasible solutions for the 390 tested instances, solving 30 of them to proven optimality for the first time. On average, it improved the best known lower and upper bounds by 15.21% and 8.41%, respectively, and reduced the computational times by more than 70% with respect to the standalone BBC.

Keywords: Benders decomposition, Branch-and-Benders-cut, Decomposition-based metaheuristic, Hybrid method, Crew scheduling and routing, Road restoration

1. Introduction

The *crew scheduling and routing problem* (CSRP) is an important network repair problem usually defined on an undirected graph $G = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of undirected arcs. The graph G represents a transportation infrastructure system composed of roads, bridges, and tunnels. Some of these components may be dramatically affected by extreme

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

events, such as natural hazards and terrorist attacks that interrupt distribution and/or evacuation operations to demand nodes. These operations are usually performed from or to a central depot. In this context, one of the decisions usually attributed to the CSRP is to determine the relief paths that must be used to connect the depot with the demand nodes, which are composed of arcs and nodes that can be damaged by the extreme events. The objective of this problem is to repair such damaged nodes as soon as possible in order to minimize the time that the demand nodes remain inaccessible from the depot. The damaged nodes must be repaired by a single crew initially located at the depot node. Thus, the CSRP also involves defining the schedule or sequence in which the damaged nodes must be repaired by the crew, in addition to the route that the crew must use to repair the damaged nodes and return to the depot in the end.

The CSRP has received increased attention in the last few years. Solution approaches for the problem range from mathematical programming models solved by commercial optimization packages to metaheuristics. Mathematical programming models for the basic variant of the CSRP have been proposed by Maya-Duque et al. (2016) and Moreno et al. (2019). Maya-Duque et al. (2016) proposed a nonlinear formulation for the problem that, according to the authors, could not be solved by optimization packages even for small-sized instances; for this reason, they did not present computational results for the proposed model. Later, Moreno et al. (2019) linearized the model presented by Maya-Duque et al. (2016) and obtained results for some small instances. Shin et al. (2019) proposed a different version for the CSRP by including additional relief distribution decisions. Mathematical formulations for other variants of the problem that do not integrate the definition of paths between the depot and the demand nodes have been proposed as well (Pramudita et al., 2012; Yan et al., 2014; Özdamar et al., 2014; Pramudita and Taniguchi, 2014; Kasaei and Salman, 2016; Akbari and Salman, 2017b,a). However, as some of these variants have proven to be NP-hard (Pramudita et al., 2012; Yan et al., 2014; Özdamar et al., 2014; Kasaei and Salman, 2016) and the models fail at solving even small instances, there has been an effort to devise heuristic and metaheuristic algorithms. In fact, the CSRP can be reduced to the traveling salesman problem (TSP), which is known to be an NP-hard problem, by considering that the paths from the depot to the demand nodes, and between the damaged nodes, are known a priori, i.e., considering only the scheduling decisions. The metaheuristics developed in the literature have focused on ant colony-based systems (Yan et al., 2014; Shin et al., 2019), tabu search (Pramudita et al., 2012; Pramudita and Taniguchi, 2014), and GRASP (Maya-Duque et al., 2016). Other specialized heuristics grounded on particular characteristics of the problem were developed in Özdamar et al. (2014); Akbari and Salman (2017a) and Akbari and Salman (2017b). Solution approaches for the restoration of disrupted links in other contexts such as power, water, and railway networks have also emerged recently (Jamshidi et al., 2018; Morshedlou et al., 2018; Zhang et al., 2019; Su et al., 2019).

Exact methods, such as dynamic programming (Maya-Duque et al., 2016) and branch-and-Benders-cut (BBC) algorithms (Moreno et al., 2019), were also designed for the CSRP. The dynamic programming method failed at solving even small instances of the problem since the number of possible states that need to be considered grows dramatically with the number of damaged nodes. The BBC approach takes advantage of the fact that when a schedule is fixed, the route of the crew and the paths to the demand nodes can be found using efficient algorithms. The BBC approach was effective to obtain feasible solutions and, in many cases, good lower bounds for the tested instances.

The BBC algorithm is a branch-and-cut algorithm in which the master problem obtained from a Benders decomposition is solved by a single search tree, and the cuts are generated within the tree using the subproblems as separation routines. This strategy has yielded satisfactory results in many applications and has been used together with other acceleration strategies to improve the performance of the method (Rahmaniani et al., 2017). For instance, Taşkin and Cevik (2013) and Gendron et al. (2014) proposed heuristics to find initial solutions to warm-start the BBC and valid inequalities to improve the lower bound of the master problem. Additionally, Taşkin and Cevik (2013) applied a local search algorithm to the master problem solutions during the execution of the BBC. Salazar-González and Santos-Hernández (2015) also proposed valid inequalities that are added in a BBC scheme together with Benders cuts and a heuristic approach to find good upper bounds. Furthermore, the authors transformed the subproblem into a classical max-flow problem instead of using a linear programming formulation in the separation procedure. Adulyasak et al. (2015) used lifting inequalities, multiple Pareto-optimal cuts generation, and an exponential-sized set of subtour elimination constraints (SECs) dynamically added together with the Benders cuts to accelerate the BBC. Arslan and Karaşan (2016) increased the efficiency of a BBC method by using multiple Pareto-optimal cuts generation and by using a construction heuristic to derive solutions for the subproblems. Gendron et al. (2016) devised a BBC method in which integer variables of the subproblem are relaxed and included in the master problem.

More recently, Shao et al. (2017) adopted a tabu list of solutions in the master problem to eliminate solutions that would invoke repetitive Benders subproblems. Fischetti et al. (2017) applied a stabilization procedure at the root node and heuristics along the nodes of the branch-and-cut tree. Errico et al. (2017) enhanced a BBC by adding SECs and other inequalities together with the Benders cuts. Additionally, the authors proposed heuristics to generate initial cuts and solutions. They also applied a local search operator during the BBC to improve the master problem solutions and to generate multiple cuts from the neighborhood of the current solution. Li et al. (2018) developed a BBC algorithm involving Pareto-optimal cuts and valid inequalities to restrict the feasible space of the master problem. Moreno et al. (2019) derived valid inequalities to strengthen the master problem formulation in the context of the CSRP. Additionally, they derived multiple cuts exploring particular characteristics of the problem and used construction heuristics to generate initial solutions. The authors also proposed specialized algorithms based on Dijkstra's algorithm to efficiently solve the subproblems.

Most of the related studies used valid inequalities to improve the lower bound of the master problem, heuristics to provide a solution to warm-start the algorithm, and strategies to efficiently solve the subproblems derived from the decomposition. Other effective strategies such as heuristics along the branch-and-cut tree to generate cuts in the neighborhood of the master problem solutions have been seldom considered in the corresponding literature. We call this strategy *hybrid branch-and-Benders-cut* (HBBC). Hybridizing Benders decomposition methods with heuristics or metaheuristics can simultaneously improve both the lower and upper bounds

(Rei et al., 2009; Raidl, 2015). The upper bound can be enhanced when new best solutions are found during the evaluation of the neighborhood of the current master problem solution, while the lower bound can be boosted via the generation of feasibility or optimality cuts from the solutions found by the heuristics. Heuristics can also reduce the size of the branch-and-cut tree by providing good solutions in early stages of the algorithm (Errico et al., 2017) and can be used as simple stabilization tools to handle the instability of the classical Benders decomposition (Rahmaniani et al., 2017).

Given this literature review, our contributions can be summarized as follows:

(i) We develop a BBC method that enhances the approach proposed by Moreno et al. (2019) using a different variable partitioning scheme that leads to a stronger relaxed master problem. This scheme keeps in the master problem the original objective function and the variables related to relief paths and scheduling decisions. In the BBC method of Moreno et al. (2019), only variables related to scheduling decisions were kept in the master problem, with no information on the original cost. With a stronger master problem, the enhanced BBC method requires fewer calls to the subproblem and hence generates fewer cuts.

(ii) We propose new valid inequalities to improve the performance of the BBC algorithm. These valid inequalities explore the special structure of the master problem and take advantage of the variables related to relief paths to further strengthen its linear relaxation.

(iii) We devise two new metaheuristics based on a genetic algorithm (GA) and simulated annealing (SA) to solve the CSRP. Our metaheuristics have a random search component that explores the space of scheduling decisions and an optimization component to find the best crew route and relief paths given a schedule. These are the first GA and SA approaches proposed for the CSRP. We show that both metaheuristics outperform the existing GRASP-based metaheuristic developed to solve the same problem.

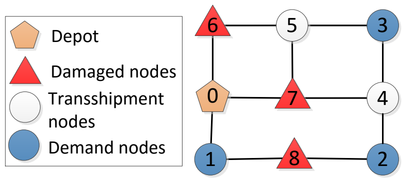
(iv) We hybridize our BBC approach by embedding metaheuristics in the branch-and-cut tree that solves the master problem. We use the metaheuristics not only to improve the master problem solutions but also to derive Benders cuts from the neighborhood of the solutions. To the best of our knowledge, this is the first time that a hybrid solution method has been proposed to solve the CSRP. Our hybrid approach significantly improves the results of the current best known approach.

The efficiency of the developed solution methods are compared using benchmark instances. As a result of the development of these techniques, the methods managed to obtain feasible solutions for the 390 benchmark instances, proving optimality for the first time on 30 of them. Moreover, they significantly improve the best known lower and upper bounds, optimality gaps, and execution times by 15.21%, 8.15%, 26.17%, and 71.14%, respectively, on average.

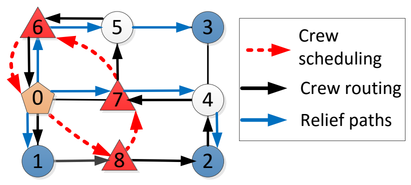
The remainder of this paper is organized as follows. The CSRP is described in Section 2. We apply Benders reformulation to the CSRP and describe the BBC algorithm in Section 3. We then describe the proposed SA and GA metaheuristics in Section 4. Section 5 presents the HBBC algorithm that combines the metaheuristics with the BBC algorithm. Finally, we report the computational results and analyses in Section 6 and draw our conclusions in Section 7.

2. Problem description and mathematical formulation

The *crew scheduling and routing problem* (CSRP) is defined on an undirected graph $G = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of undirected arcs. The subset $\mathcal{V}^r \subset \mathcal{V}$ contains the collection of damaged nodes, and $\mathcal{V}^d \subset \mathcal{V}$ characterizes those nodes for which there exists some type of demand, e.g., humanitarian assistance or relief supplies. The demand must be served from a source node (depot). There might also be transshipment nodes in set $\mathcal{V}$, thus representing the connection of arcs. A travel time τ_e and length ℓ_e are defined for each arc $e \in \mathcal{E}$. Furthermore, there exists a repair time δ_j spent by the crew to repair a damaged node $j \in \mathcal{V}^r$. Figure 1(a) shows an example of a graph G representing a damaged network.



(a) Graph G representing a damaged network.



(b) Main decisions in the CSRP.

Figure 1: Damaged network and main decisions in the CSRP.

One of the decisions attributed to the CSRP is to determine, for each demand node $i \in \mathcal{V}^d$, a path p_i connecting the depot with node i. We call such paths relief paths, and they are composed of arcs and nodes, some of which can be damaged by extreme events. Figure 1(b) shows an example of relief paths p_1, p_2, and p_3 (blue arrows). p_1 is composed of nodes 0→1, p_2 is composed of nodes 0→7→4→2, and p_3 is composed of nodes 0→6→5→3. More than one path p_i may be available for node i. For example, nodes 0→1→8→2 define an alternative path p_2. The objective of this problem is to repair the damaged nodes in the relief paths as soon as possible to minimize the time that the demand nodes remain inaccessible from the depot. A demand node i becomes accessible when the damaged nodes used in p_i are repaired. However, some demand nodes do not require the restoration of damaged nodes to become accessible. The accessibility time of these nodes is zero in these cases. In Figure 1(b), for example, nodes 2 and 3 become accessible after the restoration of damaged nodes 7 and 6, respectively, while the accessibility time of node 1 is zero. There is a total maximum distance l_i allowed for the path p_i. Thus, the sum of the lengths of the arcs used in path p_i must be less than or equal to l_i.

The damaged nodes are repaired by a single crew that departs from the depot. Thus, the CSRP also involves determining a schedule K that defines the sequence in which the damaged nodes must be repaired by the crew, in addition to the route that the crew must use to repair the nodes within this schedule. Figure 1(b) shows a schedule (red dashed lines) and a route (black arrows) followed by a crew. The schedule K is defined by the ordered set of nodes $(0, 8, 7, 6, 0)$. A route is composed of a sequence of crew paths p_{ij} that the crew must follow between two successive nodes $i - j$ in the schedule K. In Figure 1(b), for example, the crew routing is composed of the crew paths $p_{08} - p_{87} - p_{76} - p_{60}$. The path p_{87} is defined by nodes 8→2→4→7. Multiple paths p_{ij} may be available. The path defined by the nodes 8→2→4→3→5→7 is an alternative crew path p_{87}. Damaged nodes are repaired the first time they are visited, incurring the repair time. The crew can use the already repaired damaged nodes multiple times after their restoration without incurring extra repair time. Some damaged nodes may not be necessary for connecting the depot with the demand nodes. However, all damaged nodes need to be repaired. A mathematical formulation for the CSRP was proposed in Moreno et al. (2019) as a linearization of the nonlinear model introduced in Maya-Duque et al. (2016). This formulation uses different types of variables to represent the decisions illustrated in Figure 1(b). Before formally defining these variables, we briefly explain their meaning. Variables X_{ij}, for $i, j \in \mathcal{V}_0^r$, define the schedule of the crew. They do not provide the route of the crew, as they are only defined for damaged nodes. The full route of the crew is determined by variables P_{eij}, for $e \in \mathcal{E}$ and $i, j \in \mathcal{V}_0^r$, and N_{kij}, for $k \in \mathcal{V}$ and $i, j \in \mathcal{V}_0^r$. These variables determine the arcs and nodes to be visited by the crew, respectively. The exact time at which damaged node $i \in \mathcal{V}_0^r$ is repaired is represented by variable Z_i^r. Relief paths are determined by variables Y_{ej}, for $e \in \mathcal{E}$ and $j \in \mathcal{V}^d$, and V_{kj}, for $k \in \mathcal{V}$ and $j \in \mathcal{V}^d$, which define the arcs and nodes to be used in the relief paths, respectively. Finally, Z_i^d defines the accessibility time of demand node $i \in \mathcal{V}^d$. The notation used to describe the model is as follows.

Sets

$\mathcal{V}$	Set of nodes.
$\mathcal{V}^d \subset \mathcal{V}$	Set of demand nodes.
$\mathcal{V}^r \subset \mathcal{V}$	Set of damaged nodes.
$\mathcal{V}_0^r = \mathcal{V}^r \cup \{0\}$	Set of damaged nodes including the source node 0 (depot).
$\mathcal{E}$	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

$$X_{ij}\begin{cases} 1, \text{ if node } j \in \mathcal{V}_0^r \text{ is repaired immediately after node } i \in \mathcal{V}_0^r. \\ 0, \text{ otherwise.} \end{cases}$$

$$P_{eij}\begin{cases} 1, \text{ if arc } e \in \mathcal{E} \text{ is used on the path from node } i \in \mathcal{V}_0^r \text{ to node } j \in \mathcal{V}_0^r. \\ 0, \text{ otherwise.} \end{cases}$$

$$N_{kij}\begin{cases} 1, \text{ if node } k \in \mathcal{V} \text{ is used on the path from node } i \in \mathcal{V}_0^r \text{ to node } j \in \mathcal{V}_0^r. \\ 0, \text{ otherwise.} \end{cases}$$

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

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

Z^r_i Exact time at which damaged node $i \in \mathcal{V}^r_0$ is repaired.

Z^d_i Total time that demand node $i \in \mathcal{V}^d$ remains inaccessible from the depot.

Based on the defined notation, the MIP model for the CSRP is stated as follows.

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

$$\text{s.t.} \sum_{j \in \mathcal{V}^r_0} X_{ij} = 1, \; \forall\, i \in \mathcal{V}^r_0, \tag{2}$$

$$\sum_{i \in \mathcal{V}^r_0} X_{ij} = 1, \; \forall\, j \in \mathcal{V}^r_0, \tag{3}$$

$$\sum_{e \in \mathcal{E}_i} P_{eij} = X_{ij}, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r, \tag{4}$$

$$\sum_{e \in \mathcal{E}_j} P_{eij} = X_{ij}, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r, \tag{5}$$

$$\sum_{e \in \mathcal{E}_k} P_{eij} = 2N_{kij}, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r,\; k \in \mathcal{V} \setminus \{i,\; j\}, \tag{6}$$

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

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

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

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

$$Z^r_j \ge Z^r_i + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + \delta_j - (1 - X_{ij}) \cdot M, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r, \tag{11}$$

$$Z^r_j \ge Z^r_k + (N_{kij} - 1) \cdot M, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r,\; k \in \mathcal{V}^r, \tag{12}$$

$$Z^d_i \ge Z^r_j + (V_{ji} - 1) \cdot M, \; \forall\, i \in \mathcal{V}^d,\; j \in \mathcal{V}^r, \tag{13}$$

$$X_{ij} \in \{0,\, 1\}, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r_0, \tag{14}$$

$$Y_{ei}, V_{ki} \in \{0,\, 1\}, \; \forall\, i \in \mathcal{V}^d,\; k \in \mathcal{V},\; e \in \mathcal{E}, \tag{15}$$

$$Z^d_i \ge 0, \; \forall\, i \in \mathcal{V}^d, \tag{16}$$

$$P_{eij}, N_{kij} \in \{0,\, 1\}, \; \forall\, i \in \mathcal{V}^r_0,\; j \in \mathcal{V}^r_0,\; k \in \mathcal{V},\; e \in \mathcal{E}, \tag{17}$$

$$Z^r_i \ge 0, \; \forall\, i \in \mathcal{V}^r_0. \tag{18}$$

The objective function (1) minimizes the weighted sum of the time that each demand node remains inaccessible from the depot. A demand node i becomes accessible when all the damaged nodes in the path connecting node i with the depot 0 are repaired. Constraints (2) and (3) specify that each damaged node must be repaired once during the schedule of the crew. Constraints (4), (5) and (6) ensure the flow conservation in the path of the crew between two damaged nodes i and j. Similarly, constraints (7), (8) and (9) ensure the flow conservation in the relief paths from the depot to the demand nodes. Constraints (10) prohibit the use of relief paths with a distance greater than the maximum distance allowed between the depot and the demand nodes. Constraints (11) define the exact time at which the damaged nodes are repaired. These constraints also work as subtour elimination 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., unrepaired damaged 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 becomes accessible, which is based on the time when the damaged nodes in the path connecting node 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 binaries.

3. Branch-and-Benders-Cut algorithm (BBC)

In this section, we describe the novel BBC algorithm proposed for solving the CSRP based on a new Benders reformulation of the problem. Benders decomposition is a variable partitioning technique in which, usually, a relaxed master problem (RMP) considering only the complicating variables is solved, and then the complicating variables are temporarily fixed, and one or more subproblems are solved. Moreno et al. (2019) approached the CSRP using Benders decomposition, but considering only the scheduling decisions (variables X_{ij}) as complicating variables. They considered two subproblems, one to check the feasibility of the RMP solutions using only the variables related to crew routing (Z_i^r, P_{eij} and N_{kij}); and another to check the cost of the RMP solutions using only the variables related to relief paths (Z_i^d, Y_{ej} and V_{kj}). This decomposition leads to a weak RMP formulation since the objective function of the original problem, which minimizes the weighted sum of the variables Z_i^d, is considered in the second subproblem and no information of the original costs was kept in the RMP.

We propose a different variable partitioning scheme in which the variables defining the paths that connect the depot with the demand nodes (Z_i^d, Y_{ej} and V_{kj}) are kept in the RMP, in addition to the scheduling decision variables. As a consequence, the master problem becomes stronger, and the resulting BBC method is likely to require fewer calls to the subproblem, hence decreasing the number of generated cuts. In the proposed scheme, the RMP is defined as follows:

$$(RMP) \qquad \min \sum_{i \in \mathcal{V}^d} d_i \cdot Z_i^d, \tag{19}$$

$$\text{s.t.} (2), (3), (7) - (10), (14) - (16), \tag{20}$$

$$Z_i^d \geq \theta_j + (V_{ji} - 1) \cdot M, \; \forall \; i \in \mathcal{V}^d, \; j \in \mathcal{V}^r, \tag{21}$$

$$\theta_j \geq \theta_i + \delta_j - M \cdot (1 - X_{ij}), \; \forall \; i \in \mathcal{V}_0^r, \; j \in \mathcal{V}^r. \tag{22}$$

$$\theta_j \geq 0, \; \forall \; j \in \mathcal{V}^r. \tag{23}$$

Constraints (21) define the time at which each demand node i becomes accessible. Constraints (22) are introduced to set a lower bound for the time at which the damaged nodes are repaired. These constraints also act as subtour elimination constraints. Constraints (23) impose the domain of the decision variables θ_j. The value of variable θ_j is underestimated because the RMP does not consider the routing decisions of the crew. The full route of the crew consists of paths connecting the consecutive damaged nodes in the schedule defined by variables X_{ij}. Since damaged nodes can obstruct the access to other damaged nodes of the network, the paths available for the crew at a specific moment depend on which nodes are still damaged at that moment, which, in turn, depends on the scheduling decisions. Thus, the paths available for the crew change dynamically during the restoration according to the schedule. Without considering routing decisions in the RMP, the damaged points that are not accessible at a given moment might be selected first in the schedule, making the schedule infeasible in practice. Therefore, the RMP lacks feasibility cuts to avoid infeasible schedules and optimality cuts to set the real values of the variables θ_j.

To derive feasibility and optimality cuts for the RMP, we solve a subproblem that defines the route of the crew from a scheduling solution. The subproblem can be stated as follows:

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

$$\text{s.t.} \quad (6), (12), (17), (18), \tag{25}$$

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

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

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

in which parameter $\widehat{X}_{ij}$ 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, SP determines the shortest path with arcs and nodes defined by variables P_{eij} and N_{kij}, respectively. The actual value of the variable θ_j in the RMP is given by variable Z_j^r in the SP. Hence, if subproblem SP is feasible, we need to add optimality cuts to the RMP to update the cost of variable θ_j. Otherwise, if subproblem SP is infeasible, we need to add feasibility cuts to the RMP to cut off infeasible scheduling solutions. The feasibility and optimality cuts are defined by Propositions 1 and 2, respectively. In both propositions, $K = (v_0, v_1, ..., v_{(h-1)}, v_h, ..., v_p, ..., v_{|\mathcal{V}^r|})$ is a given schedule for the crew, where v_i is the ith damaged node to be repaired and $v_0 = 0$. This schedule is obtained by solving the RMP and corresponds to the solution $\widehat{X}_{v_{(i-1)} v_i} = 1,\ \forall i = 1, ..., |\mathcal{V}^r|$.

Proposition 1 (Moreno et al. (2019)). *For a given index $h > 0$, let $S_h = \{v_0, v_1, ..., v_{(h-1)}, v_h\}$ be the sequence of the first h damaged nodes in a schedule K of the crew, and assume that schedule 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 cut is violated and has to be added to the RMP:*

$$\sum_{i \in S_h \setminus \{v_h\}} \ \sum_{\substack{j \in S_h \setminus \{v_0\}: \\ j \neq i}} X_{ij} \leq |S_h| - 2. \tag{29}$$

Proof. The proof of Proposition 1 is given in Moreno et al. (2019). □

Proposition 2. *Assume that K is a feasible schedule, with values $\widehat{Z}_{v_j}^r$ computed in subproblem*

SP, for each $j \in \mathcal{V}^r$. Then, the following optimality multicuts are valid inequalities of the RMP:

$$\theta_{v_j} \geq \widehat{Z}^r_{v_j} \cdot (\sum_{i=1}^{j} X_{v_{(i-1)}v_i} - (j-1)),\ \forall\ j \in \mathcal{V}^r, \tag{30}$$

$$\theta_{v_l} \geq (\widehat{Z}^r_{v_j} + t_{v_j v_l} + \delta_{v_l}) \cdot (\sum_{i=1}^{j} X_{v_{(i-1)}v_i} - (j-1)),\ \forall\ j \in \mathcal{V}^r,\ \ l \in \mathcal{V}^r, l > j, \tag{31}$$

in which t_{ij} is the minimum travel time from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$ and δ_j is the repair time of damaged node j.

Proof. In cuts (30), the term $\sum_{i=1}^{j} X_{v_{(i-1)}v_i} - (j-1)$ is equal to 1 if the partial schedule $K' = (v_0, v_1, ..., v_j)$ for a given j is part of the solution of the RMP. Therefore, the cuts become $\theta_{v_j} \geq \widehat{Z}^r_{v_j}$, and thus the lower bound $\widehat{Z}^r_{v_j}$ is activated for variable θ_{v_j}. If the partial schedule K' is not considered in the solution of the RMP, the term $\sum_{i=1}^{j} X_{v_{(i-1)}v_i} - (j-1)$ is smaller than 1 and the cuts become deactivated. Similarly, in cuts (31), if the partial schedule K' is considered in the RMP solution, the lower bound $\widehat{Z}^r_{v_j} + t_{v_j v_l} + \delta_{v_l}$ is activated for all nodes v_l with $l > j$. The lower bound $\widehat{Z}^r_{v_j} + t_{v_j v_l} + \delta_{v_l}$ is valid because it is known that nodes v_l need to be repaired at some moment after node v_j and the crew must spend at least $t_{v_j v_l}$ time units to reach node v_l from node v_j and δ_{v_l} time units to repair it. $\square$

Note that cuts (30) are sufficient for the variables θ_{v_j} assuming their actual values $\widehat{Z}^r_{v_j}$ in the RMP. However, the additional cuts (31) can be added to speed up the performance of the method. To illustrate Proposition 2, consider the schedule $K = (v_0, v_1, v_2) = (0, 2, 1)$ that is assumed to be feasible. Cuts (30) lead to $\theta_2 \geq \widehat{Z}^r_2 \cdot (X_{02})$ and $\theta_1 \geq \widehat{Z}^r_1 \cdot (X_{02} + X_{21})$. Additionally, cuts (31) result in $\theta_1 \geq (\widehat{Z}^r_2 + t_{21} + \delta_1) \cdot (X_{02})$.

Together with the Benders cuts (29), (30) and (31), we propose the use of subtour elimination constraints (SECs) based on the Dantzig–Fulkerson–Johnson (DFJ) formulation (Dantzig et al., 1954). SECs based on DFJ can lead to stronger linear relaxations than those based on constraints (22) of the RMP. They are stated as follows:

$$\sum_{i \in S} \sum_{j \in \bar{S}} X_{ij} + \sum_{i \in \bar{S}} \sum_{j \in S} X_{ij} \geq 2, \forall\ S \subset \mathcal{V}^r : 2 \leq |S| \leq |\mathcal{V}^r| - 1, \tag{32}$$

in which $\bar{S} = \mathcal{V}^r_0 \setminus S$. Because the formulation based on DFJ contains an exponentially large number of SECs, a typical strategy is to gradually add the constraints to the formulation through a branch-and-cut scheme (Adulyasak et al., 2015). In practice, relatively few SECs are needed (Öncan et al., 2009). In our BBC, we add them after solving the LP problem at each node of the branch-and-cut tree. To detect the violated subtour constraints in a fractional solution $\widehat{X}$, we solve a series of minimum $s-t$ cut problems in a support graph $G^* = (N^*, E^*)$, in which $N^* = \mathcal{V}^r_0$ and $E^* = \{(i,j) | \widehat{X}_{ij} > 0\}$. We set the depot as the source node s and the damaged node $i \in \mathcal{V}^r$ as the sink node t. A violated subtour constraint is identified every time the value of the resulting minimum cut is less than 2. Then, we separate the corresponding subtour constraint in the form of inequalities (32). In the implementation, we use the minimum $s-t$ cut algorithm of the Concorde Callable Library (Applegate et al., 2018). With constraints (32), we do not need constraints (22) in the RMP anymore. Nevertheless, we keep constraints (22) because they help to improve the lower bound value of the variables $\theta_j, \forall\, j \in \mathcal{V}^r$.

3.1. Valid inequalities

We derive valid inequalities to strengthen the LP relaxation of the RMP model and improve the convergence of the branch-and-cut algorithm. Valid inequalities can be of great importance in Benders-based methods because the decomposition causes the master problem to lose information of the variables considered in the subproblems.

If some damaged nodes are used in the path from the depot to the demand node j, all these damaged nodes need to be repaired before the demand node j becomes accessible. Then, the time at which the demand node j becomes accessible is higher than or equal to the sum of the repair times of the damaged nodes used in the path plus the minimum travel time to arrive at these damaged nodes. Based on this, we can define the following valid inequalities:

$$Z_j^d \geq \sum_{k \in \mathcal{V}^r} V_{kj} \cdot (\delta_k + t_k^*),\ \forall\, j \in \mathcal{V}^d, \tag{33}$$

in which $t_k^* = \min\limits_{i \in \mathcal{V}_0^r : i \neq k} \{t_{ik}\},\ \forall\, k \in \mathcal{V}^r$.

The next set of valid inequalities is based on the maximum distance l_i allowed between the depot and the demand nodes. If the shortest distance from the depot to a node k plus the shortest distance from node k to the demand node i is greater than the maximum distance l_i, then node k cannot be used in the path from the depot to the demand node i. The valid inequalities are defined as follows:

$$V_{ki} \leq 0,\ \forall\, i \in \mathcal{V}^d, k \in \mathcal{V} : dist_{0k} + dist_{ki} > l_i, \tag{34}$$

in which $dist_{ki}$ is the shortest distance from node k to node i, and it is evaluated using Dijkstra's algorithm. In addition, we use all the valid inequalities proposed by Moreno et al. (2019) in our RMP.

4. Metaheuristic algorithms

In this section, we present a genetic algorithm (GA) and a simulated annealing (SA) tailored for the CSRP. These metaheuristics operate on a TSP subproblem representing the scheduling decisions and call for specialized algorithms to optimize the crew routing and the relief path decisions as well as to determine the feasibility and cost of the proposed schedule in the original CSRP. Therefore, the proposed metaheuristics do not explicitly consider all the possible crew routes and relief paths but only the best crew route and relief paths of a given scheduling solution, which significantly reduces the space of solutions explored by them. Additionally, they use five local search operators to diversify the search in the solution space. The way the operators are applied varies from one metaheuristic to the other, yet in both cases, they act as essential steps to escape from local optimal solutions. Such operators work by finding a neighbor of a current solution, and they can thus be seen as low-level heuristics that are selected by a higher-level algorithm (metaheuristic) that guides the search. This type of strategy has been referred to as hyper-heuristics (Burke et al., 2003; Drake et al., 2019), and successful applications of hyper-heuristics based on GA and SA metaheuristics can be found in the literature (Han and Kendall, 2003; Dowsland et al., 2007; Bai et al., 2012). Subsections 4.1 and 4.2 describe the proposed GA and SA methods, respectively.

4.1. Genetic algorithm

GA is a metaheuristic method based on three basic principles of the biological evolution process: reproduction, natural selection, and diversity of individuals. GAs have been used together with exact algorithms in a few applications (Lin et al., 2004; Poojari and Beasley, 2009) and are widely used in the context of routing problems (Karakatič and Podgorelec, 2015).

Algorithm 1 presents a basic scheme of the genetic algorithm developed in this work. GA starts with a population composed of a set of initial solutions (individuals) generated with a construction heuristic. The individuals are evaluated using a fitness function, and they evolve through a series of iterations (generations) by applying operators of selection of parents (natural selection), crossover (reproduction), and mutation (diversity of individuals). The procedure is repeated until it reaches some stopping criterion (e.g., maximum number of iterations or maximum computational time). The solution representation, construction heuristic, fitness function, selection, crossover and mutation of individuals are described in the following subsections.

Algorithm 1 Basic scheme of the GA metaheuristic.

```
1: Generate initial feasible individuals using the construction heuristic (see Section 4.1.3)
2: while stopping criteria are not reached do
3:     Evaluate the individuals with the feasibility and optimality check algorithms (see Section 4.1.2);
4:     Update the best solution;
5:     Perform selection (see Section 4.1.4) and crossover (see Section 4.1.5);
6:     Evaluate the individuals with the feasibility and optimality check algorithms (see Section 4.1.2);
7:     Perform mutation (see Section 4.1.6);
8: end while
```

4.1.1. Solution representation

Let $K = (v_0, v_1, \dots, v_i, \dots, v_{|\mathcal{V}^r|})$ be a schedule for the crew, in which v_i is the ith damaged node to be repaired and $v_0 = 0$ is the depot node. To represent the solution corresponding to schedule K, we use a vector with $|\mathcal{V}^r| + 2$ positions. The first $|\mathcal{V}^r| + 1$ positions indicate the order of the damaged nodes in the schedule of the crew, whereas the last position indicates the objective value of the solution in the original problem. Figure 2 shows an example of a vector representing a solution with 5 damaged nodes and an objective value of 100. Infeasible solutions are represented with an objective value of ∞ in the last position of the vector.

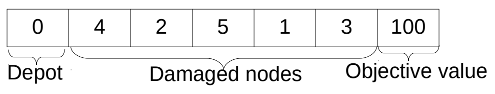


Figure 2: Vector representing a solution of the CSRP problem.

Note that only scheduling decisions are represented in a solution, as depicted in Figure 2. According to Moreno et al. (2019), for each fixed scheduling decision, we can optimally define the paths between damaged nodes and between the depot and the demand nodes using Dijkstra-based algorithms. Thus, the other decisions of the CSRP can be straightforwardly derived from a given crew schedule.

4.1.2. Evaluation of individuals

A fitness function in GA takes a candidate solution of the problem and returns a value that represents the quality of the solution. In our GA algorithm, the quality of a given solution is evaluated based on the cost of the corresponding schedule K in the original CSRP formulation. To evaluate this cost, we use two algorithms proposed in Moreno et al. (2019), which we call the feasibility check algorithm and the optimality check algorithm, as presented in Appendix A. The feasibility check algorithm takes a schedule K and verifies whether it is feasible or not for the original CSRP. If the solution is infeasible, we set a cost equal to ∞ in the last position of the solution vector. Otherwise, the algorithm returns the exact time at which the damaged nodes in schedule K are repaired by the crew. Subsequently, we call the optimality check algorithm that evaluates the exact time at which the demand nodes become accessible and the total cost for the solution in the original CSRP.

4.1.3. Initial population

Our initial population is created by using a construction heuristic that generates feasible solutions for the CSRP. The heuristic consists of generating a feasible solution by sequentially adding damaged nodes to the schedule in an iterative process. The construction heuristic is outlined in Algorithm 2. Let $K = (v_0, v_1, \ldots, v_l)$ be a partial schedule having the depot node $v_0 = 0$ and l nodes from set $\mathcal{V}^r$, for a given $l > 0$. Let $\mathcal{F}_K$ be the set of the nodes that can be reached from v_l, the last node added to this partial schedule. A node $j \in \mathcal{V}^r$ can be reached from v_l if there exists a path from v_l to j without using a damaged node that has not been repaired yet, and node j is not in the partial schedule K. In such a case, we say that j is a feasible node.

In the construction heuristic, we initially have $K = (v_0)$. At the end of each iteration, we randomly select a feasible node from the set $\mathcal{F}_K$, which is added to the end of the partial schedule K (line 12 of Algorithm 2). To update the set $\mathcal{F}_K$, we need to remove from it the last node v_l added to K (line 3), and then we execute Dijkstra's algorithm to find paths between node v_l and the nodes in $\mathcal{V}^r \setminus \mathcal{F}_K$ that are not in the partial schedule yet (line 6). If there is a path from v_l to some node $j \in \mathcal{V}^r \setminus \mathcal{F}_K$ that is not in K, then node j is added to the set $\mathcal{F}_K$ (line 8). Finally, when schedule K is completed, the solution is evaluated using the feasibility and optimality check algorithms (line 14). Notice that only feasible nodes can be selected at each iteration, and , as a consequence, feasible solutions are always generated.

Algorithm 2 construction heuristic for the CSRP.

```
1: Set node 0 as the first node in schedule $K$, i.e., $v_0 := 0$, $K = (v_0)$, $\mathcal{F}_K = \{v_0\}$;
2: for $l = 0$ to $|\mathcal{V}^r| - 1$ do
3:     Remove node $v_l$ from set $\mathcal{F}_K$;
4:     for $j = 1$ to $|\mathcal{V}^r|$ do
5:         if node $j$ is in $\mathcal{V}^r \setminus \mathcal{F}_K$ but not in $K$ then
6:             Find a path between node $v_l$ and node $j$ without using damaged nodes not repaired yet;
7:             if a path between node $v_l$ and node $j$ exists then
8:                 Add node $j$ to set $\mathcal{F}_K$;
9:             end if
10:        end if
11:    end for
12:    Randomly select a node $i \in \mathcal{F}_K$ and add it to schedule $K$, i.e., set $v_{(l+1)} := i$;
13: end for
14: Compute the cost of schedule $K$ using the feasibility and optimality check algorithms;
```

4.1.4. Parent selection

The selection of individuals to evolve from one generation to the next one is based on a tournament with k competitors, a so-called k-tournament, in which k individuals are randomly compared and the best (the one with the smallest cost) is selected for the reproduction step. A tournament with multiple competitors may yield good solutions quickly but has a higher chance of settling at local optima (Karakatič and Podgorelec, 2015). To avoid this issue, the k-tournament selection can be combined with aggressive mutation strategies. The selected best individuals can be crossed over by exchanging pieces with others and can either mutate or remain unaltered until the next generation. Additionally, we apply an elitist strategy that consists of always passing the best individual of the population from one generation to the next one.

4.1.5. Crossover

We implemented three popular crossover operators for permutation representation: partial mapped crossover (PMX), ordered crossover (OX), and cycle crossover (CX) (Larranaga et al., 1999; Kumar et al., 2012). When the individuals are selected for reproduction, one of the three strategies is randomly applied to generate the offspring.

4.1.6. Mutation

Mutation is usually used as an operation to prevent the GA from getting stuck on local optimal solutions (Balin, 2011). In our GA, the individuals are randomly selected for mutation if they represent feasible solutions. On the other hand, when an individual represents an infeasible solution, we force its mutation in an attempt to make it feasible. We use five local search operators as mutation strategies: swap, 2-opt, or-opt-1, or-opt-2, and or-opt-3. When the individuals are selected for mutation, one of these five strategies is randomly applied to generate the offspring. The first local search operator is an exchange (swap) of position 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. The three last local search operators are repositioning operators (or-opt-k) in which k adjacent nodes are removed from the schedule and reinserted at a different location in the schedule.

4.1.7. Parameters of the GA metaheuristic

In the GA proposed in this work, we have to adjust the following parameters to guarantee a better performance of the metaheuristic: maximum number of iterations, size of the population, selection probability, mutation probability, and parameter k for the k-tournament. The values selected for the parameters are described in Section 6.1.

4.2. Simulated annealing algorithm

SA is a randomized search method that exploits an analogy with the thermodynamic process of the cooling of metals, gradually adjusting a parameter called "temperature". At high temperatures, the method searches in a large space of solutions, while at low temperatures, solutions with worsening objective values are less likely to be accepted. SA has the advantage of usually being easier to implement and less time consuming than more sophisticated metaheuristics while still providing good overall results (Galvão et al., 2005), in particular for related problems such as the TSP (Ohlmann and Thomas, 2007) and the inventory routing problem (Alvarez et al., 2018). Additionally, as pointed out by Gogna and Tayal (2013) and enforced by our computational experiments, SA is well suited to problems with a large number of local optima, such as the CSRP.

A basic scheme of the SA implemented in this study is presented in Algorithm 3. The representation of the solution in our SA is the same as that used in our GA, i.e., a vector representing the scheduling decisions and the objective value of the solution. SA starts with a set of multiple initial solutions generated with the construction heuristic (Algorithm 2). At each iteration, one neighbor of each solution is randomly derived using one of the five local search operators (swap, 2-opt, or-opt-1, or-opt-2, and or-opt-3) defined in Subsection 4.1.6. The neighbors are evaluated by using the feasibility and optimality check algorithms presented in Appendix A. We perform a few inner iterations before updating the temperature value T ($T > 0$). At the end of the inner iterations, we update the temperature value, and all the solutions are replaced by the best solution found.

Algorithm 3 Basic scheme of the SA metaheuristic.

```
 1: Generate multiple initial feasible solutions using the construction heuristic (see Algorithm 2);
 2: while stopping criteria is not reached do
 3:     while inner iterations are not completed do
 4:         Apply local search operators to derive the neighbors of the solutions (see Section 4.1.6);
 5:         Evaluate the solutions with the feasibility and optimality check algorithms (see Appendix A);
 6:         Check the acceptance criteria and update the solutions (see Section 4.2.1);
 7:     end while
 8:     Update the best solution;
 9:     Update temperature;
10:     Replace all solutions by the best solution;
11: end while
```

4.2.1. Acceptance criteria

In the inner iterations of the proposed SA algorithm, a solution corresponding to a schedule K can be replaced by a neighbor solution corresponding to a different schedule K' according to an acceptance criterion probability given by $\min\{1, e^{\Delta/T}\}$, where $\Delta = z^K - z^{K'}$ and z^K ($z^{K'}$)

denotes the objective value of schedule K (K') in the CSRP. If $z^K \geq z^{K'}$, then $\min\{1, e^{\Delta/T}\} = 1$. In this case, as K' is better than or equal to K, it is accepted and replaces K. Otherwise, K' can still be accepted to replace K with probability $\min\{1, e^{\Delta/T}\} < 1$. If K' is infeasible, then it is not accepted because $\Delta \to -\infty$ and $\min\{1, e^{\Delta/T}\} \to 0$. Note that higher values of T at the beginning of the method imply higher probabilities of acceptance. On the other hand, smaller values of T in the last iterations of the SA imply smaller probabilities of accepting worse solutions.

4.2.2. Parameters of the SA metaheuristic

The SA proposed in this study has the following parameters that need to be adjusted by the user: maximum number of iterations, number of inner iterations per temperature, number of multiple initial solutions, initial temperature, minimum temperature, and cooling rate. The values selected for the parameters are described in Section 6.1.

5. Hybrid branch-and-Benders-cut algorithm (HBBC)

The hybrid approach combines the BBC algorithm developed in Section 3 and the metaheuristic algorithms proposed in Section 4. More specifically, we call a metaheuristic inside the branch-and-cut tree to explore the neighborhood of the current incumbent solution. The solutions found in the neighborhood of the current incumbent are used to add feasibility and optimality cuts to the RMP. The cuts added from the metaheuristic solutions are as defined in (29)-(31), and although they are not required to guarantee the optimality or feasibility of the solutions in the RMP, they can accelerate the convergence of the BBC method. The cuts required to guarantee the feasibility and optimality of the solutions in the RMP are still generated by solving subproblem SP. Therefore, the HBBC is an exact method.

Figure 3 depicts a basic scheme of the HBBC algorithm at each node of the branch-and-cut tree. At each node i, we solve the linear relaxation of the current RMP, denoted by LP_i. If LP_i is infeasible or if the objective value of the LP_i solution (OF_i) is greater than or equal to the objective value of the current incumbent solution, then node i is pruned. Otherwise, we solve the minimum cut problem to identify violated SECs of type (32). If necessary, we add the violated SECs, and LP_i is solved again. Otherwise, the integrality constraints are checked, and if any component of the binary variables is fractional in the solution of LP_i, then the branching is performed. Before performing the branching, we call a metaheuristic at the corresponding node i with the fractional solution. In this case, the initial metaheuristic solutions (initial population for GA or initial multistart for SA) are composed of (i) an integer solution obtained from a rounding heuristic; (ii) the incumbent solution; and (iii) solutions randomly generated. The rounding heuristic works as follows. Starting in the depot, let i be the last node added to schedule K, and select the damaged node j with the highest value $\widehat{X}_{ij}$ that has not been included in K so far. Then, include j at the end of K, and repeat the process iteratively until schedule K is completed.

Every time the LP_i solution is integer feasible, we call the subproblem SP to verify the violation of feasibility or optimality cuts. Since we keep constraints (22) in the RMP, there is no need to verify violated SECs for the integer solutions. If no feasibility or optimality cuts are obtained, then the LP$_i$ solution is feasible for the original problem and the solution is set as the new incumbent solution. Otherwise, LP$_i$ must be resolved, and the previous steps are applied again. If a feasible integer solution of the CSRP is found, we call the metaheuristic to improve this solution and generate additional cuts. In this case, the initial metaheuristic solutions are composed of the last found integer solution, the incumbent solution, and solutions randomly generated. If the metaheuristics find a better solution, the current incumbent solution is updated.

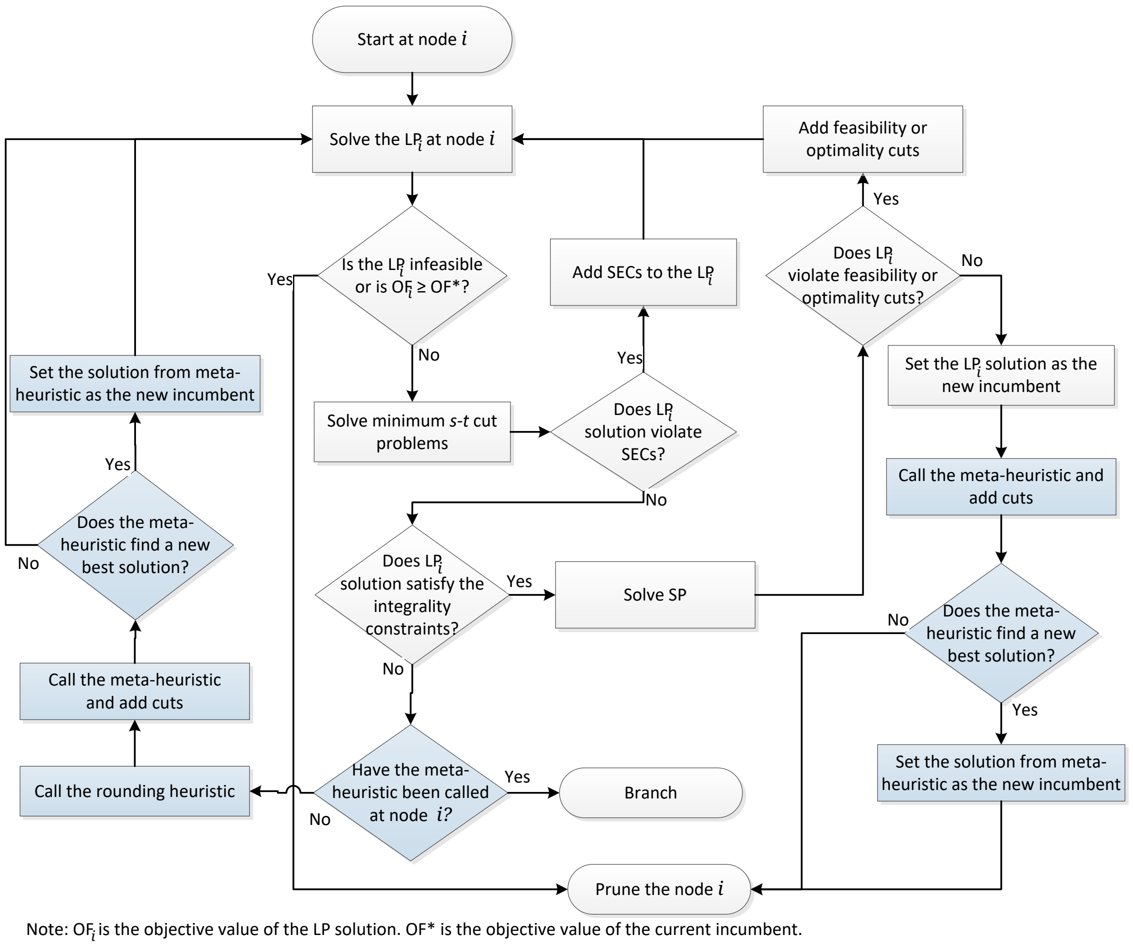

Start at node i
Solve the LP$_i$ at node i
Add feasibility or optimality cuts
Yes
Is the LP$_i$ infeasible or is OF$_i$ ≥ OF*?
Yes
No
Add SECs to the LP$_i$
Does LP$_i$ violate feasibility or optimality cuts?
No
Set the solution from meta-heuristic as the new incumbent
Solve minimum s-t cut problems
Yes
Does LP$_i$ solution violate SECs?
Set the LP$_i$ solution as the new incumbent
Call the meta-heuristic and add cuts
Yes
Does the meta-heuristic find a new best solution?
No
No
Does LP$_i$ solution satisfy the integrality constraints?
Yes
Solve SP
No
Does the meta-heuristic find a new best solution?
Yes
No
Call the meta-heuristic and add cuts
Call the rounding heuristic
Have the meta-heuristic been called at node i?
Yes
Branch
No
Set the solution from meta-heuristic as the new incumbent
Prune the node i
Note: OF$_i$ is the objective value of the LP solution. OF* is the objective value of the current incumbent.

Figure 3: Flowchart illustrating how the metaheuristics are combined with the BBC method.

Note that the metaheuristics are called at most once per node to avoid stagnation in infinite cycles. Since calling a metaheuristic at every node is inefficient because many unnecessary cuts may be added and perhaps much time spent, we call the metaheuristics only at nodes with an integer solution and at some predefined nodes with a fractional solution (for example, at the first 100 nodes of the tree). Similarly, it could be inefficient to call the separation procedure to add violated SECs at each node of the tree. Therefore, we call the separation procedure to detect violated SECs at the same nodes with a fractional solution for which the metaheuristics are called. The cuts generated by the metaheuristic in a node i with a fractional solution are immediately added to the LP$_i$ subproblem, and the subproblem is solved again. The cuts generated by the metaheuristic in a node i with an integer solution are added to a pool of constraints checked later in the tree. Some general-purpose optimization software can use automated cuts and/or heuristics that are not included in Figure 3.

6. Computational experiments

In this section, we present experimentation campaigns conducted with the scope of comparing the performance of the proposed solution methods. All the methods were coded in C++ programming language and run on a Linux PC with an Intel Core i7 CPU at 3.4 GHz and 16 GB of RAM using a single thread. SECs and Benders cuts are added using the Callback classes available in the Concert Technology Library. The RMP is solved by CPLEX Optimization Solver 12.7. The SP problem is solved by a specialized algorithm (Moreno et al., 2019) instead of using the CPLEX solver. To avoid running out of memory, we allow CPLEX to store the branch-and-bound tree in a file. As we use lazy constraints Callback, CPLEX automatically turns off nonlinear reductions and dual reductions. The stopping criterion on CPLEX 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 for most of the computational experiments. We also conduct a few experiments varying the default value of some CPLEX parameters, as reported in Section 6.3. For the metaheuristics GA and SA, the stopping criterion was given by either the elapsed time exceeding the time limit of 3,600 seconds or by performing 500 iterations without improving the best solution. As the metaheuristics SA and GA presented very similar overall results (as shown in Table 3), we performed the computational experiments of the HBBC method using only SA, which obtained solutions with a smaller average cost. The time limit of SA inside the nodes of the branch-and-cut tree was set as 600 seconds in the root node and 60 seconds in the other nodes.

The algorithms were tested using 390 benchmark instances from the literature (Maya-Duque et al., 2016; Moreno et al., 2019). Originally, the instances were derived from undamaged original networks by varying two parameters, namely, α and β. Parameter α defines the proportion of damaged arcs in the network. For example, $\alpha = 0.1$ indicates that 10% of the arcs of the original network are damaged. For each damaged arc, one or more damaged nodes were considered. Parameter β specifies the factor by which the distance between the depot and the demand nodes can increase in relation to the shortest distance. Let $dist_{0i}$ be the shortest distance between the depot and the demand node i. Parameter $\beta = 0.1$ indicates, for example, that $l_i = (1+0.1) \cdot dist_{0i}$. We consider instances generated from original networks with up to 100 nodes.

Table 1 shows the characteristics of the set of instances. The class and numbers of demand nodes, total nodes and arcs in the original networks can be seen in columns 1, 2, 3 and 4 of Table 1, respectively. The total numbers of nodes and arcs in the damaged networks can be seen in columns 5 and 6 of Table 1, respectively. These numbers 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 $\alpha = 50\%$. Thus, the damaged networks generated from original network 1 have from 27 to 45 nodes and from 40 to 60 arcs. The α and β parameters are shown in columns 7 and 8, respectively. By combining the values of α and β, 20 instances were generated for classes 1-15, and 10 instances were generated for classes 16-24, totaling 390 instances.

Table 1: Set of instances.

Original network (class)	Demand nodes	Original network Total nodes	Original network Total arcs	Damaged networks Total nodes	Damaged networks Total arcs	Values for α (%)		Values for β (%)		Total instances	
1	19	25	40	27 to 45	42 to 60	5, 10, 25, 30, 50		5, 10, 25, 50		20	
2	19	25	37	26 to 43	38 to 55	5, 10, 25, 30, 50		5, 10, 25, 50		20	
3	19	25	39	26 to 44	40 to 58	5, 10, 25, 30, 50		5, 10, 25, 50		20	
4	24	30	83	34 to 71	87 to 124	5, 10, 25, 30, 50		5, 10, 25, 50		20	
5	24	30	89	34 to 74	93 to 133	5, 10, 25, 30, 50		5, 10, 25, 50		20	
6	24	30	84	34 to 72	88 to 126	5, 10, 25, 30, 50		5, 10, 25, 50		20	
7	28	35	118	40 to 94	123 to 177	5, 10, 25, 30, 50		5, 10, 25, 50		20	
8	28	35	115	40 to 92	120 to 172	5, 10, 25, 30, 50		5, 10, 25, 50		20	
9	28	35	113	40 to 91	118 to 169	5, 10, 25, 30, 50		5, 10, 25, 50		20	
10	15	20	39	21 to 39	40 to 58	5, 10, 25, 30, 50		5, 10, 25, 50		20	
11	15	20	37	21 to 38	38 to 55	5, 10, 25, 30, 50		5, 10, 25, 50		20	
12	15	20	37	21 to 38	38 to 55	5, 10, 25, 30, 50		5, 10, 25, 50		20	
13	35	40	146	47 to 113	153 to 219	5, 10, 25, 30, 50		5, 10, 25, 50		20	
14	35	40	143	47 to 111	150 to 214	5, 10, 25, 30, 50		5, 10, 25, 50		20	
15	35	40	143	47 to 111	150 to 214	5, 10, 25, 30, 50		5, 10, 25, 50		20	
16	50	60	191	69 to 155	200 to 286	5, 25, 50	10, 30	05, 10	25, 50	6	4
17	50	60	197	69 to 158	206 to 295	5, 25, 50	10, 30	25, 50	05, 10	6	4
18	50	60	196	69 to 158	205 to 294	5, 25, 50	10, 30	05, 10	25, 50	6	4
19	70	80	247	92 to 203	259 to 370	5, 25, 50	10, 30	25, 50	05, 10	6	4
20	70	80	245	92 to 202	257 to 367	5, 25, 50	10, 30	05, 10	25, 50	6	4
21	70	80	248	92 to 204	260 to 372	5, 25, 50	10, 30	25, 50	05, 10	6	4
22	90	100	274	113 to 237	287 to 411	5, 25, 50	10, 30	05, 10	25, 50	6	4
23	90	100	271	113 to 235	284 to 406	5, 25, 50	10, 30	25, 50	05, 10	6	4
24	90	100	273	113 to 236	286 to 409	5, 25, 50	10, 30	05, 10	25, 50	6	4
Total										390	

6.1. Parameter tuning

As in most metaheuristics, the satisfactory performance of both the GA and the SA depends on the configuration choices for a set of key parameters. To appropriately calibrate such parameters, we use the ParamILS algorithm (Hutter et al., 2009). It is an automated tuning method that has shown very good performance in many applications (Montero et al., 2014). ParamILS iteratively improves the performance of a set of parameter configurations by searching in its neighborhood for another configuration with better quality. For this purpose, an initial configuration and discrete ranges for the set of parameters must be provided by the user. We also tested different strategies for the generation of cuts in the HBBC approach. The parameter tuning was carried out with 48 instances of different sizes, two from each class of Table 1. The instances were solved many times using different random seeds.

Table 2 shows the parameters of the metaheuristics GA and SA, the discrete ranges defined according to preliminary experiments for each parameter and their values in the best configuration found by ParamILS. Note that, for both the metaheuristics, the size of the initial solution set is relatively small (10 solutions). This behavior was expected since our evaluation of the solutions (feasibility and optimality check algorithms) is very expensive to be performed over a large set of solutions at each iteration. The other expected result was the high mutation probability for the GA. The CSRP is a degenerate problem (we can find several solutions with the same cost in the same neighborhood), which causes stagnation in local optimal solutions. The frequent application of mutation operators was shown to be a good strategy to escape from local optima. Recall that the mutation probability is defined only for feasible solutions, as the infeasible solutions are always forced to mutate.

Table 2: Ranges and best configurations for the parameters of the GA, SA and HBBC methods.

Solution method	Parameter	Tested values	Final value
GA	Size of the population	{5, 10, 15, 20, 25}	10
	Selection probability	{0.90, 0.92, 0.94, 0.96, 0.98}	0.98
	Mutation probability	{0.05, 0.1, 0.2, 0.3, 0.4, 0.5}	0.5
	Parameter k	{2, 3, 4, 5}	3
SA	Size of the set of initial solutions	{5, 10, 15, 20, 25}	10
	Inner iterations per temperature	{2, 3, 4, 5}	2
	Initial temperature	{100, 300, 500, 1000, 1500}	500
	Minimum temperature	{1E-6, 1E-4, 0.01, 0.1}	1E-6
	Cooling rate	{0.95, 0.96, 0.97, 0.98, 0.99}	0.99
HBBC	Frequency	(1) Only at root node (600 seconds)[1] and at nodes with integer solutions (30 seconds).[1] (2) (1) + every 100 nodes (30 seconds).[1] (3) (1) + at the first 100 nodes (30 seconds).[1] (4) (1) + decreasing frequency.[2]	(4)
	Maximum number of solutions	$\{50, 100, 500, 5 \cdot \mathcal{V}^r, 10 \cdot \mathcal{V}^r, 20 \cdot \mathcal{V}^r\}$	$10 \cdot \mathcal{V}^r$
	Type of cuts	(1) Optimality + feasibility cuts. (2) Only optimality cuts. (3) Only feasibility cuts.	(2)

[1] Time limit set for the metaheuristics at the nodes.
[2] From node 0 to node 100, we call the metaheuristics at every 10 nodes; from node 100 to node 1000, we call the metaheuristics at every 100 nodes; and from node 1000 onward, we call the metaheuristics at every 1000 nodes.

For the HBBC, we varied the frequency of application of the metaheuristics in the branch-and-cut tree and the quantity and type of cuts that are added to the master problem from the metaheuristic solutions. Table 2 also shows the parameters tested for the HBBC method. For instance, the parameter frequency for the HBBC method defines how often the metaheuristics are called in the branch-and-cut tree. Four options were evaluated for this parameter: (1) apply the metaheuristics only at the first three nodes and at nodes with an integer solution; (2) apply the metaheuristics at the nodes of option (1) and additionally at every 100 nodes with a fractional solution; (3) apply the metaheuristics at the nodes of option (1) and additionally at the first 100 nodes with a fractional solution; and (4) apply the metaheuristics at the nodes of option (1) and at nodes with a fractional solution in a decreasing frequency. The decreasing frequency consists of calling the metaheuristics at every 10 nodes for nodes 0 to 100; at every 100 nodes for nodes 100 to 1000; and at every 1000 nodes for node 1000 onward. The last strategy showed the best performance, as we observe that it is more likely to find new best solutions with the metaheuristics at the first nodes of the tree when the incumbent solution is farther from the optimal solution than in the last nodes of the tree. Additionally, it is more likely to generate useful cuts in the first nodes because fewer cuts have been added to the RMP. Regarding the type of cuts derived from the metaheuristic solutions, it is more useful to add only optimality cuts in the HBBC, as they have a greater impact on the lower bound than feasibility cuts. Finally, we limited the number of solutions of the metaheuristics from which we can add cuts to the RMP since the generation of a large number of cuts can slow down the linear subproblems at the nodes. The computational experiments presented in the next sections were conducted using the best configuration found by the ParamILS algorithm. The separation routines for the SECs at nodes with fractional solutions are called at the same nodes as the metaheuristics.

6.2. Computational performance of the metaheuristic approaches

This section presents the results of the metaheuristics GA and SA. Before running the computational experiments with all the instances described in Table 1, we solved each one of the 48 instances used in the parameter tuning 10 times with different random seeds. The objective of this experiment was to analyze the differences in the quality of the solutions provided by the metaheuristics. The tests were performed setting a time limit of 3,600 seconds for each run. The average, maximum and minimum objective values over the ten repetitions for each class are shown in Table 3. We also evaluated the relative difference of the maximum value with respect to the minimum value ($\frac{\text{Max}-\text{Min}}{\text{Min}}$). Note that, on average, the maximum value over the ten repetitions was only 1.57% and 1.17% higher than the minimum value for metaheuristics GA and SA, respectively. In fact, for some instances, the result over the ten repetitions was the same, while for the other instances, the difference of the maximum value in relation to the minimum value was never higher than 8.12% for the GA and 8.28% for SA. Furthermore, such difference was higher than 4% in only 5 out of the 24 instance classes in the metaheuristic GA and in only 1 out of the 24 classes in the metaheuristic SA. Basically, in the instances with a smaller number of damaged nodes, the space of solutions to be explored is smaller than in the other instances, and the metaheuristics are able to find similar near-optimal solutions over the ten repetitions.

Table 3: Average results of the SA and GA metaheuristics for the instances used in the parameter tuning.

	GA					SA				
Class	Avg. cost	Min. cost	Max. cost	$\frac{\text{Max}-\text{Min}}{\text{Min}}$ (%)	Avg. time (sec.)	Avg. cost	Min. cost	Max. cost	$\frac{\text{Max}-\text{Min}}{\text{Min}}$ (%)	Avg. time (sec.)
1	22,675	22,675	22,675	0.00	7.71	22,675	22,675	22,675	0.00	2.04
2	76,367	76,367	76,367	0.00	5.12	76,367	76,367	76,367	0.00	1.44
3	60,645	60,645	60,645	0.00	6.72	60,645	60,645	60,645	0.00	1.99
4	50,479	50,479	50,479	0.00	23.91	50,479	50,479	50,479	0.00	6.94
5	36,944	36,944	36,944	0.00	29.18	36,944	36,944	36,944	0.00	9.12
6	60,841	60,841	60,841	0.00	16.57	60,841	60,841	60,841	0.00	6.35
7	50,416	50,416	50,416	0.00	36.38	50,416	50,416	50,416	0.00	13.82
8	69,196	69,196	69,196	0.00	60.77	69,196	69,196	69,196	0.00	19.62
9	64,564	64,564	64,564	0.00	53.16	64,564	64,564	64,564	0.00	20.57
10	68,605	68,605	68,605	0.00	5.65	68,605	68,605	68,605	0.00	1.52
11	76,268	76,268	76,268	0.00	5.44	76,268	76,268	76,268	0.00	1.48
12	64,619	64,619	64,619	0.00	3.37	64,619	64,619	64,619	0.00	0.95
13	46,179	46,179	46,179	0.00	68.70	46,179	46,179	46,179	0.00	19.37
14	129,183	129,108	129,407	0.23	95.22	129,108	129,108	129,108	0.00	36.43
15	74,357	74,357	74,357	0.00	99.06	74,376	74,357	74,431	0.10	37.77
16	89,097	88,956	89,338	0.43	256.58	89,051	88,956	89,298	0.38	110.53
17	44,169	43,971	44,333	0.82	172.76	44,040	43,878	44,452	1.31	74.38
18	134,518	134,024	135,526	1.12	629.64	134,042	132,332	136,241	2.95	209.90
19	64,070	62,389	66,253	6.19	942.34	61,808	61,057	62,626	2.57	382.37
20	85,896	81,169	87,756	8.12	682.38	76,732	75,390	78,039	3.51	261.72
21	77,520	74,540	79,548	6.72	1,352.30	69,449	68,696	70,790	3.05	529.08
22	246,686	245,172	247,895	1.11	2,129.45	238,805	235,724	244,200	3.60	854.04
23	100,642	97,625	105,440	8.01	2,154.62	95,131	92,772	100,453	8.28	677.67
24	188,774	184,512	193,558	4.90	2,511.13	182,371	180,337	184,546	2.33	1,055.83
Avg.	82,613	81,817	83,384	1.57	472.84	80,946	80,433	81,749	1.17	180.62

Note in Table 3 that the average solution cost is slightly smaller when the problem is solved by SA, which provided solutions with an average objective value 2.02% better than the solutions of the GA for an execution time of 3,600 seconds. Figure 4 shows the improvement of the average objective function value of the 48 instances solved by the metaheuristics SA and GA along 3,600 seconds of execution in relation to the average objective function value of the initial solutions. As expected, we observed that for both the metaheuristics, the reduction in the objective function value is faster in the first iterations. For the GA (SA), the average cost of the initial solutions is 272,302 (291,194), which decreases to 106,813 (109,055) in 600 seconds and to 82,613 (80,946) in 3,600 seconds. Thus, for the instances solved by the GA (SA), the average objective function value decreases by 60.77% (62.55%) in the first 600 seconds and by 69.66% (72.20%) in 3,600 seconds. The SA presented a slightly better performance in terms of the objective function values since the problem typically has many local optimal solutions with the same objective value in a neighborhood and SA has the ability to avoid becoming trapped in local optimal solutions. Moreover, the average computational time of SA was 61.8% smaller than that of the GA. It is worth mentioning that we also ran additional experiments with a longer time limit (3 hours), but the overall improvement was less than 1% on average.

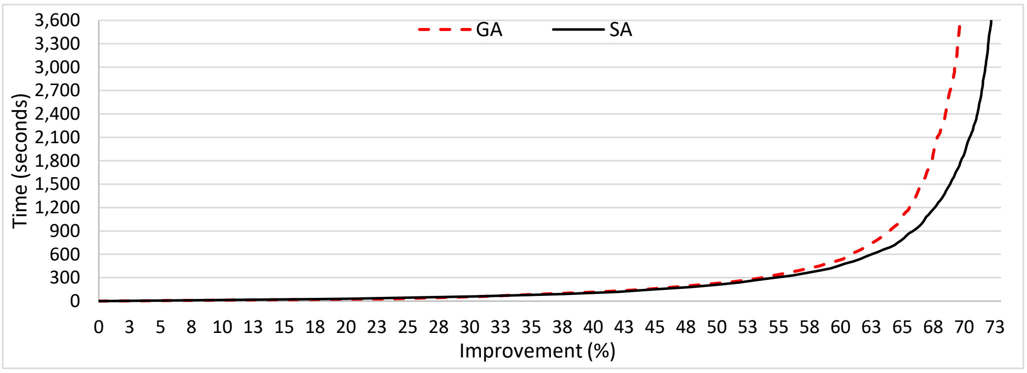


Figure 4: Average cost of the 48 instances solved by SA and the GA along 3,600 seconds of execution.

Table 4 shows the average results of the metaheuristic SA considering all instances of the classes presented in the first column. Since we observed a robust behavior over the 10 runs of our SA in the results reported in Table 3, we carried out the experiment reported in Table 4 with a single run of this method. We compare the results of our SA with the results of the GRASP metaheuristic of Maya-Duque et al. (2016), which was the only metaheuristic so far developed to solve the same variant of the problem addressed in this work. We present the results of GRASP for classes 1-15 only because we did not have access to the solutions of the other classes. We also compare the results of our SA with the best variant of the exact BBC of Moreno et al. (2019), referred to as GR-BBC0.

Columns 6 and 7 of Table 4 show the relative reduction in the objective value of the solutions found with the SA with respect to the solutions found with GRASP and GR-BBC0, respectively. On average, for classes 1-15, the cost of the solutions of the metaheuristic SA decreases by 1.46% and 0.18% in relation to the cost of the solutions of GRASP and GR-BBC0, respectively. The improvements with respect to GRASP were up to 7.02% in classes 1-15. In relation to GR-BBC0, the higher improvement in classes 1-15 was only 2.22% since the solutions obtained with GR-BBC0 for these instances are close to the optimal solutions. For the other classes, the improvements were up to 55%. The improvement for the larger instance classes comes from the effectiveness of the local search operators that helps SA to escape from local optimal solutions. The GR-BBC0 approach proposed by Moreno et al. (2019) stagnates in local optima for larger instances of the problem.

6.3. Computational performance of the exact approaches

In this section, we analyze the performance of our BBC and HBBC methods. Table 5 summarizes the different solution strategies that we tested. The first two strategies BBC1 and

Table 4: Average results of the metaheuristic SA for the different instance classes (time limit of 3,600 seconds).

	Objective function value			$\frac{\text{GRASP}-\text{SA}}{\text{GRASP}}$	$\frac{\text{GR-BBC0}-\text{SA}}{\text{GR-BBC0}}$
Class	GRASP[1]	GR-BBC0[1]	SA	(%)	(%)
1	9,745	9,745	9,745	0.00	0.00
2	34,089	34,089	34,089	0.00	0.00
3	49,987	49,862	49,862	0.25	0.00
4	18,247	18,037	18,122	0.68	−0.47
5	18,152	18,485	18,074	0.43	2.22
6	21,253	20,917	20,917	1.58	0.00
7	36,873	36,511	36,511	0.98	0.00
8	26,382	26,049	26,146	0.90	−0.37
9	35,224	33,953	33,903	3.75	0.15
10	48,546	48,460	48,460	0.18	0.00
11	39,213	38,538	38,765	1.14	−0.59
12	28,876	28,037	28,037	2.91	0.00
13	23,536	23,566	23,528	0.03	0.16
14	87,163	81,032	81,042	7.02	−0.01
15	52,085	52,201	51,385	1.34	1.56
16	–	38,737	38,851	–	−0.30
17	–	30,448	30,537	–	−0.29
18	–	97,476	95,516	–	2.01
19	–	65,093	46,048	–	29.26
20	–	71,173	42,037	–	40.94
21	–	75,602	59,025	–	21.93
22	–	211,487	146,526	–	30.72
23	–	98,827	58,012	–	41.30
24	–	209,435	93,868	–	55.18
Avg. 1-15	35,291	34,632	34,572	1.41	0.18
Avg. 16-24	–	99,809	67,824	–	24.53
Avg. All	–	59,073	47,042	–	9.31

The character "–" indicates no available value.
[1] GRASP proposed in Maya-Duque et al. (2016) and BBC proposed in Moreno et al. (2019).

BBC2 compare the new Benders reformulation of the problem with and without using the valid inequalities (VIs) defined in Section 3.1. The BBC3 strategy shows the impact of adding the SECs dynamically together with the Benders cuts. All the HBBC strategies are based on BBC3. In the HBBC1 strategy, the SA metaheuristic is used in the root node to find good-quality initial solutions for the problem but without generating feasibility and/or optimality cuts. In the HBBC2 method, on the other hand, the metaheuristic is additionally called in the nodes with integer solutions and in some nodes with fractional solutions to improve the incumbent solution and to generate Benders cuts.

The modification of some default parameters of the solver can positively influence the performance of the branch-and-cut method (Baz et al., 2009; Moreno et al., 2016, 2018). Therefore, the BBC3 and HBBC2 approaches were also tested varying the default configuration of the solver CPLEX to solve the RMP, leading to BBC3* and HBBC2*. For both, we changed parameters that could lead to improvements in the lower bound of the solutions. We modified the emphasis of the branch-and-cut algorithm to optimality rather than feasibility, setting the CPLEX parameter `MIPEmphasis = 3`. This configuration increases the lower bound faster but possibly with a poor detection of feasible solutions along the optimization. We also set the selection of nodes to be processed according to the node with the smallest objective function for the associated LP relaxation, setting the CPLEX parameter `NodeSel = 1`. This strategy looks at the nodes with smaller bounds to improve them first. Finally, we modified the order of separation of the different types of cuts at nodes with fractional solutions, using method *isAfterCutLoop()* of the callback procedures. In the root node, we keep the default settings of CPLEX, while in the remaining nodes, we call the metaheuristic and add our Benders cuts only after all automatized cuts of CPLEX have been generated (i.e., if *isAfterCutLoop()* returns `True`). This way, we avoid generating Benders cuts on fractional solutions that may be cut off by subsequent automatized cuts of CPLEX.

In the last strategy of Table 5, GR-HBBC2*, we apply a graph reduction (GR) strategy used in the literature (Moreno et al., 2019) to speed up the convergence of the solution method. Basically, the idea of the GR is to solve the problem over different subgraphs with a reduced number of demand and damaged nodes and derive lower bounds for the variables of the original problem based on the solution of the reduced subgraphs. The subgraphs are usually generated from an initial feasible solution of the problem, and the performance of the GR strategy highly depends on this initial solution. Since we do not have a trivial initial solution for the BBC strategies, we consider the GR only with the best HBBC approach. A description of the GR strategy is provided in Appendix B.

Table 5: Characteristics of the solution methods.

Solution method	Description
BBC1	New Benders reformulation of the problem.
BBC2	BBC1 + VIs.
BBC3	BBC2 + SECs.
BBC3*	BBC3 varying the default configurations of some CPLEX parameters.
HBBC1	BBC3 + SA in the root node.
HBBC2	HBBC1 + SA in nodes with integer solution and in some nodes with fractional solutions.
HBBC2*	HBBC2 varying the default configurations of some CPLEX parameters.
GR-HBBC2*	HBBC2* + Graph reduction strategy.

Figure 5 presents the performance profiles (Dolan and Moré, 2002) for the proposed approaches. The performance is based on the optimality gap, computed as $gap = \frac{Z^U - Z^L}{Z^U}$, in which Z^U is the upper bound or cost of the best integer solution and Z^L is the lower bound. The value $P(f, q)$ (y-axis) when $q > 0$ (x-axis) indicates the fraction of instances for which a strategy f provides solutions with a gap within a factor of 2^q of the best obtained gap. The value of $P(f, q)$ when $q = 0$ represents the fraction of instances for which the strategy f reached the best gap. For a given instance, the best gap is the lowest gap found considering all the approaches. Clearly, the hybrid strategies outperform the standalone BBC strategies. Also, we can observe that the GR strategy significantly improves the performance of the HBBC approaches.

Table 6 shows the number of optimal solutions (#opt), the proportion of optimal solutions (%opt), the average bounds and gap, and the average elapsed time of the different solution strategies. For the sake of comparison, the table also shows the results of the methods BBC0 and GR-BBC0 of Moreno et al. (2019), which are so far the best exact approaches in the literature. BBC0 uses a simple heuristic to provide an initial solution for the BBC algorithm, while GR-BBC0 additionally relies on graph reduction. The average results of the exact solution methods for different instance classes grouped according to the size of the network are presented in the Appendix C.

Note that the use of the valid inequalities improves the performance of the solution method, mainly in relation to the average lower bound and average gap. The average lower bound increases by 17.60%, from 17,427 in BBC1 to 20,494 in BBC2. The average gap is reduced from

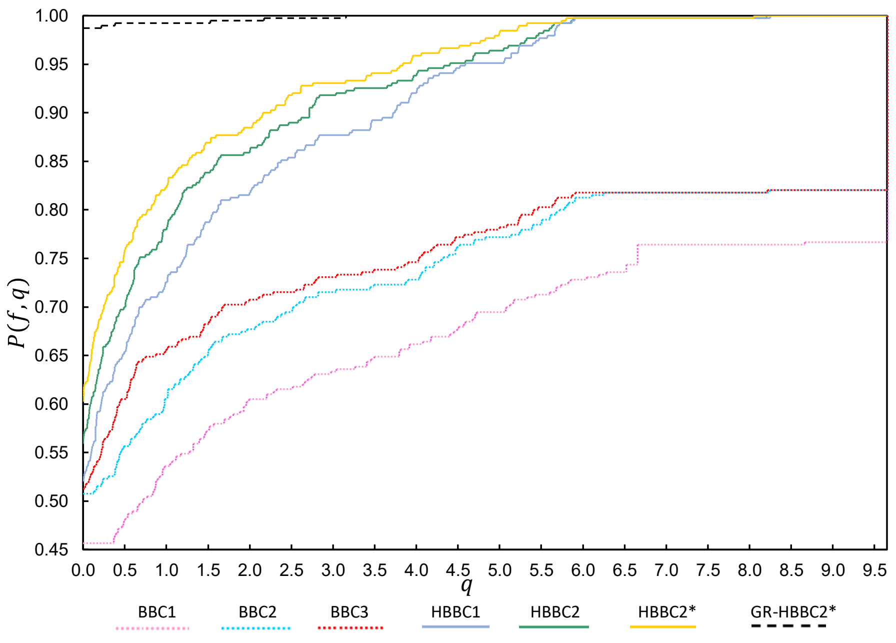


Figure 5: Performance profiles based on gap for the proposed solution methods.

Table 6: Comparison of the exact BBC and HBBC approaches.

Solution method	#ins	#opt	%opt	Avg. upper bound	Avg. lower bound	Avg. gap (%)	Avg. time (sec.)	Avg. best time[1](sec.)
BBC1	390	169	43.33	113,048	17,427	28.73	2,238.42	2,058.13
BBC2	390	171	43.85	110,763	20,494	25.03	2,201.55	1,889.79
BBC3	390	171	43.85	110,291	20,499	24.99	2,203.59	1,891.82
BBC3*	390	171	43.85	117,355	21,157	24.67	2,215.35	1,910.44
HBBC1	390	201	51.54	42,377	23,625	21.01	1,762.74	59.23
HBBC2	390	215	55.13	42,187	24,089	19.75	1,646.56	92.48
HBBC2*	390	215	55.13	42,158	24,855	18.15	1,651.39	83.35
BBC0[2]	390	186	47.69	53,342	16,042	31.87	1,906.38	450.57
GR-HBBC2*	390	239	61.28	42,118	29,836	10.90	1,473.87	101.07
GR-BBC0[2]	390	209	53.59	49,673	27,521	14.90	1,672.49	350.25

[1] Time spent to find the best upper bound.
[2] BBC proposed in Moreno et al. (2019) with (GR-BBC0) and without (BBC0) the GR strategy.

28.73% to 25.03%, a reduction of 12.87%. The average cost of the solutions and the average computational time are not significantly affected by the addition of the valid inequalities, while BBC2 proves optimality for two additional instances with respect to BBC1. BBC2 and BBC3 present a similar performance, with a slight improvement with the use of the SECs in the BBC3 algorithm. Differently from the traditional TSP and VRP problems, in which the SECs have yielded good results, the changes caused by the SECs in the solutions of the LP problems to eliminate the subtours do not appear to directly affect the objective function of such LP problems.

The use of the metaheuristic SA in the root node (HBBC1) to warm-start the BBC algorithm significantly improves the result of the BBC3 strategy, mainly with respect to the upper bound and the time spent to find the best solution. The average cost of the solutions is reduced from 110,291 to 42,376, a 61.58% reduction. The average time spent to find the best solution is 32 times smaller in HBBC1, being reduced from 1,891 to 59 seconds, a 96.87% reduction. The average lower bound and the average gap are also improved by the HBBC1 strategy. The average lower bound increases by 15.25%, from 20,499 to 23,625, while the average gap is reduced by 15.95%, from 24.99 to 21.01. Furthermore, HBBC1 proves optimality for 30 additional instances in relation to BBC3. In the HBBC2 algorithm, the use of the SA metaheuristic to derive cuts and tighten the linear relaxation improves the convergence of the method. The average gap decreases from 21.01% to 19.75%. Although the reduction in the gap appears insignificant, we can observe that HBBC2 proves the optimality for 14 additional instances in relation to the HBBC1 method. In fact, for some instance classes, the reduction in the average gap was up to 26%. Note also that HBBC2 increases the average lower bound of the solutions, while it reduces the average cost with respect to the solutions obtained with the HBBC1 approach.

Regarding the impact of varying the default configuration of CPLEX in strategies BBC3 and HBBC2, Table 6 shows that BBC3* obtained better lower bounds and gaps but worse upper bounds on average since the new configuration emphasizes the improvement of the lower bound rather than having a good-quality solution. For HBBC2*, the negative effect of prioritizing the lower bound is neutralized by the use of the metaheuristic to improve the upper bound. The average gap was reduced from 19.75 to 18.15, a reduction of 8%, with respect to HBBC2. The reduction was up to 21% for some instances. When compared with BBC0, HBBC2* reduces the average upper bound and gap by 20.97% and 43.05%, respectively, and increases the lower bound by 54.94%.

Finally, the graph reduction strategy significantly improved the results of the HBBC2* approach. The GR-HBBC2* obtained the optimal solutions for 61.28% of the instances and reduced the average gap to 10.9%, a reduction of 39.94% with respect to the HBBC2* strategy. Table 7 shows the average results of GR-HBBC2* compared with the results of the GR-BBC0 strategy. Table 7 also presents the ratio of the GR-HBBC2* solutions in relation to the GR-BBC0 solutions evaluated as $\frac{\text{Value in GR-BBC0}-\text{Value in GR-HBBC2*}}{\text{Value in GR-BBC0}}$. A ratio higher than zero indicates a reduction in the value of the GR-HBBC2* method, whereas a ratio smaller than zero indicates an increase in the value.

Evidently, GR-HBBC2* outperforms GR-BBC0. For example, the average reduction in the upper bound considering all the instances is 15.21%. For some instances, the reduction is up to 42.97%. Similarly, the improvement of the lower bound in some instance classes is up to 36.15%. The average gap was reduced by 26.17% with GR-HBBC2*, whereas the time spent to find the best solution was reduced by 71.14% on average, considering all the instances. To confirm whether the performances of GR-HBBC2* and GR-BBC0 are statistically different in terms of the upper bound, lower bound, gap and time, we carried out Friedman statistical tests (Conover, 1999) for the instance classes presented in Table 7. The null hypothesis is that there is no significant performance difference between GR-BBC0 and GR-HBBC2*. Table 8 gives the corresponding p values for the Friedman tests. Regarding the lower bound, gap and best time, we observe that the null hypothesis is rejected for every instance class at confidence levels ranging between 0.000 and 0.045. Therefore, the performances of GR-BBC0 and GR-HBBC2* are significantly different with respect to the lower bound, gap and best time in each one of the considered instance classes. For the upper bound and total computational time, although the differences are not statistically significant for some instance classes, the strategy GR-HBBC2* obtained always the same or better average results than GR-BBC0*.

Table 7: Comparison of the GR-HBBC2* algorithm with the BBC approach from the literature.

Solution method	Instance classes	#ins	#opt	%opt	Avg. upper bound	Avg. lower bound	Avg. gap (%)	Avg. time (sec.)	Avg. best time[1](sec.)
GR-BBC0[2]	1, 2, 3	60	48	80.00	31,232	28,237	3.29	867.81	142.58
	4, 5, 6	60	33	55.00	19,146	14,865	9.84	1,499.68	260.53
	7, 8, 9	60	28	46.67	32,171	23,223	14.16	1,904.82	33.98
	10, 11, 12	60	48	80.00	38,345	35,623	2.60	749.47	164.62
	13, 14, 15	60	24	40.00	52,266	33,991	16.71	2,132.32	126.54
	16, 17, 18	30	12	40.00	55,554	34,985	18.68	2,142.24	529.64
	19, 20, 21	30	10	33.33	70,623	18,801	37.39	2,410.55	1,024.54
	22, 23, 24	30	6	20.00	173,250	32,110	42.70	2,881.42	1,542.53
	All instances	390	209	53.59	49,673	27,521	14.77	1,672.49	350.25
GR-HBBC2*	1, 2, 3	60	54	90.00	31,232	30,076	1.02	713.16	0.14
	4, 5, 6	60	42	70.00	19,006	15,667	5.27	1,086.71	2.20
	7, 8, 9	60	32	53.33	32,145	25,431	9.49	1,744.60	5.51
	10, 11, 12	60	49	81.67	38,345	36,122	2.10	737.94	2.88
	13, 14, 15	60	31	51.67	51,963	36,521	12.72	1,746.30	74.48
	16, 17, 18	30	13	43.33	54,588	36,713	14.88	2,047.63	301.98
	19, 20, 21	30	12	40.00	48,757	19,810	31.94	2,173.21	290.58
	22, 23, 24	30	6	20.00	98,812	43,716	33.71	2,881.99	550.99
	All instances	390	239	61.28	42,118	29,836	10.90	1,473.87	101.07
Ratio (%)	1, 2, 3		12.50	12.50	0.00	6.51	−69.11	−17.82	−99.90
	4, 5, 6		27.27	27.27	−0.73	5.40	−46.38	−27.54	−99.15
	7, 8, 9		14.29	14.29	−0.08	9.51	−33.02	−8.41	−83.77
	10, 11, 12		2.08	2.08	0.00	1.40	−19.19	−1.54	−98.25
	13, 14, 15		29.17	29.17	−0.58	7.44	−23.87	−18.10	−41.14
	16, 17, 18		8.33	8.33	−1.74	4.94	−20.34	−4.42	−42.98
	19, 20, 21		20.00	20.00	−30.96	5.37	−14.59	−9.85	−71.64
	22, 23, 24		0.00	0.00	−42.97	36.15	−21.05	0.02	−64.28
	All instances		14.35	14.35	−15.21	8.41	−26.17	−11.88	−71.14

[1] Time spent to find the best upper bound.
[2] BBC proposed in Moreno et al. (2019) that also uses GR.

Table 8: The statistic values (p values) of the Friedman test for GR-BBC0 vs GR-HBBC2*.

Instance Class	Upper bound	Lower bound	Gap (%)	Time (sec.)	Best time[1](sec.)
1, 2, 3	**0.366**	0.007	0.039	**0.606**	0.000
4, 5, 6	**0.245**	0.020	0.005	0.039	0.000
7, 8, 9	**0.606**	0.001	0.000	**0.053**	0.001
10, 11, 12	**0.121**	0.039	0.039	**0.606**	0.000
13, 14, 15	**0.197**	0.007	0.007	**0.897**	0.007
16, 17, 18	**0.465**	0.045	0.028	**0.361**	0.005
19, 20, 21	0.028	0.018	0.003	**0.197**	0.001
22, 23, 24	0.003	0.000	0.000	**0.051**	0.001

[1] Time spent to find the best upper bound.
p values > 0.05 are highlighted in **bold**.

7. Conclusions and future research

This paper has proposed two novel metaheuristics, a branch-and-Benders-cut (BBC) algorithm and a hybrid approach (HBBC), to solve the road restoration crew scheduling and routing problem (CSRP). The metaheuristics are the first genetic algorithm and simulated annealing proposed for the CSRP. They are based on the decomposition of the problem into smaller sub-problems and the use of specialized algorithms to evaluate the candidate solutions. The BBC is based on an improved Benders reformulation of the problem and enhances previous approaches by using a different variable partitioning scheme. Valid inequalities for the problem have been proposed as well. The HBBC is an exact hybrid method that uses a metaheuristic to obtain good-quality solutions at early stages of the search tree as well as to improve the performance of solving the master problem by exploring the neighborhood of the incumbent solutions to generate more effective Benders cuts.

The results of extensive computational experiments with 390 benchmark instances showed that both metaheuristics outperformed the only metaheuristic available in the literature for the CSRP. For the BBC approach, the new variable partitioning scheme and the proposed valid inequalities were shown to be effective to increase the lower bound of the master problem. The computational results also provide evidence that the combination of the metaheuristic with the BBC, resulting in the hybrid algorithm HBBC, significantly reduces the cost of the solutions and the time spent to find good-quality solutions. The lower bound and gap of the solutions were also improved with the use of the metaheuristic within the BBC, especially when additional cuts from the neighborhood of the master problem solutions are generated.

We have observed that, while the BBC presented a solution with a higher cost when varying the parameters of the solver, the HBBC is able to take advantage of the new parameter configuration to improve the lower bound, the gap and the cost of the solutions. Basically, the HBBC counteracts the elevation of the cost in the BBC by exploring the neighborhood of the master problem solutions. By incorporating the graph reduction technique in the HBBC, we observed a significant reduction in the average gap, mainly because the graph reduction helps to increase the lower bound of the solutions. With the GR-HBBC, we effectively reduce the cost, gap and computational time of most of the instances with respect to the best exact approach proposed in the literature. In addition to their theoretical relevance, the improvements obtained with the proposed approaches may also have great value to aid decision making in practice. The reduction in the cost of the solutions directly impacts the time at which the demand nodes are accessible from the supply node, thus reducing the time that victims in the affected areas wait for supplies, evacuation, rescue and medical assistance.

Several avenues for future work can be identified. The use of other reformulation techniques to solve the problem, such as the Dantzig-Wolfe decomposition, followed by a branch-price-and-cut method, could be explored for the large-scale instances. Extending the problem and solution approaches to consider distinct characteristics such as multiple crews, relief distribution and an alternative objective function to take into account network vulnerability, for example, is also a promising research direction. Since the information about the actual situation of the damaged nodes after the extreme events is limited, and the consequences of the extreme events over the transportation networks cannot be accurately predicted, another encouraging area of investigation relies on considering the inherent uncertainties present in the CSRP.

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 numbers 141973/2016-1 and 304601/2017-9]. The third author expresses thanks for the support of both the British Academy via BA/Leverhulme Small Research Grant SRG18R1\180939 and the UEBS First Grant Venture Fund.

Appendix A. Algorithms to check feasibility and cost of the schedule solutions

Let $K = (v_0, v_1, ..., v_{(h-1)}, v_h, ..., v_p, ..., v_{|\mathcal{V}^r|})$ be a schedule for the crew, where v_i is the ith damaged node to be repaired and $v_0 = 0$. The Algorithm 4 finds the optimal paths between damaged nodes when a schedule K is fixed. The algorithm is executed over the graph $G = (\mathcal{V}, \mathcal{E})$. Initially, the cost C_e of each arc $e \in \mathcal{E}$ is set to ∞ if the arc e is adjacent to a damaged node or equal to τ_e otherwise (line 1 of Algorithm 4). Iteratively, the cost C_e of each arc adjacent to damaged nodes is reset as $\tau_e + \delta_{v_j}$ (line 3), and Dijkstra's algorithm is used to find the shortest path between nodes v_{j-1} and v_j (line 4). If a path between nodes v_{j-1} and v_j exists, the cost $\mathcal{C}$ of the path must be less than ∞ and the value of variable $Z^r_{v_j}$ is updated (line 6). Otherwise, the solution is infeasible, and a cost ∞ is defined for schedule K.

Algorithm 4 feasibility check algorithm (adapted from Moreno et al. (2019)).

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

If Algorithm 4 finishes with feasible paths between the damaged nodes, Algorithm 5 is used to find the cost of the solution in the original CSRP. Algorithm 5 finds the optimal paths between the depot and the demand nodes. Initially, the cost C_e of the arcs in the network is set as ℓ_e for all the arcs (line 1 of Algorithm 5). Then, the cost of each arc adjacent to the damaged nodes is set as ∞ (line 3), starting with the last damaged node ($v_{|\mathcal{V}^r|}$) in the schedule of the crew. Dijkstra's algorithm is used to find the shortest path between node v_0 and all the demand nodes $i \in \mathcal{V}^d$ (line 4). For a given node v_j, if the cost $\mathcal{C}_i$ to reach a demand node i is larger than the maximum distance l_i (line 6), we conclude that node v_j is necessary to find a path with a cost smaller than the maximum distance l_i, so the time instant Z^d_i in which the demand node i becomes accessible must be set as $Z^r_{v_j}$ (line 7). Note that we update Z^d_i only if it was not updated in previous iterations; thus, 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, we calculate the cost of schedule K (line 11), and such cost is defined as the solution value in the last position of the vector that represents this solution in the metaheuristic algorithms (see Figure 2 for an illustration).

Algorithm 5 optimality check algorithm (adapted from Moreno et al. (2019)).

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

Appendix B. Graph reduction strategy

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. G^{LF} is defined 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. The subgraph G^{LF} is further reduced by removing transshipment nodes that are not directly connected to damaged nodes. For each node i removed from G^{LF}, the arcs adjacent to this node are deleted, and new arcs are created 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 is denoted by $\bar{G}^{LF}$. From a feasible solution of the CSRP defined using $\bar{G}^{LF}$, valid inequalities can be derived for the original problem, as pointed out in Proposition 3.

Proposition 3. *Given $L \subseteq \mathcal{V}^r$ and $F \subseteq \mathcal{V}^d$, let $K^{\bar{G}^{LF}}$ be the schedule corresponding to an optimal solution of the CSRP, defined using the reduced graph $\bar{G}^{LF}$ 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 related to $K^{\bar{G}^{LF}}$, for all $i \in F$. Then, the following inequalities are valid for the RMP of the original CSRP defined using graph G:*

$$\sum_{i \in F:\, d_i \cdot \widehat{\theta}_i^{\bar{G}^{LF}} > 0} d_i \cdot \theta_i \geq \widehat{\Theta}^{\bar{G}^{LF}}, \tag{B.1}$$

$$\sum_{j \in L} V_{ji} \geq 1, \; \forall i \in F : \widehat{\theta}_i^{\bar{G}^{LF}} > 0. \tag{B.2}$$

Proof. The proof of equation (B.1) is given in Moreno et al. (2019). If $\widehat{\theta}_i^{\bar{G}^{LF}} > 0$, the relief paths to demand node i use at least one of the damaged nodes in set L, and thus the equation (B.2) is valid. □

Appendix C. Additional computational results

Tables C.9 and C.10 present the average results of the BBC and HBBC solution methods for different classes of instances grouped according to the size of the network.

Table C.9: Comparison of the exact BBC solution approaches.

Solution method	Instance classes	#ins	#opt	%opt	Avg. upper bound	Avg. lower bound	Avg. gap (%)	Avg. time (sec.)	Avg. best time[1](sec.)
BBC1	1, 2, 3	60	43	71.67	30,502	24,411	8.14	1,139.93	1,044.04
	4, 5, 6	60	30	50.00	6,336	10,325	9.40	1,839.95	1,680.40
	7, 8, 9	60	24	40.00	19,330	15,950	16.98	2,193.77	2,023.08
	10, 11, 12	60	41	68.33	38,539	31,004	7.31	1,236.69	1,174.83
	13, 14, 15	60	20	33.33	8,196	19,246	3.06	2,526.45	2,328.36
	16, 17, 18	30	8	26.67	297,915	1,768	73.33	2,782.49	2,503.31
	19, 20, 21	30	2	6.67	171,547	12,581	63.13	2,959.21	2,765.48
	22, 23, 24	30	1	3.33	538,291	26,534	72.45	3,407.97	3,137.27
	All	390	169	43.33	113,048	17,427	28.73	2,238.42	2,058.13
BBC2	1, 2, 3	60	43	71.67	32,151	25,295	6.72	1,073.87	1,056.54
	4, 5, 6	60	30	50.00	8,627	6,520	7.74	1,813.94	1,573.02
	7, 8, 9	60	24	40.00	15,232	7,226	13.91	2,167.96	1,895.85
	10, 11, 12	60	43	71.67	38,694	31,977	6.29	1,057.97	1,007.75
	13, 14, 15	60	20	33.33	8,196	7,881	2.29	2,472.36	2,194.44
	16, 17, 18	30	8	26.67	297,915	32,875	51.86	2,754.94	2,258.25
	19, 20, 21	30	2	6.67	171,547	15,830	55.38	2,929.91	2,615.22
	22, 23, 24	30	1	3.33	538,291	36,207	66.48	3,374.23	2,987.47
	All	390	171	43.85	110,763	20,494	25.03	2201.55	1,889.79
BBC3	1, 2, 3	60	43	71.67	32,151	25,328	6.71	1,080.51	1,003.94
	4, 5, 6	60	30	50.00	8,704	6,528	7.76	1,815.49	1,595.92
	7, 8, 9	60	24	40.00	15,306	7,222	13.93	2,224.59	2,127.39
	10, 11, 12	60	43	71.67	38,694	31,978	6.29	1,059.75	1,020.17
	13, 14, 15	60	20	33.33	8,205	7,881	2.29	2,474.54	2,030.34
	16, 17, 18	30	8	26.67	295,963	32,881	51.78	2,758.30	2,250.02
	19, 20, 21	30	2	6.67	166,133	15,834	55.30	2,926.88	2,606.95
	22, 23, 24	30	1	3.33	526,655	36,212	65.85	3,324.88	3,051.20
	All	390	171	43.85	110,291	20,499	24.99	2,203.59	1,891.82
BBC3*	1, 2, 3	60	43	71.67	32,151	26,098	5.51	1,153.71	1,135.08
	4, 5, 6	60	30	50.00	8,897	7,044	6.50	1,814.71	1,598.69
	7, 8, 9	60	24	40.00	15,159	7,646	12.68	2,098.03	1,734.70
	10, 11, 12	60	43	71.67	38,694	32,564	5.56	1,101.18	1,052.99
	13, 14, 15	60	20	33.33	8,541	8,131	2.20	2,470.04	2,192.38
	16, 17, 18	30	8	26.67	357,327	33,251	53.27	2,751.29	2,205.26
	19, 20, 21	30	2	6.67	198,129	16,378	54.31	2,928.72	2,624.16
	22, 23, 24	30	1	3.33	580,125	36,540	66.57	3,424.34	3,033.84
	All	390	171	43.85	117,355	21,157	24.67	2,215.35	1,910.44

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

Table C.10: Comparison of the exact HBBC solution approaches.

Solution method	Instance classes	#ins	#opt	%opt	Avg. upper bound	Avg. lower bound	Avg. gap (%)	Avg. time (sec.)	Avg. best time[1](sec.)
HBBC1	1, 2, 3	60	48	80.00	31,232	25,355	6.06	753.86	0.28
	4, 5, 6	60	34	56.67	19,267	11,038	17.12	1,562.06	1.62
	7, 8, 9	60	29	48.33	32,326	16,930	23.99	1,872.77	9.09
	10, 11, 12	60	47	78.33	38,345	31,782	6.33	815.79	0.63
	13, 14, 15	60	21	35.00	52,084	26,009	27.05	2,343.41	24.79
	16, 17, 18	30	10	33.33	55,102	32,789	23.20	2,402.79	88.30
	19, 20, 21	30	8	26.67	49,574	15,830	43.12	2,644.80	268.34
	22, 23, 24	30	4	13.33	99,715	36,271	45.67	3,172.22	340.58
	All	390	201	51.54	42,377	23,625	21.01	1,762.74	59.23
HBBC2	1, 2, 3	60	51	85.00	31,232	26,645	4.48	634.24	0.19
	4, 5, 6	60	38	63.33	19,038	11,527	15.04	1,331.01	2.42
	7, 8, 9	60	30	50.00	32,154	17,132	23.06	1,824.65	7.15
	10, 11, 12	60	47	78.33	38,345	32,075	5.96	817.21	0.88
	13, 14, 15	60	26	43.33	51,997	26,584	24.81	2,063.03	30.03
	16, 17, 18	30	12	40.00	54,637	32,985	22.25	2,174.72	103.51
	19, 20, 21	30	8	26.67	49,078	15,942	42.17	2,648.38	356.15
	22, 23, 24	30	3	10.00	99,181	36,309	45.60	3,241.94	661.24
	All	390	215	55.13	42,187	24,089	19.75	1,646.56	92.48
HBBC2*	1, 2, 3	60	51	85.00	31,232	27,396	3.54	630.57	0.32
	4, 5, 6	60	38	63.33	19,006	12,340	12.54	1,333.82	2.57
	7, 8, 9	60	30	50.00	32,145	18,209	20.86	1,864.03	13.77
	10, 11, 12	60	47	78.33	38,345	32,755	5.23	806.55	1.48
	13, 14, 15	60	26	43.33	51,963	27,505	23.57	2,068.16	32.22
	16, 17, 18	30	12	40.00	54,637	33,369	20.86	2,172.90	155.64
	19, 20, 21	30	8	26.67	48,934	16,482	40.07	2,648.29	327.90
	22, 23, 24	30	3	10.00	99,096	36,854	43.56	3,240.62	499.23
	All	390	215	55.13	42,158	24,855	18.15	1,651.39	83.35
GR-HBBC2*	1, 2, 3	60	54	90.00	31,232	30,076	1.02	713.16	0.14
	4, 5, 6	60	42	70.00	19,006	15,667	5.27	1,086.71	2.20
	7, 8, 9	60	32	53.33	32,145	25,431	9.49	1,744.60	5.51
	10, 11, 12	60	49	81.67	38,345	36,122	2.10	737.94	2.88
	13, 14, 15	60	31	51.67	51,963	36,521	12.72	1,746.30	74.48
	16, 17, 18	30	13	43.33	54,588	36,713	14.88	2,047.63	301.98
	19, 20, 21	30	12	40.00	48,757	19,810	31.94	2,173.21	290.58
	22, 23, 24	30	6	20.00	98,812	43,716	33.71	2,881.99	550.99
	All	390	239	61.28	42,118	29,836	10.90	1,473.87	101.07

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

References

Adulyasak, Y., Cordeau, J.-F., Jans, R., 2015. Benders Decomposition for Production Routing Under Demand Uncertainty. Operations Research 63 (4), 851–867.
URL http://pubsonline.informs.org/doi/10.1287/opre.2015.1401

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.

Alvarez, A., Munari, P., Morabito, R., 2018. Iterated local search and simulated annealing algorithms for the inventory routing problem. International Transactions in Operational Research 25 (6), 1785–1809.
URL http://doi.wiley.com/10.1111/itor.12547

Applegate, D., Bixby, R., Chvátal, V., Cook, W., 2018. Concorde TSP solver. Accessed on 01/02/2018.
URL www.tsp.gatech.edu/concorde.html

Arslan, O., Karaşan, O. E., 2016. A Benders decomposition approach for the charging station location problem with plug-in hybrid electric vehicles. Transportation Research Part B: Methodological 93, 1339–1351.

Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., McCollum, B., 2012. A simulated annealing hyper-heuristic methodology for flexible decision support. 4OR 10 (1), 43–66.
URL http://link.springer.com/10.1007/s10288-011-0182-8

Balin, S., 2011. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Information Sciences 181 (17), 3551–3569.
URL http://linkinghub.elsevier.com/retrieve/pii/S0020025511001782

Baz, M., Hunsaker, B., Prokopyev, O., mar 2009. How much do we "pay" for using default parameters? Computational Optimization and Applications 48 (1), 91–108.
URL http://link.springer.com/10.1007/s10589-009-9238-5

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S., 2003. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. Springer US, Boston, MA, pp. 457–474.
URL https://doi.org/10.1007/0-306-48056-5_16

Conover, W., 1999. Practical nonparametric statistics, john wiley & sons. INC, New York.

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.

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

Dowsland, K. A., Soubeiga, E., Burke, E., 2007. A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. European Journal of Operational Research 179 (3), 759–774.
URL https://linkinghub.elsevier.com/retrieve/pii/S0377221705007356

Drake, J. H., Kheiri, A., Özcan, E., Burke, E. K., 2019. Recent Advances in Selection Hyper-heuristics. European Journal of Operational Research.
URL https://linkinghub.elsevier.com/retrieve/pii/S0377221719306526

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

Fischetti, M., Ljubic, I., Sinnl, M., 2017. Redesigning Benders Decomposition for Large-Scale Facility Location. Management Science 63 (7), 2146–2162.
URL http://pubsonline.informs.org/doi/10.1287/mnsc.2016.2461

Galvão, R. D., Chiyoshi, F. Y., Morabito, R., 2005. Towards unified formulations and extensions of two classical probabilistic location models. Computers & Operations Research 32 (1), 15–33.
URL https://linkinghub.elsevier.com/retrieve/pii/S0305054803002004

Gendron, B., Lucena, A., da Cunha, A. S., Simonetti, L., 2014. Benders Decomposition, Branch-and-Cut, and Hybrid Algorithms for the Minimum Connected Dominating Set Problem. INFORMS Journal on Computing 26 (4), 645–657.
URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.2013.0589

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

Gogna, A., Tayal, A., dec 2013. Metaheuristics: review and application. Journal of Experimental & Theoretical Artificial Intelligence 25 (4), 503–526.
URL http://dx.doi.org/10.1080/0952813X.2013.782347

Han, L., Kendall, G., 2003. Guided Operators for a Hyper-Heuristic Genetic Algorithm. In: Lecture Notes in Computer Science. Vol. 2903. pp. 807–820.
URL http://link.springer.com/10.1007/978-3-540-24581-0{_}69

Hutter, F., Hoos, H. H., Leyton-Brown, K., Stützle, T., 2009. Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36 (1), 267–306.
URL http://dl.acm.org/citation.cfm?id=1734953.1734959

Jamshidi, A., Hajizadeh, S., Su, Z., Naeimi, M., Núñez, A., Dollevoet, R., Schutter, B. D., Li, Z., 2018. A decision support approach for condition-based maintenance of rails based on big data analysis. Transportation Research Part C: Emerging Technologies 95, 185 – 206.
URL http://www.sciencedirect.com/science/article/pii/S0968090X18309859

Karakatič, S., Podgorelec, V., 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing Journal 27, 519–532.

Kasaei, M., Salman, F. S., 2016. Arc routing problems to restore connectivity of a road network. Transportation Research Part E: Logistics and Transportation Review 95, 177–206.
URL http://dx.doi.org/10.1016/j.tre.2016.09.012

Kumar, R., Kumar, N., Karambir, 2012. A Comparative Analysis of PMX , CX and OX Crossover operators for solving Travelling Salesman Problem. International Journal of Latest Research in Science and Technology 1 (2), 98–101.
URL http://www.mnkjournals.com/ijlrst{_}files/download/Vol1Issue2/303-Naveen.pdf

Larranaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S., 1999. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 13 (Holland 1975), 129–170.

Li, H., Jian, X., Chang, X., Lu, Y., 2018. The generalized rollon-rolloff vehicle routing problem and savings-based algorithm. Transportation Research Part B: Methodological 113, 1–23.
URL https://doi.org/10.1016/j.trb.2018.05.005

Lin, Z.-Z., Bean, J. C., White, C. C., 2004. A Hybrid Genetic/Optimization Algorithm for Finite-Horizon, Partially Observed Markov Decision Processes. INFORMS Journal on Computing 16 (1), 27–38.
URL http://pubsonline.informs.org/doi/10.1287/ijoc.1020.0024

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

Montero, E., Riff, M.-c., Neveu, B., 2014. A beginner's guide to tuning methods. Applied Soft Computing 17, 39–51.
URL http://dx.doi.org/10.1016/j.asoc.2013.12.017

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

Moreno, A., Alem, D., Ferreira, D., Clark, A., 2018. An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. European Journal

of Operational Research 269 (3), 1050–1071.
URL https://doi.org/10.1016/j.ejor.2018.02.022

Moreno, A., Munari, P., Alem, D., may 2019. A branch-and-Benders-cut algorithm for the Crew Scheduling and Routing Problem in road restoration. European Journal of Operational Research 275 (1), 16–34.
URL https://doi.org/10.1016/j.ejor.2018.11.004

Morshedlou, N., González, A. D., Barker, K., 2018. Work crew routing problem for infrastructure network restoration. Transportation Research Part B: Methodological 118, 66–89.
URL https://linkinghub.elsevier.com/retrieve/pii/S0191261518303539

Ohlmann, J. W., Thomas, B. W., 2007. A Compressed-Annealing Heuristic for the Traveling Salesman Problem with Time Windows. INFORMS Journal on Computing 19 (1), 80–90.

Öncan, T., Altinel, I. K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Computers & Operations Research 36 (3), 637–654.
URL http://linkinghub.elsevier.com/retrieve/pii/S0305054807002468

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

Poojari, C. A., Beasley, J. E., 2009. Improving benders decomposition using a genetic algorithm. European Journal of Operational Research 199 (1), 89–97.
URL http://www.sciencedirect.com/science/article/pii/S0377221708009740

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 259 (3), 801–817.
URL http://dx.doi.org/10.1016/j.ejor.2016.12.005

Raidl, G. R., 2015. Decomposition based hybrid metaheuristics. European Journal of Operational Research 244 (1), 66–76.
URL http://linkinghub.elsevier.com/retrieve/pii/S0377221714009874

Rei, W., Cordeau, J.-F., Gendreau, M., Soriano, P., 2009. Accelerating Benders Decomposition by Local Branching. INFORMS Journal on Computing 21 (2), 333–345.
URL http://pubsonline.informs.org/doi/10.1287/ijoc.1080.0296

Salazar-González, J.-J., Santos-Hernández, B., 2015. The split-demand one-commodity pickup-and-delivery travelling salesman problem. Transportation Research Part B: Methodological 75, 58–73.
URL https://linkinghub.elsevier.com/retrieve/pii/S0191261515000429

Shao, S., Sherali, H. D., Haouari, M., 2017. A Novel Model and Decomposition Approach for the Integrated Airline Fleet Assignment, Aircraft Routing, and Crew Pairing Problem. Transportation Science 51 (1), 233–249.
URL http://pubsonline.informs.org/doi/10.1287/trsc.2015.0623

Shin, Y., Kim, S., Moon, I., 2019. Integrated optimal scheduling of repair crew and relief vehicle after disaster. Computers and Operations Research 105, 237–247.
URL https://doi.org/10.1016/j.cor.2019.01.015

Su, Z., Jamshidi, A., Núñez, A., Baldi, S., De Schutter, B., 2019. Integrated condition-based track maintenance planning and crew scheduling of railway networks. Transportation Research Part C: Emerging Technologies 105 (June), 359–384.
URL https://doi.org/10.1016/j.trc.2019.05.045

Taşkin, Z. C., Cevik, M., 2013. Combinatorial Benders cuts for decomposing IMRT fluence maps using rectangular apertures. Computers & Operations Research 40 (9), 2178–2186.

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.

Zhang, Y., D'Ariano, A., He, B., Peng, Q., 2019. Microscopic optimization model and algorithm for integrating train timetabling and track maintenance task scheduling. Transportation Research Part B: Methodological 127, 237–278.
URL https://doi.org/10.1016/j.trb.2019.07.010