



Innovative Applications of O.R.

## Network repair crew scheduling and routing for emergency relief distribution problem

Pablo A. Maya Duque<sup>a,\*</sup>, Irina S. Dolinskaya<sup>b</sup>, Kenneth Sörensen<sup>c</sup><sup>a</sup>INCAS, Faculty of Engineering, Universidad de Antioquia, Medellín, Colombia<sup>b</sup>Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, United States<sup>c</sup>ANT/OR, Faculty of Applied Economics, University of Antwerp, Antwerp, Belgium

### ARTICLE INFO

#### Article history:

Received 22 December 2014

Accepted 11 June 2015

Available online 20 June 2015

#### Keywords:

Network repair

Repair crew scheduling

Repair crew routing

Dynamic programming

Iterated greedy-randomized constructive procedure

### ABSTRACT

Every year, hundreds of thousands of people are affected by natural disasters. The number of casualties is usually increased by lack of clean water, food, shelter, and adequate medical care during the aftermath. One of the main problems influencing relief distribution is the state of the post-disaster road network. In this paper, we consider the problem of scheduling the emergency repair of a rural road network that has been damaged by the occurrence of a natural disaster. This problem, which we call the Network Repair Crew Scheduling and Routing Problem addresses the scheduling and routing of a repair crew optimizing accessibility to the towns and villages that demand humanitarian relief by repairing roads. We develop both an exact dynamic programming (DP) algorithm and an iterated greedy-randomized constructive procedure to solve the problem and compare the performance of both approaches on small- to medium-scale instances. Our numerical analysis of the solution structure validates the optimization model and provides managerial insights into the problem and its solutions.

© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

### 1. Introduction

Every year, hundreds of thousands of people are affected by natural disasters such as floods and earthquakes, especially in less developed regions of the planet (Guha-Sapir, Vos, Below, & Ponsérre, 2011). An important observation, however, is that the number of casualties is usually increased by lack of clean water, food, shelter, and adequate medical care during the aftermath (PAHO, 2000). An adequate logistics reply to a disaster is therefore crucial. It has been conjectured that 80 percent of the disaster relief effort consists of logistics (Trunick, 2005). To a first approximation, therefore, disaster relief is logistics.

One of the main problems influencing the delivery of food, shelter, and medical supplies to affected regions is the state of the road network. In many situations, it is not a lack of supplies that kills people, but the impossibility to get those supplies to the people that need it. In Haiti, for example, extensive media coverage of the 2010 earthquake resulted in a large excess stock of relief supplies. However, distributing those supplies to the affected villages proved far more difficult as road infrastructure had been damaged or destroyed (Pedraza Martínez, Stapleton, & Van Wassenhove, 2010; Van

Wassenhove, Martínez, & Stapleton, 2010). Further complicating the repair of the road network is the often limited availability of road repair capacity, especially in impoverished regions of the world. Therefore, it is crucial that road repair is planned and executed in the most efficient way possible.

In this paper, we consider the problem of scheduling and routing the emergency repair crew of a rural road network that has been damaged by the occurrence of a natural disaster. We call this problem the *Network Repair Crew Scheduling and Routing Problem*, abbreviated as NRCSR. The NRCSR addresses the scheduling and routing of a single repair crew, starting from a single depot, while optimizing accessibility to the towns and villages that demand humanitarian relief. Extending this work to more than one repair crew and more than one depot is left for future research.

The problem is defined on an undirected and connected graph  $G = (\mathcal{V}, \mathcal{E})$  of which the nodes ( $\mathcal{V}$ ) are either *demand* nodes ( $\mathcal{V}_d$ ) or *damaged* nodes requiring *repair* ( $\mathcal{V}_r$ ). (Note that without loss of generality, we represent a damaged road link by a node located in the middle of the corresponding edge. Therefore, repairing a road connection is equivalent to repairing a node, and we use the two terms interchangeably.) There is one supply node that corresponds to the location in which the relief supplies are positioned and from which the repair crew initially departs. This node is called the *depot*. Demand nodes correspond to locations (usually villages) that

\* Corresponding author: Tel.: +57 4 219 55 75.

E-mail address: [pmayaduque@gmail.com](mailto:pmayaduque@gmail.com) (P.A. Maya Duque).

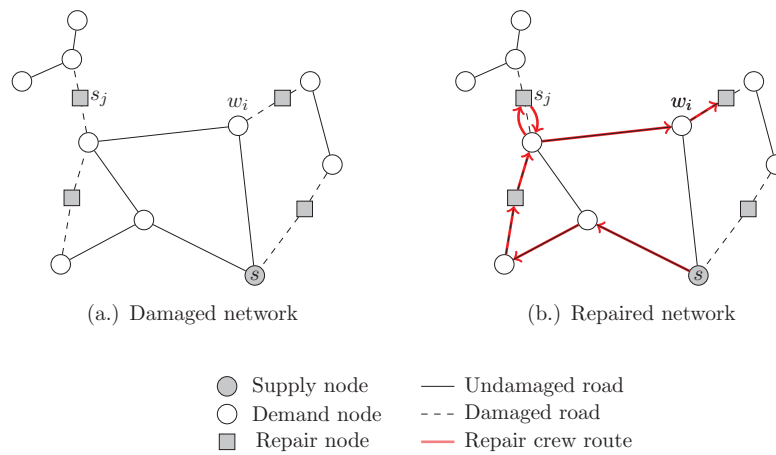


Fig. 1. Example of a damaged network and repair crew route for NRCSR.

demand humanitarian relief. The importance of a village (node  $i$ ) is represented by a *demand*, i.e., a weight factor  $w_i$ , that might, e.g., correspond to the number of inhabitants. Damaged (repair) nodes represent the locations where the work of the repair crew is needed. Such nodes have a *repair time*  $s_j$  (for node  $j$ ) that represents the time the repair crew spends on its first visit. Without loss of generality, we do not distinguish between demand nodes and transshipment nodes, i.e., cross points where two or more roads come together, since such nodes can be modeled as demand nodes with zero demand. Each edge  $e_{ij} \in \mathcal{E}$  represents a road that connects two nodes  $i, j \in \mathcal{V}$ . A travel time  $t_{ij}$  is defined for each edge  $e_{ij}$  to represent the time it takes the repair crew to traverse it.

The aim of this problem is to determine the optimal sequence in which the crew should traverse the graph, starting from the depot node. Every time it encounters a damaged node that it has not visited before, it repairs the node and incurs the *repair time* of this node. On subsequent visits the crew can pass that node without incurring any additional time. When damaged nodes are repaired, demand nodes can become *accessible*. A demand node  $i$  is called accessible if there exists a path connecting this node to the depot that contains only undamaged and/or repaired nodes, and is not longer than a certain maximum distance  $D_i$ . The maximum distance  $D_i$  is node-specific and can be computed based on pre-disaster conditions (e.g., the distance between a demand node and the depot should not be more than twice the distance it was before the disaster). Thus, in addition to the travel time  $t_{ij}$ , each edge  $e_{ij}$  has a distance measure, denoted  $d_{ij}$ , that is used to evaluate nodes' accessibility.

For each demand node, the schedule of the crew determines the moment in time at which this node becomes accessible. The objective function of the problem is the sum of the moments at which each demand node becomes accessible weighted by the node demand  $w_i$ . The objective of the network repair problem is to determine the schedule and route of the repair crew that minimizes this objective function. In general, it is not necessary for the repair crew to visit all damaged nodes. Fig. 1 illustrates an example problem and solution for the NRCSR with four damaged nodes.

The NRCSR captures the most important aspects of the problem of routing a road repair crew in order to restore accessibility to demand points affected by damaged roads. To solve this problem, we develop two very different approaches. A dynamic programming (DP) model is able to find optimal solutions to small and medium-sized instances and provides insight into the fundamental structure of the problem. An efficient iterated greedy-randomized constructive procedure (IGRCP) is able to solve medium- to large-scale instances in very small computing times. A comparison of the performance of both approaches yields interesting insight into the underlying problem structure.

The problem of scheduling the repair operations in a post-disaster situation can be modeled in many different ways and the exact formulation of the NRCSR is the result of several design choices. As mentioned, one of the main causes of human casualties is the fact that a disaster tends to disconnect remote population centers from the main supply hub. In poorer countries around the world, people are usually dependent on a very limited number of economic centers (usually larger cities) for sustenance. The demand nodes in the NRCSR may therefore represent small villages or settlements, temporary medical facilities, shelters, or food distribution points. The supply node on the other hand will represent a city, an airport, or a port area where the relief goods arrive. When these demand points can no longer be reached by road from the supply point, the results are often dramatic. This is especially the case in rural areas, that lack the redundancy in the road network that characterizes cities. For this reason, the NRCSR is explicitly defined on *sparse* networks, that are more common in rural than in urban situations, and the graphs underlying all generated test instances are sparse.

By explicitly considering the routing of the repair crew, the NRCSR—unlike most previous contributions—adds a crucial time component that makes the problem more realistic and more suitable for real-life post-disaster situations. Like in real life, roads cannot be repaired until they can be reached by a repair crew. As mentioned, the objective function of the NRCSR minimizes the demand-weighted moment in time at which all the demand nodes are connected to the supply node. The demand assigned to each demand node should be interpreted as a function of the importance of connecting this node, e.g., its population size. The objective function also takes the urgency of connecting the nodes into account, as even less important demand nodes (with small demand) may cause a large objective function increase when ignored for too long. In a sense, the NRCSR models aspects of both the immediate response (routing of the repair crew) and the recovery (connecting the demand nodes as soon as possible) phases.

In a post-disaster immediate response situation, budget constraints are not usually a predominant issue, and the focus is on restoring connectivity of the affected regions with the limited physical means available. Nevertheless, an in-depth analysis of solutions produced by our solution approaches for the NRCSR reveal interesting managerial results with considerable implications for further budgetary analysis. More specifically, our analysis shows that only a limited number of damaged nodes (around 30 percent) needs to be repaired in order to restore full accessibility. Additionally, when studying the optimal or best-known solutions, we observe that the repair operations have, on average, a diminishing rate of return, connecting a lot of demand in the beginning and much less demand at the end. However, this simple observation hides a large amount of

variation, and cannot be compressed into some simple rule-of-thumb that could replace an efficient algorithm. Moreover, we are able to demonstrate the importance of planning the repair several stages ahead by comparing our results to a greedy myopic heuristic.

The remainder of this paper is organized as follows. The next section reviews the literature on related network repair problems. In Sections 3 and 4 we develop two radically different solution approaches for the NRCSR: a dynamic programming method and an iterated greedy-randomized constructive procedure (IGRCP), respectively. Both approaches are compared in Section 5, and the strong and weak points of each method are exposed. Section 5 also contains an analysis of the structure of optimal or high-quality solutions and managerial insights obtained from the numerical results. Conclusions and suggestions for future research can be found in Section 6. A detailed list of symbols used in the paper can be found in Appendix A. For a mixed-integer linear programming formulation of the problem, which we had to forgo as an inefficient solution method, readers are referred to Maya Duque, Dolinskaya, and Sorensen (2014, Appendix).

## 2. Literature review

In recent years, the field of Operations Research and Management Sciences (OR/MS) has turned its attention to disaster management and humanitarian logistics in a big way. As pointed out by authors such as Ergun, Karakus, Keskinocak, Swann, and Villarreal (2010) and Van Wassenhove (2003), this subject is growing in importance due to the positive impact that the efficient planning of resources in humanitarian situations and the application of OR/MS techniques can have on alleviating the consequences of a disaster. However, several authors agree that disaster management is a topic in which the OR/MS community still has to strengthen its critical mass. Moreover, Altay and Green (2006) and Ergun et al. (2010) highlight that *disaster recovery*, i.e., the planning of actions taken during the *reconstruction phase*, is one of the main areas in which more research is needed. A recent survey by Kunz and Reiner (2012) confirms that “only ten papers specifically address the reconstruction phase” and stresses the importance of this stage by stating that “the quality of the logistical activities during this phase strongly impacts the success of the whole disaster recovery process, especially in terms of sustainability and long-term effectiveness (Beamon & Balcik, 2008; Benson, Twigg, & Myers, 2001; Besiou, Stapleton, & Van Wassenhove, 2011; Kovács & Spens, 2011).” Several authors such as Ergun et al. (2010) and Sheu (2007) point out the need for comprehensive models that: integrate multiple stages of the disaster management process as a way to account for the interaction between the decisions made at different stages; are adaptive to incorporate new information when a change occurs, due to the fact that in a disaster framework, information is usually limited at the beginning, and as time passes, more and more accurate information becomes available. Similarly, Holguín-Veras, Pérez, Jaller, Wassenhove, and Aros-Vera (2013) highlight that single-period formulations cannot account for the inter-temporal effects due to the impacts of delivery actions that accrue over subsequent time periods. Therefore, there is a need for models that involve the timing in which the recovery actions are taken when evaluating the performance of this stage of the disaster management process.

The problem we address in this paper lies on the intersection of the aforementioned areas of research. It mainly focuses on the recovery phase but also touches on the immediate response phase. Therefore, it integrates different phases of the disaster management process as it considers the emergency recovery of the road network while capturing the urgency of repairing a road to facilitate the relief distribution. The objective function of our problem accounts for the temporal impact of the repair decisions which is neglected in classical single-period models. Three different decisions are taken into account, on different managerial levels: decide which damaged nodes have to be repaired; assign the various repair activities to a repair

team in a certain order; determine the actual route or sequence of nodes to be visited by the repair crew. Most papers in the literature address the first or second group of decisions and do not take into account the additional constraints that might be needed when routing the repair crew(s), such as precedence constraints to ensure a connected path that allows to traverse the network and reach the damaged arcs. Only a handful of papers address the actual routing of the repair crews. However, these contributions rely on intricate approaches that are not easily scalable to larger problems.

Most of the papers that simultaneously tackle the three different decisions discussed here, use mathematical programming to formulate the network recovery problem. Due to the complexity of those models, they are usually tested only in few small sized instances or heuristic approaches are used to find good feasible solutions. Additionally, some of those papers take into account several objective functions which are, in general, handled through classical multi-objective methods such as weighted sum approaches and goal programming.

Chen and Tzeng (1999) consider the problem of scheduling and routing a set of work-troops in order to repair a set of damaged points scattered throughout a road-network after a large-scale earthquake. Their model consists of an upper level that schedules the work teams and a lower level that is an asymmetric traffic assignment model. Three objectives are considered, and a fuzzy genetic algorithm multi-objective approach is applied to solve the model heuristically. The approach is tested on a single small instance based on the Northridge earthquake (USA, 1994). Feng and Wang (2003) focus on the road network repair activities that occur over the first 72 hours after the disaster. The authors propose a multi-objective programming model that assigns the repair tasks to the work teams and defines the route to be followed by each team while maximizing the performance of the emergency road repair activities, maximizing the number of people that benefit from it, and minimizing the risk for rescue crews. A case study presenting the Chi-Chi earthquake is used to assess the model on a small instance involving disruptions in 6 out of 62 arcs. Yan and Shih (2009) propose a mathematical bi-objective model based on two time-space networks, one for the emergency repair and another for the relief distribution, which considers multiple commodities and minimizes the length of time required for both emergency roadway repair and relief distribution. In a later paper, Yan and Shih (2012) develop an ant colony system based algorithm to solve an emergency roadway repair time-space network flow problem, and apply the presented method to the Chi-Chi earthquake.

The papers that do not explicitly consider the routing of the repair crew usually focus on prioritizing the elements of the network that should be upgraded or repaired while considering interdependencies among related infrastructure and budget constraints. (For example, Brown and Vassiliou (1993) introduce a decision support system that relies on optimization, simulation and the decision maker input and allows for prioritization and interdependence between the tasks.) These papers often solve larger instances as some of the more complicating constraints (e.g., precedence constraints) are not needed.

Karlaftis, Kepaptsoglou, and Lambropoulos (2007) develop a methodology to optimally prioritize bridge repair in an urban transport network following a natural disaster. The resource allocation problem is formulated as a three-stage model, and a genetic algorithm is implemented to solve it. An instance considering 400 bridges in the surroundings of Athens is used to test the approach and to draw useful insights regarding the magnitude of repairs and necessary funds for repairing a bridge network following an unforeseen event. A multi-objective network optimization model is proposed by Matisziw, Murray, and Grubestic (2010) to facilitate identification and scheduling of potential recovery scenarios following disruptions involving substantial loss of network nodes and arcs. The network performance is optimized across a planning horizon restricted to the constraints on resources for recovery efforts. The developed model

is applied to support recovery planning for a telecommunication backbone network, which differs from road networks in the sense that a constraint to enforce accessibility to the damaged nodes is not required. Cavdaroglu, Hammel, Mitchell, Sharkey, and Wallace (2013) consider the problem of restoring critical interdependent infrastructure after a non-routine event causes disruptions. The authors propose a mixed-integer program that integrates three different decisions: the set of components to be installed or repaired (i.e., restoration decisions), the assignment of selected components (tasks) to available work groups, and the order in which each work group will complete the tasks assigned to it. The latter set of decisions involves a network design phase that does not explicitly considers precedence constraints. The objective function of this problem measures how well the services are being restored over the horizon of the restoration plan. A heuristic solution method is proposed and tested on a data set that is a realistic representation of the power and telecommunication systems of a large portion of Manhattan.

Maya Duque and Sorensen (2011) consider the problem of allocating scarce resources to repair a rural road network after it has been damaged by a natural or man-made disaster in order to improve the accessibility to a set of vertices in a network. The authors propose a solution approach based on the GRASP and VNS metaheuristics that aims to maximize the accessibility of as many people as possible to the main cities or regional centers where the economic and social infrastructure is usually located. In Maya Duque, Coene, Goos, Sørensen, and Spieksma (2013) the study of the problem is extended to analyze the complexity of specific cases, propose a linear integer programming formulation and two heuristic solution approaches. The problem addressed lies on the strategic and tactical level but does not involve the operational level as the actual scheduling and routing of the upgrading work is not considered. Tuzun Aksu and Ozdamar (2014) focus on the planning of road restoration efforts during disaster response and recovery. The road restoration work is scheduled with the goal of maximizing the total weighted earliness of all cleared paths as a measure of the network accessibility for all locations in the area during the restoration process. The authors propose a dynamic path based mathematical model in which a set of access paths are defined for each node in the network and the union of these paths compose the total set of pre-defined access paths to be cleared. Liberatore, Ortuño, Tirado, Vitoriano, and Scaparra (2014) develop a hierarchical optimization model balancing a combination of objective functions associated with distribution of relief supplies in a post-disaster setting where damaged roads with reduced reliability are considered. Their model selects a set of damaged links to be repaired given a fixed budget.

Stilp, Ergun, and Kekinocak (2012) study a related problem of managing debris collection and disposal operations following a disaster. While their work addresses various stages of debris management operations, the so-called *clearance problem with complete information*, focusing on clearing out the roads immediately following a disaster, is the most relevant to our problem. The authors consider a multi-period problem on a network with a given subset of supply and demand nodes. A subset of network arcs are blocked by debris and have to be cleared to connect supply to demand. Maximum budget caps how much debris (and correspondingly, how many arcs) can be cleared each time period. The objective of the clearance problem is to minimize total penalty for unmet demand incurred each period. The problem is modeled using mixed integer programming and solved using both CPLEX and a set of dedicated heuristics. Unlike Stilp et al. (2012), we do not have discrete time periods, and our model captures the actual time it takes for a repair crew to reach a damaged point and to repair it, before the crew can move on to the next task. Therefore, our solution includes the explicit route of the repair crew in the time-spatial domain.

A number of classic optimization problems, such as prize collecting traveling salesman problem (Balas, 1989), the vehicle routing

problem (e.g., Laporte, 1992; Toth and Vigo, 2002), the prize collecting Steiner tree problem (Goemans & Williamson, 1995), and the orienteering problem (Golden, Levy, & Vohra, 1987), also have similarities with our work, where damaged points in our problem correspond to “customers” to be visited to collect a “reward”. In this way, our problem can be thought of as an optimal routing problem between a subset of repair nodes. However, a number of significant distinctions between our problem and these classic optimization problems prevent the straightforward extension of existing solution approaches. First, our problem seeks to simultaneously find the repair schedule and the paths between the nodes to be repaired, since those two entities are interdependent. Furthermore, the order in which we visit the repair nodes has significant impact on the set of feasible paths between the nodes and on the “reward” collected when we visit them. Finally, the actual moment in time at which each node is repaired is also a factor in the total collected “reward”, equating it to a time-dependent setting. Also note that unlike the Steiner tree problem setting, a tree solution structure is suboptimal in most instances of our problem.

### 3. Dynamic programming model for NRCSR

First, we outline our dynamic programming (DP) model in words, then provide the mathematical formulation of the problem, and finally discuss the implementation. For a list of symbols used see Appendix A. A MIP formulation of our problem is given in Maya Duque et al. (2014, Appendix), but a direct implementation of this model in a commercial MIP solver resulted in an intractable solution method even for small problem instances.

Our DP model keeps track of the repair crew as it repairs one node in  $\mathcal{V}_r$  after another, and makes sequential decisions on which node the crew should repair next. When a node is repaired, a subset of demand nodes that have not been accessible so far can become connected to the supply node, and a “cost” is incurred corresponding to the new satisfied demand weighted by the time when the connection is established. A constraint is implicitly included requiring to connect all demand nodes to the supply node, and the weighted time the algorithm connects all the nodes in  $\mathcal{V}_d$  corresponds to the cost that is being minimized. Thus, the state of our model needs to keep track of the current time and location of the repair crew, as well as the subset of damaged nodes that have not been yet repaired, and the subset of demand nodes that are not yet accessible from the depot. Our DP formulation requires an additional auxiliary node  $n + 1$  to be added to the graph. This node corresponds to a dummy depot connected to all the damaged nodes (i.e., nodes in  $\mathcal{V}_r$ ) through arcs with zero travel time. Note that the DP model explicitly tracks the crew as it moves from one repair node to the next, while the fastest feasible paths between the repair nodes are calculated at each state transition to compute the corresponding travel time. Finally, we assume that the DP state is updated after repair of the current node is completed.

#### 3.1. Formulation

Formally, this problem can be expressed as a dynamic programming model as follows.

DP state:  $s = (i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r)$ , where

$i \in (\mathcal{V}_r \setminus \bar{\mathcal{V}}_r) \cup \{0\} \cup \{n + 1\}$	current repair node location of the crew,
$t \geq 0$	current time,
$\bar{\mathcal{V}}_d \subseteq \mathcal{V}_d$	demand nodes that are not yet accessible,
$\bar{\mathcal{V}}_r \subseteq \mathcal{V}_r$	damaged nodes that are not yet repaired.

DP action:  $a = j$ , corresponding to the decision for the repair crew to move to node  $j$ , where, given the current state  $s = (i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r)$ , the

action space  $\mathcal{A}(s)$  is

$$\mathcal{A}(s) = \begin{cases} \{j \in \bar{\mathcal{V}}_r : \tau(i, j, \bar{\mathcal{V}}_r) < \infty\}, & \text{if } \bar{\mathcal{V}}_d \neq \emptyset \\ \{n+1\}, & \text{if } \bar{\mathcal{V}}_d = \emptyset \end{cases}$$

Here,  $\tau(s, a) = \tau(i, j, \bar{\mathcal{V}}_r)$  is the function that returns minimum travel time between nodes  $i$  and  $j$  over all the paths that do not pass through nodes in  $\bar{\mathcal{V}}_r$ . The function returns infinity if no such paths exist. The fastest path problem from  $i$  to  $j$  for all  $j \in \bar{\mathcal{V}}_r$  can be solved efficiently with one execution of a Dijkstra's algorithm (Dijkstra, 1959), similar to the discussion of the accessibility function below.

*DP state transition:*  $g(s, a)$  is the state transition function that returns a state to which the system transitions when action  $a$  is chosen in state  $s$ . That is,  $s = (i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r) \xrightarrow{(a=j)} (i', t', \bar{\mathcal{V}}'_d, \bar{\mathcal{V}}'_r) = s'$ , where

$$\begin{aligned} i' &= j, \\ t' &= t + \tau(i, j, \bar{\mathcal{V}}_r) + s_j, \\ \bar{\mathcal{V}}'_d &= \bar{\mathcal{V}}_d \cup v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j), \\ \bar{\mathcal{V}}'_r &= \bar{\mathcal{V}}_r \setminus \{j\}. \end{aligned}$$

Here,  $v(s, a) = v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)$  is an *accessibility function* that returns a subset of nodes in  $\bar{\mathcal{V}}_d$  that become accessible when node  $j \in \bar{\mathcal{V}}_r$  is repaired. Note that  $v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)$  can return an empty set when no new nodes are connected to the supply node by repairing node  $j$ . A set of shortest (distance) path problems on the updated network (where node  $j$  can be passed without incurring the repair cost) needs to be solved in order to evaluate  $v(\cdot)$ . An efficient way to do so is discussed later in this section.

*DP action cost:*  $c(s, a)$  is the instantaneous cost of action  $a$  when in state  $s$ , where for  $s = (i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r)$  and  $a = j$  we have

$$c(i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j) = (t + \tau(i, j, \bar{\mathcal{V}}_r) + s_j) \cdot \sum_{k \in v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)} w_k, \quad (1)$$

*Recursive equation:* Let  $f(s)$  denote the minimum cost incurred by the system reaching from the initial state  $s_0 = (0, 0, \mathcal{V}_d, \mathcal{V}_r)$  to the current state  $s = (i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r)$ . Then, we can write the following dynamic programming (Bellman's) recursive equation:

$$f(s') = \min_{\{s, s.t., g(s,a)=s', a \in \mathcal{A}(s)\}} \{f(s) + c(s, a)\}, \quad (2)$$

where  $f(s_0) = 0$ . Recursively solving Eq. (2) we find the minimum value for

$$\min_{\{s=(i,t,\bar{\mathcal{V}}_d,\bar{\mathcal{V}}_r), s.t., i=n+1\}} f(s),$$

which corresponds to the minimum value of our problem objective function.

*Accessibility function:* Recall,  $v(s, a) = v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)$  is an *accessibility function* that returns a subset of nodes in  $\bar{\mathcal{V}}_d$  that become accessible when node  $j \in \bar{\mathcal{V}}_r$  is repaired. This function can be efficiently evaluated as follows.

First, let  $\mathcal{G}(\bar{\mathcal{V}}_r) \subseteq \mathcal{G} = (\mathcal{V}, \mathcal{E})$ , be a modified network where the nodes of  $\bar{\mathcal{V}}_r$  and the arcs adjacent to nodes in  $\bar{\mathcal{V}}_r$  are omitted from the original graph  $\mathcal{G}$ . This accounts for the fact that the demand nodes cannot be connected to the supply node through the damaged nodes that have not been yet repaired. Then, observe that a previously unaccessible demand node  $k \in \bar{\mathcal{V}}_d$  becomes accessible when node  $j \in \bar{\mathcal{V}}_r$  is repaired if and only if there is a path from  $k$  to the depot node (node 0) in network  $\mathcal{G}(\bar{\mathcal{V}}_r \setminus \{j\})$  that is not longer than  $D_k$  and passes through node  $j$ . If there was a path from  $k$  to 0 that is not longer than  $D_k$  and does not pass through node  $j$ , then that node would have been accessible prior to repairing node  $j$ , i.e., it would not have been in  $\bar{\mathcal{V}}_d$ . Thus, to evaluate the accessibility function  $v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)$ , we find the shortest distance path from each node in the set  $\bar{\mathcal{V}}_d$  to node  $j$  and then add the shortest distance from  $j$  to 0 in network  $\mathcal{G}(\bar{\mathcal{V}}_r \setminus \{j\})$ . For each node  $k \in \bar{\mathcal{V}}_d$  that has the sum of the two distances less than or equal to  $D_k$ , repairing node  $j$  would connect that node to the supply node, and the accessibility function  $v(\cdot)$  returns a set of those nodes.

Note that the algorithm has to evaluate  $v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j)$  for all  $j \in \bar{\mathcal{V}}_r$ , each of which would require to find a minimum path from  $j$  to all the nodes in  $\bar{\mathcal{V}}_d$ . We can improve the implementation efficiency by intelligently sequencing these operations. To this end, the algorithm uses Dijkstra's algorithm, which efficiently finds a minimum cost path in a static network from one node to all other nodes. At each iteration of our dynamic programming functional Eq. (2), we implement Dijkstra's algorithm on  $\mathcal{G}$  where every node in  $\bar{\mathcal{V}}_r$  is terminal. That is, as soon as the algorithm reaches one of the nodes that have not been previously repaired (i.e., in set  $\bar{\mathcal{V}}_r$ ), the paths are not allowed to continue through that node. Thus, we find a shortest path from supply node 0 to all nodes in  $\bar{\mathcal{V}}_r$ , and let  $\beta_j$  for  $j \in \bar{\mathcal{V}}_r$  denote such distance. Then, for each  $j \in \bar{\mathcal{V}}_r$  such that  $\beta_j \leq \max_{k \in \bar{\mathcal{V}}_d} D_k$  we find a shortest path in network  $\mathcal{G}(\bar{\mathcal{V}}_r \setminus \{j\})$  from  $j$  to all nodes in  $\bar{\mathcal{V}}_d$ , and let  $\delta_{jk}$  for  $k \in \bar{\mathcal{V}}_d$  denote such distance. Then,  $v(\bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j) = \{k \in \bar{\mathcal{V}}_d, s.t., \beta_j + \delta_{jk} \leq D_k\}$ . Observe, that our dynamic programming model requires solving a set of smaller dynamic programming models (shortest paths from node 0 to nodes in  $\bar{\mathcal{V}}_d$  on a partially repaired network) to evaluate the cost associated with each decision in the action space.

### 3.2. Implementation

There are a number of observations and problem structure properties we use to facilitate efficient implementation of the DP model.

#### 3.2.1. Initialization

To initialize the DP recursive equation, the algorithm preprocesses the problem instance to find a more compact initial state than  $s_0 = (0, 0, \mathcal{V}_d, \mathcal{V}_r)$ . We note that in a given instance, a number of demand nodes can maintain their accessibility despite the damage to the road network, and  $s_0$  does not need to include all the nodes in  $\mathcal{V}_d$ . Thus, a shortest path algorithm is run on the damaged network (i.e., no nodes in  $\mathcal{V}_r$  are assumed to be repaired) to find the minimum distance between the supply node 0 and all the demand nodes. Then, the distances found are compared to the values of  $D_i$ 's, and nodes in  $\mathcal{V}_d$  that are already accessible are omitted from the initial state  $s_0$ . In addition, the demand nodes with  $w_i = 0$  are omitted, since these nodes correspond to transshipment (intersection) nodes and, regardless of their accessibility status, do not contribute to the value of the objective function.

#### 3.2.2. Cost function reformulation

The cost function defined in Eq. (1) assigns a positive value to an action corresponding to repairing a damaged node that connects new demand nodes to the supply and a zero value when a repair action does not establish accessibility for any demand nodes. When a network is severely damaged requiring to repair multiple nodes to improve accessibility, our DP state network has a significant number of action arcs correspond to zero cost, and only a small subset of actions has value larger than zero. Implementation of minimum cost algorithms on such networks is inefficient since a lot of actions and nodes have identical values (i.e., equal contribution to the objective function). To facilitate more efficient implementation of the DP model, we redefine the cost function as follows,

$$c'(i, t, \bar{\mathcal{V}}_d, \bar{\mathcal{V}}_r, j) = (\tau(i, j, \bar{\mathcal{V}}_r) + s_j) \cdot \sum_{k \in \bar{\mathcal{V}}_d} w_k. \quad (3)$$

Thus, Eq. (3) assigns a cost of going from node  $i$  to node  $j$  equal to the penalty accrued during that move (plus repair time at node  $j$ ) for all the demand nodes that have not been accessible at that time. Then, the original recursive Eq. (2) is solved while substituting cost function  $c(\cdot)$  with the new cost function  $c'(\cdot)$ . Observe that the modified formulation results in exactly the same objective function value, the only difference being how we sum the cost: all at once at the time of connecting a demand node to the supply (1), or progressively as crew moves along the path (3).

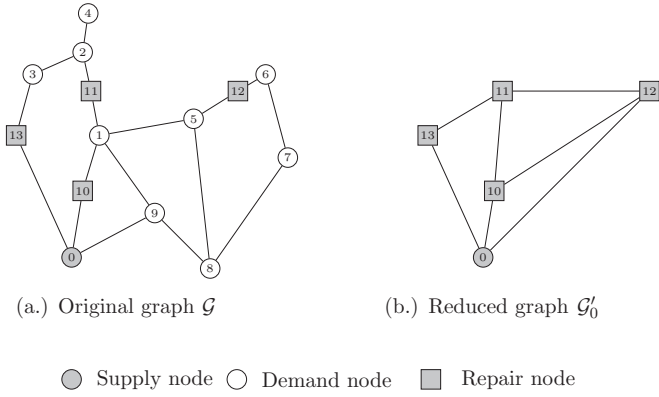


Fig. 2. Graph reduction process.

#### 4. An iterated greedy-randomized constructive procedure for the NRCSR

Due to the limitation on the size of problems that can be solved by our DP model, we present an alternative approach to tackle the NRCSR: an iterated greedy-randomized constructive procedure. IGRCP is motivated by a GRASP (greedy randomized adaptive search procedure), a constructive metaheuristic introduced by Feo and Resende (1995) that generally consists of two phases: construction of a feasible solution and improvement of the constructed solution. Both phases are usually iterated several times, after which the best solution is reported. The construction phase uses a controlled amount of randomness in order to overcome the myopic behavior of a greedy constructive heuristic. The improvement phase uses, in most cases, a local search strategy that iteratively improves the initial solution by making small changes to it. IGRCP also consists of the construction and improvement phases, but unlike GRASP, we do not iterate between them, since our construction phase does not depend on the solution found in previous iterations. Instead, we implement multiple runs of the construction plus improvement routine. Due to the random nature of the algorithm, this multistart approach might return different solutions, which we compare and analyze later in the paper.

In Section 4.1, we present a set of basic definitions for our IGRCP solution approach. The construction phase is described in Section 4.2, while the improvement phase is presented in Section 4.3. Finally, the technical details related to the implementation are discussed in Section 4.4.

##### 4.1. Heuristic notation and definitions

We consider the undirected and connected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  that defines the NRCSR, introduced in Section 1, and refer to it as the *original graph*. Additionally, we define a *reduced graph*  $\mathcal{G}'_0 = (\mathcal{V}'_0, \mathcal{E}'_0)$ , in which only the repair nodes and the depot node appear. In this reduced graph the set of vertices is  $\mathcal{V}'_0 = \mathcal{V}_r \cup \{0\}$ . The set of edges  $\mathcal{E}'_0$  consists of those node pairs  $(i, j) \in \mathcal{E}'_0$  between which a path in  $\mathcal{G}$  exists that does not contain any other nodes in  $\mathcal{V}'_0$  (i.e., that does not pass through nodes in  $\mathcal{V}_r$  that have not been repaired). The cost of edge  $(i, j) \in \mathcal{E}'_0$  is equal to the minimum travel time between nodes  $i$  and  $j$  in  $\mathcal{G}$ . Note that, for a given pair of nodes  $i$  and  $j$ , such a path might not exist. If that is the case, no edge is present in the reduced graph  $\mathcal{G}'_0$ . Fig. 2 presents an example of an original graph and the associated reduced graph.

Let  $\mathbf{x} = [x_1, x_2, \dots, x_k]$  where  $k \leq |\mathcal{V}_r|$  and  $x_n \in \mathcal{V}_r$  for  $n = 1, \dots, k$  be a candidate solution of the NRCSR. Such a candidate solution is a permutation of a subset of the damaged nodes, and corresponds to the sequence to be followed by the repair crew, where  $x_n$  is the  $n$ th repair node on the crew's schedule (for  $n = 1, \dots, k$ ). Note, however,

that not all permutations might be feasible due to existing precedence relationships. These precedence relationships are due to the fact that there might be some damaged nodes that can be reached only after another damaged node has been repaired. Therefore, we define a solution  $\mathbf{x}$  to be *feasible* if the sequence that it represents makes all the demand nodes accessible to the depot and all precedence relationships hold.

For a given permutation  $\mathbf{x} = [x_1, \dots, x_k]$  where  $k \leq |\mathcal{V}_r|$ ,  $\mathcal{G}'_n = (\mathcal{V}'_n, \mathcal{E}'_n)$  for  $1 \leq n \leq k$  denotes the updated reduced graph after nodes  $x_1, \dots, x_n$  have been repaired and the repair crew is located at node  $x_n$ . Then,  $\mathcal{V}'_n = \mathcal{V}_r \setminus \{x_1, \dots, x_{n-1}\}$  and  $\mathcal{E}'_n$  is the set of edges such that  $(i, j) \in \mathcal{E}'_n$  for  $i, j \in \mathcal{V}'_n$  if there exists a path in  $\mathcal{G}$  between nodes  $i$  and  $j$  that does not contain any other nodes in  $\mathcal{V}'_n$ .

Then, for a given  $\mathbf{x}$  we define,

$t(x_n)$	Time at which node $x_n$ is repaired;
$\tilde{\mathcal{V}}_d^n \subseteq \mathcal{V}_d$	Demand nodes that remain inaccessible after repairing nodes $x_1, \dots, x_n$ in sequence $\mathbf{x}$ ;
$\tilde{\mathcal{V}}_r^n \subseteq \mathcal{V}_r$	Damaged nodes that have not been repaired after repairing nodes $x_1, \dots, x_n$ in sequence $\mathbf{x}$ ;
$\mathcal{G}'_n = (\mathcal{V}'_n, \mathcal{E}'_n)$	Reduced graph after repairing nodes $x_1, \dots, x_n$ in sequence $\mathbf{x}$ ;
$\delta(x_n, \mathcal{G}'_n)$	Set of nodes adjacent to $x_n$ in the graph $\mathcal{G}'_n$ , that is $\delta(x_n, \mathcal{G}'_n) := \{i \in \tilde{\mathcal{V}}_r^n : (x_n, i) \in \mathcal{E}'_n\}$ .

Similarly to the description of our DP model, we define an *accessibility function*  $v(\mathbf{x}, i)$ , for  $\mathbf{x} = [x_1, \dots, x_k]$  and  $i \in \tilde{\mathcal{V}}_d^k$ . Function  $v(\mathbf{x}, i)$  returns the subset of nodes from  $\tilde{\mathcal{V}}_d^k$  that become accessible when node  $i$  is repaired after first having repaired nodes  $x_1, \dots, x_k$  (that is, when the damaged node  $i$  is added in the position  $k + 1$  to the repair sequence). In Section 4.4.2 we present an efficient way to evaluate the accessibility function  $v(\mathbf{x}, i)$  in this context.

Finally, solution  $\mathbf{x} = [x_1, \dots, x_k]$  has an associated objective value  $f(\mathbf{x})$  that is equal to the demand-weighted sum of the time when each demand node in  $\mathcal{V}_d$  becomes accessible. Thus, the objective value  $f(\mathbf{x})$  of a solution  $\mathbf{x} = [x_1, \dots, x_k]$  is

$$f(\mathbf{x}) = \sum_{n=1}^k \left\{ t(x_n) \cdot \sum_{\substack{i \in v(\mathbf{x}, x_n): \\ \mathbf{x}' = [x_1, \dots, x_{n-1}, i]}} w_i \right\}. \tag{4}$$

##### 4.2. Construction phase

The construction phase of our IGRCP algorithm aims to generate an initial feasible solution for the NRCSR. Although some randomness is needed to generate a diverse set of initial solutions during different runs of the algorithm, a completely random procedure (e.g., a random permutation of the damaged nodes) is generally not advised. This is due to the fact that such procedure might either fail to generate a feasible solution or only generate low quality initial solutions. Therefore, we propose a procedure that generates feasible solutions for the NRCSR taking into account the pre-disaster network conditions.

The repair sequence is initialized at time 0 with the depot node (i.e.,  $n = 0, x_0 = 0, \mathbf{x} = \emptyset$ ) and the following steps are iterated until either all damaged nodes are repaired or all demand nodes have become accessible, whichever comes first.

- Step 1. Choose the next node to be repaired ( $x_{n+1}$ ) from the set of nodes adjacent to  $x_n$  ( $\delta(x_n, \mathcal{G}'_n)$ ). (Section 4.2.1 describes the procedure that is used to select node  $x_{n+1}$ .)
- Step 2. Update the repair time for node  $x_{n+1}$ :

$$t(x_{n+1}) = t(x_n) + t'_{x_n x_{n+1}} + s_{x_{n+1}}, \tag{5}$$

where  $t'_{x_n x_{n+1}}$  is the travel time between damaged nodes  $x_n$  and  $x_{n+1}$  in the reduced graph  $\mathcal{G}'_n$  (i.e., the time to traverse

**Algorithm 1** Construction Phase

---

```

1: Initialization:  $n = 0, x_n = 0, \mathcal{G}'_0 = (\mathcal{V}'_0, \mathcal{E}'_0), \mathbf{x} = \emptyset$ 
2: while  $\tilde{\mathcal{V}}_d^n \neq \emptyset$  and  $\tilde{\mathcal{V}}_r^n \neq \emptyset$  do
3:   Select a node  $x_{n+1}$  from  $\delta(x_n, \mathcal{G}'_n)$  to be repaired
4:   Update  $t(x_{n+1})$ 
5:   Evaluate the accessibility function  $v(\mathbf{x}, x_{n+1})$ 
6:   Update the sets  $\tilde{\mathcal{V}}_d^{n+1}$  and  $\tilde{\mathcal{V}}_r^{n+1}$ 
7:   Update the objective function  $f(\mathbf{x}, x_{n+1})$ 
8:   Update the repair sequence  $\mathbf{x} \leftarrow [\mathbf{x}, x_{n+1}]$ 
9:   Set  $n \leftarrow n + 1$ 
10:  Update the reduced graph  $\mathcal{G}'_n = (\mathcal{V}'_n, \mathcal{E}'_n)$  and set  $\delta(x_n, \mathcal{G}'_n)$ 
11: end while

```

---

the edge that connects nodes  $x_n$  and  $x_{n+1}$  in the reduced graph  $\mathcal{G}'_n$ .

Step 3. Evaluate the accessibility function  $v(\mathbf{x}, x_{n+1})$  where  $\mathbf{x} = [x_1, \dots, x_n]$ . (Section 4.4.2 describes this procedure.) Use the set of nodes returned to update  $\tilde{\mathcal{V}}_d^{n+1}$  and  $\tilde{\mathcal{V}}_r^{n+1}$ , as follows,

$$\tilde{\mathcal{V}}_d^{n+1} = \tilde{\mathcal{V}}_d^n \setminus v(\mathbf{x}, x_{n+1}), \quad (6)$$

$$\tilde{\mathcal{V}}_r^{n+1} = \tilde{\mathcal{V}}_r^n \setminus \{x_{n+1}\}. \quad (7)$$

Update the value of the objective function,

$$f(x_1, \dots, x_{n+1}) = f(x_1, \dots, x_n) + b(\mathbf{x}, x_{n+1}), \quad (8)$$

where  $b(\mathbf{x}, x_{n+1})$  is the marginal contribution to the objective function value of repairing node  $x_{n+1}$  after having repaired the nodes  $\mathbf{x} = [x_1, \dots, x_n]$ . That is,

$$b(\mathbf{x}, x_{n+1}) = t(x_{n+1}) \cdot \sum_{i \in v(\mathbf{x}, x_{n+1})} w_i. \quad (9)$$

Step 4. Update the new repair sequence  $\mathbf{x} \leftarrow [\mathbf{x}, x_{n+1}]$ . Set  $n \leftarrow n + 1$ , update the reduced graph  $\mathcal{G}'_n = (\mathcal{V}'_n, \mathcal{E}'_n)$  and identify the set of adjacent nodes  $\delta(x_n, \mathcal{G}'_n)$ .

In Algorithm 1, we present a schematic overview of this procedure.

#### 4.2.1. Selection of the next node to be repaired

Algorithm 1 (Step 1) requires that, at each iteration and for a given current candidate solution  $\mathbf{x} = [x_1, \dots, x_n]$ , we choose the next node to be repaired ( $x_{n+1}$ ) from the list of adjacent nodes  $\delta(x_n, \mathcal{G}'_n)$ . Depending on how this selection is performed, different solutions might be obtained. We propose a selection procedure that encompasses two basic principles: (1) it uses information about the network before the disaster (i.e., in which all nodes in  $\mathcal{V}_r$  are non-damaged); and (2) it includes randomness such that different solutions might be generated every time the constructive phase is executed. This procedure is carried out as follows.

Given the original network  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and assuming that all damaged nodes are repaired (corresponding to the network before the disaster), we solve a minimum cost flow problem in which the cost corresponds to the traveled distance; each demand node  $i$  has a demand  $w_i$ ; each damaged node has a demand equal to zero; and the depot has a supply capacity equal to  $\sum_{i \in \mathcal{V}_d} w_i$ . Based on the solution of this minimum cost flow problem, parameter  $u_i$  is computed for each damaged node. The value of  $u_i$  is equal to the amount of flow that passes through node  $i$  and reflects the importance of node  $i$  in the pre-disaster conditions.

Next, given a candidate solution  $\mathbf{x} = [x_1, \dots, x_n]$ , the next node to be repaired ( $x_{n+1}$ ) has to be selected from the set of nodes adjacent to  $x_n$  ( $\delta(x_n, \mathcal{G}'_n)$ ). To accomplish this, for each node  $k \in \delta(x_n, \mathcal{G}'_n)$  we compute a probability  $p_k$  of selecting that node to be the next node  $x_{n+1}$ . Probability  $p_k$  is computed using Eq. (10). The first component of

Eq. (10) represents the importance of node  $k$  relative to all the nodes in  $\delta(x_n, \mathcal{G}'_n)$ . This component is multiplied by  $\lambda \in [0, 1]$  that weighs the importance of the pre-disaster conditions. The second component of Eq. (10) distributes the remaining (i.e.,  $(1 - \lambda)$ -weighted) probability of selecting node  $k$  equally among all the nodes in  $\delta(x_n, \mathcal{G}'_n)$ .

$$p_k = \lambda \frac{u_k}{\sum_{j \in \delta(x_n, \mathcal{G}'_n)} u_j} + (1 - \lambda) \frac{1}{|\delta(x_n, \mathcal{G}'_n)|}. \quad (10)$$

Using the presented procedure to generate the initial solution guides our algorithm to solutions that resemble the network before the disaster while simultaneously introducing diversity to the set of initial solutions. This random diversity of the initial solutions facilitates better exploration of the solution space.

#### 4.3. Improvement phase

The improvement phase of our IGRCP algorithm uses a local search algorithm to improve the solution generated during the construction phase. The local search improves this solution by iteratively making small changes, called *moves* to the current solution. The set of solutions that can be reached from the current solution by applying a single move, is called the *neighborhood* of that solution. In each iteration, a solution from the neighborhood of the current solution is selected to become the new current solution according to a pre-specified *move strategy*. The local search stops when all solutions in the neighborhood of the current solution have a worse objective function value, or when a given maximum number of iterations,  $\theta$ , is reached.

The first step of the local search algorithm is to clean up the solution generated during the construction phase,  $\mathbf{x} = [x_1, \dots, x_k]$  for  $k \leq |\mathcal{V}_r|$ . More specifically, the algorithm removes all the repair nodes in the sequence that do not improve connectivity nor accessibility of any other node. To do this, for each node  $x_i \in \mathbf{x}$ ,  $\tilde{\mathcal{V}}_d^{i-1}$  is compared to  $\tilde{\mathcal{V}}_d^i$ . If both sets are the same, repairing node  $x_i$  has not improved the accessibility of the network. Equivalently, the algorithm evaluates the accessibility function discussed in Section 4.4.2 to check whether  $v(\mathbf{x}', x_i) \neq \emptyset$ , for  $\mathbf{x}' = [x_1, \dots, x_{i-1}]$ . It also compares sets of repair nodes that can be reached before and after repairing node  $x_i$ , i.e., it compares  $\delta(0, \mathcal{G}'_{i-1})$  and  $\delta(0, \mathcal{G}'_i)$ . If  $x_i$  itself is the only difference between those two sets, the *connectivity* of the network has not been improved. Thus, we define a Boolean function  $h(x_i)$  for all  $x_i \in \mathbf{x}$  that is equal to 1 if repairing node  $x_i$  after nodes  $x_1, \dots, x_{i-1}$  in  $\mathbf{x}$  have been already repaired improves at least one of either *connectivity* or *accessibility* of the network. Otherwise,  $h(x_i)$  is set to 0. The cleanup step of the improvement phase evaluates  $h(x_i)$  for all  $x_i \in \mathbf{x}$  and removes all  $x_i$  for which  $h(x_i) = 0$  from the repair sequence  $\mathbf{x}$ . We denote the new solution by  $\mathbf{x}_0 = [x_1, \dots, x_{k_0}]$ . Note that the new solution  $\mathbf{x}_0$  corresponds to a feasible solution with an objective function value that is guaranteed to be no worse than the initial solution  $\mathbf{x}$ .

After the initial solution has been cleaned up, the second stage of the improvement phase is executed, which reorders the repair nodes in  $\mathbf{x}_0$  in order to improve the solution. This is carried out by iteratively relocating elements in  $\mathbf{x}_0$  one node at a time. To perform a *relocation move* three distinct steps are executed:

Step 1. **Select node  $x_i \in \mathbf{x}_0$  to be relocated.** The algorithm chooses node  $x_i$  that has the largest marginal contribution to the objective function, i.e.,

$$i = \arg \max_l \{b(x_l) : l \in (1, \dots, k_0)\}. \quad (11)$$

Step 2. **Choose the position  $j$  to which node  $x_i$  will be relocated.** Starting with  $j = 1$ , the algorithm checks whether inserting  $x_i$  into the  $j$ th position is feasible and decreases the objective value  $f(\cdot)$ . The algorithm iteratively increases  $j$  by one until it finds such a position or reaches the current position of node  $x_i$ , i.e.,  $j = i$ . (See Section 4.4.1 for details on how the feasibility check is performed.)

**Algorithm 2** Improvement Phase

```

1: Pre-processing: clean up solution deleting nodes with  $h(x_i) = 0$ 
2: repeat
3:   Sort  $\mathcal{L}(\mathbf{x}_0)$  in descending order of  $b(x_i)$ 
4:   for  $i = 1 \dots k_0$  do
5:     for  $j = 1 \dots i$  do
6:        $\hat{\mathbf{x}} \leftarrow r(\mathbf{x}_0, i, j)$ 
7:       if  $\hat{\mathbf{x}}$  is feasible AND  $f(\hat{\mathbf{x}}) < f(\mathbf{x}_0)$  then
8:          $\mathbf{x}_0 \leftarrow \hat{\mathbf{x}}$ 
9:         Return to line 3
10:      end if
11:    end for
12:  end for
13: until Stopping criterion is met
    
```

If no such  $j$  is found, the algorithm goes back to Step 1 and selects the  $x_i$  with the next largest value of  $b(x_i)$ .

**Step 3. Update the new solution.** Let the relocation function  $r(\mathbf{x}_0, i, j)$  return the solution resulting from relocating  $x_i \in \mathbf{x}_0$  into the  $j$ th position and update  $b(x_i)$  values for  $i \in (1, \dots, k_0)$  for the new repair sequence. That is

$$r(\mathbf{x}_0, i, j) = [x_1, \dots, x_{j-1}, x_i, x_j, \dots, x_{i-1}, x_{i+1}, \dots, x_{k_0}]. \quad (12)$$

The intuition behind *Step 1* is to select the damaged node from the repair sequence that has the largest impact on the objective function (possibly, due to its connectivity to a large number of demand nodes or to the demand nodes with high demand values) and attempt to decrease its marginal value by inserting it as early as possible into the repair sequence. Note that another variation of the presented meta-heuristic is to implement the cleanup step on the updated solution after each reordering iteration, but our numerical tests demonstrate no benefit of doing so.

In order to efficiently explore the neighborhood of the current solution  $\mathbf{x}_0 = [x_1, \dots, x_{k_0}]$  we form a list  $\mathcal{L}(\mathbf{x}_0)$  containing the nodes to be repaired,  $(x_1, \dots, x_{k_0})$ . This list is sorted in decreasing order with respect to the values  $b(x_i)$  associated with each damaged node  $x_i \in \mathbf{x}_0$ . The sorted list is explored from the beginning until we find the first node to be relocated that improves the solution. When a better solution is found, we update the solution and resort the list. **Algorithm 2** presents a schematic overview of the improvement phase.

**4.4. Implementation**

Here, we discuss the nuances of implementation of the presented IGRCP algorithm. First, in **Section 4.4.1**, we describe the procedure to determine whether a solution obtained by a relocation move is feasible. Then, in **Section 4.4.2**, we present the procedure to evaluate the accessibility function  $v(\mathbf{x}, i)$ .

**4.4.1. Feasibility check**

Each iteration of the improvement phase, described in **Section 4.3**, requires the algorithm to explore the neighborhood of the current solution  $\mathbf{x}_0$ . To this end, for each neighboring solution  $\hat{\mathbf{x}}$  obtained by relocating a repair node  $x_i \in \mathbf{x}_0$ , the algorithm first evaluates whether  $\hat{\mathbf{x}}$  is feasible, and then whether it improves the current solution  $\mathbf{x}_0$ . In case a node  $x_i$  is relocated to position  $j$  that precedes its current position (i.e.,  $j < i$ ), the solution  $\hat{\mathbf{x}}$  obtained is feasible if node  $x_i \in \delta(x_{j-1}, \mathcal{G}_{j-1}^i)$ , where node  $x_{j-1}$  is the node in the position  $j - 1$  of the repair sequence.

**4.4.2. Accessibility function Evaluation**

Each time a node  $i$  is repaired and added to the repair sequence in solution  $\mathbf{x}$ , the accessibility function  $v(\mathbf{x}', x_n)$ , for  $x_n = i$  and  $\mathbf{x}' = [x_1, \dots, x_{n-1}]$ , has to be evaluated in order to determine the demand

nodes that become accessible. This function requires to compute, for each demand node  $j \in \bar{\mathcal{V}}_d^n$  (i.e., each demand node  $j$  that is not yet accessible), the shortest path from  $j$  to the depot and to compare the length of that shortest path with the maximum acceptable distance  $D_j$ .

To improve the efficiency of the algorithm, we formulate the problem of finding the shortest path from the depot to all the demand nodes as a minimum cost flow problem (Maya Duque et al., 2013).

$$\min \sum_{(i,j) \in \mathcal{E}} l_{ij} y_{ij} \quad (13)$$

s.t.

$$\sum_{j:(i,j) \in \mathcal{E}} y_{ij} - \sum_{j:(j,i) \in \mathcal{E}} y_{ij} = d_i \quad \forall i \in \mathcal{V} \quad (14)$$

$$y_{ij} \geq 0 \quad \forall (i, j) \in \mathcal{E} \quad (15)$$

In this model, for each edge  $e_{ij}$  connecting nodes  $i$  and  $j$ , two decision variables are defined:  $y_{ij}$  corresponding to the edge being traversed from  $i$  to  $j$ , and  $y_{ji}$  corresponding to the traversal from  $j$  to  $i$ . Each demand node  $i$  has a demand  $d_i = w_i$ ; each damaged node has a zero demand; and the depot has a supply capacity  $d_i$  equal to  $\sum_{i \in \mathcal{V}_d} w_i$ . The parameter  $l_{ij}$  depends on the length of edge  $e_{ij}$  and the status of the nodes that it connects, i.e., whether  $i$  and  $j$  are damaged or repaired. Thus, when evaluating the accessibility function after repairing node  $x_n$ , the value of the parameter  $l_{ij}$  would depend on the set of damaged nodes that remain unrepaired,  $\bar{\mathcal{V}}_d^n$ . If, at least, either  $i$  or  $j \in \bar{\mathcal{V}}_d^n$ ,  $l_{ij} = l_{ji} = d_{ij} + M$ , otherwise  $l_{ij} = l_{ji} = d_{ij}$ , where  $M$  is a large constant value.

Based on the optimal value of decision variables  $y_{ij}$  of the minimum cost flow problem, the shortest path from the depot to a given node  $j$  can be obtained by tracking back the edges that are used to pass the flow. Thus, the shortest path from the depot to  $j$  is equal to the sum of the lengths of the edges that carry the flow between these two nodes. For each node  $j$  that was not accessible after having repaired nodes  $x_1, \dots, x_{n-1}$  (i.e., for all  $j \in \bar{\mathcal{V}}_d^{n-1}$ ), the shortest paths obtained from the solution of (13)–(15) are compared to the maximum acceptable distances  $D_j$ 's. The function  $v(\mathbf{x}', x_n)$ , for  $\mathbf{x}' = [x_1, \dots, x_{n-1}]$ , returns a subset of nodes  $j \in \bar{\mathcal{V}}_d^{n-1}$  for which the obtained shortest paths are not greater than  $D_j$ .

Note that, since at each iteration only one additional damaged node is repaired, only a few of the coefficients  $l_{ij}$  have to be updated from one iteration to the next. Therefore, the minimum cost flow problem can be efficiently reoptimized at each iteration.

**5. Computational experiments**

In this section, we evaluate the two solution approaches we propose for the NRCSR. To this end, we use a set of randomly generated instances of the problem which are available from <http://antor.ua.ac.be/downloads/NRCSR>. The dynamic programming algorithm was coded in MATLAB while the IGRCP algorithm was implemented in Java and CPLEX using Concert Technology (IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.2).

**5.1. Instance generation**

To generate the instances necessary for the experiments, we use the network generator GNETGEN, which is a modification of the widely used NETGEN generator proposed by Klingman, Napier, and Stutz (1974). The generator creates a minimum cost flow network for a given number of nodes  $n = |\mathcal{V}|$  and edges  $|\mathcal{E}|$ . In the generated network, each node has associated a supply/demand value, being the supply equal to the total demand, and each edge has a cost for being traversed. In our case, this cost represents the travel distance  $d_{ij}$



**Table 1**  
Sets of instances generated for the computational experiments.

	$S_1$	$S_2$
Number of nodes	21, 26, 31, 36, 41	61, 81, 101, 201, 301, 401
Instances per network size	3	3
Values of $\alpha$ (percent)	5, 10, 25, 30, 50	5, 10, 25, 30, 50
Values of $\beta$ (percent)	5, 10, 25, 50	5, 10, 25, 50
Total instances	300	360

between the two nodes  $i$  and  $j$  connected by edge  $e_{ij}$ , and it is a variable within the interval  $[0, 10]$ . This network is transformed into a network repair instance using the following procedure.

For each edge  $e_{ij}$ , we generate the parameter  $t_{ij}$  that represents the travel time. This parameter is defined as a function of  $d_{ij}$  and an average travel speed  $v$  as shown in Eq. (16), where  $r$  is a uniformly distributed value in the interval  $[0, 1]$ .

$$t_{ij} = (1 + r) \frac{d_{ij}}{v} \tag{16}$$

Next, a number of damaged nodes  $n_d = |\mathcal{V}_d|$  is determined. We use a parameter  $\alpha$  that specifies the percentage of damage of the network, i.e., the percentage of the edges in the network that are damaged by the disaster. Thus, the number of damaged edges is  $n_d = \lceil \alpha |\mathcal{E}| \rceil$ .

We randomly select  $n_d$  edges. For each selected edge  $e$ : (1) an intermediate point on the edge is randomly chosen, and (2) a damaged node  $j$  is created corresponding to that intermediate point. The repair time  $s_j$  for the damaged node is set as a random variable uniformly distributed in the interval  $[10, 60]$ . (3) The edge  $e$  is replaced by two new edges, each of them connecting one of the extremes of edge  $e$  and node  $j$ . (4) The distance and travel time for each of the new edges depend on the proportion that it represents of the original edge  $e$ .

Finally, for each node  $i$ , we must generate the maximum acceptable distance  $D_i$  to the depot. That is, the maximum acceptable length of an undamaged or repaired path from  $i$  to the depot. The parameter  $D_i$  is defined as a function of  $SP_i$ , the shortest path from  $i$  to the depot on the network in which all damaged nodes are repaired (i.e., the pre-disaster conditions). In order to compute  $D_i$ , we use a parameter  $\beta$  that represents the maximum tolerable percentage by which the path connecting  $i$  to the depot can increase. Thus, for each node  $i$  the maximum acceptable distance  $D_i$  is defined as  $(1 + \beta)SP_i$ .

Two sets of instances are considered for the computational experiments. First, we consider a set  $S_1$  of small instances, a majority of which can be solved exactly by our dynamic programming model. We use these instances to analyze the performance of the DP approach, evaluate the components of the IGRCP algorithm, and compare the two solution methods. Second, we analyze a set  $S_2$ , containing medium and large size instances, that is used to test the performance of the IGRCP algorithm. Note that using each one of the minimum cost flow networks generated by GNETGEN, several instances can be created by using different combinations of the parameters  $\alpha$  and  $\beta$ . Table 1 shows the number of nodes considered in each instance, the number of instances generated for each number of nodes, the values used for parameters  $\alpha$  and  $\beta$ , and the total number of instances in each set.

### 5.2. Dynamic programming

The performance of our dynamic programming model is evaluated using set  $S_1$ , which consists of 300 small instances. Limiting the maximum computing time to 24 hours for each instance, the algorithm found optimal solutions for 225 of those instances. Table 2 presents the number of instances solved to optimality for each combination of number of nodes in the network and value of the parameter  $\alpha$ , while Table 3 shows the number of optimal solutions found for each combination of number of nodes and value of the parameter  $\beta$ . The number

**Table 2**  
Number of optimal solutions found by the DP for each combination of number of nodes and values of  $\alpha$ , out of 12 ( $S_1$  set of instances).

No. nodes	$\alpha$				
	5	10	25	30	50
21	12	12	12	12	9
26	12	12	12	12	12
31	12	12	12	11	1
36	12	12	5	3	0
41	12	12	2	2	0

**Table 3**  
Number of optimal solutions found by the DP for each combination of number of nodes and values of  $\beta$ , out of 15 ( $S_1$  set of instances).

No. nodes	$\beta$			
	5	10	25	50
21	14	14	14	15
26	15	15	15	15
31	11	12	12	13
36	6	7	9	10
41	6	6	8	8

**Table 4**  
CPU time (in seconds) with respect to the instance size and values of  $\alpha$  ( $S_1$  set of instances).

No. nodes	$\alpha$				
	5	10	25	30	50
21	0.03	0.03	1.73	34.88	10,243.21
26	0.02	0.03	1.26	15.94	19,251.06
31	0.03	0.12	1657.24	14,542.48	232.74
36	0.06	0.88	7642.19	205.56	-
41	0.19	23.85	5640.91	24,334.51	-

of optimal solutions found decreases when the number of nodes and the level of damage of the network ( $\alpha$ ) increase. On the other hand, the number of instances solved to optimality slightly increases when  $\beta$ , the maximum tolerable percentage in which the path connecting  $i$  to the depot can augment, increases.

Table 4 presents the average computing time for each combination of number of nodes and value of parameter  $\alpha$  that were solved within 24 hours. It increases when both, the value of  $\alpha$  and the number of nodes, increase. However, the effect of the parameter  $\alpha$  is considerably stronger. The average computing time for the instances with 31 nodes and  $\alpha = 25$  percent is about 4 hours. From this point on, the increasing pattern of the computing times changes due to the fact that the number of instances that DP is able to solve to optimality within the 24 hours decreases.

### 5.3. IGRCP algorithm

We first use a subset of the small instances in  $S_1$  and the corresponding optimal solutions found by dynamic programming model to evaluate the significance of the different parameters of the IGRCP algorithm. Then, using the optimal parameter setting, we evaluate the effectiveness of our IGRCP algorithm to generate good feasible solutions and discuss its general performance in terms of computing time and the effect of the different components and parameters of the algorithm when solving the larger instances ( $S_2$ ).

#### 5.3.1. Parameter tuning

The IGRCP algorithm we propose in this paper has three parameters that can be tuned: the importance granted to the pre-disaster conditions when generating the initial solution ( $\lambda$ ), the maximum

**Table 5**

Average (and standard deviation) CPU time (in seconds) with respect to the instance size and values of  $\alpha$  ( $S_1$  set of instances).

No. nodes	$\alpha$				
	5	10	25	30	50
21	0.5 (0.1)	0.6 (0.1)	1.3 (0.4)	2.0 (0.8)	7.2 (4.7)
26	0.6 (0.1)	0.7 (0.1)	1.5 (0.6)	2.2 (1.2)	7.4 (4.4)
31	0.7 (0.1)	0.9 (0.1)	2.6 (1.1)	5.0 (4.3)	77.4 (55.3)
36	0.8 (0.1)	1.2 (0.2)	9.8 (4.6)	19.6 (12.3)	282.7 (191.4)
41	1.1 (0.1)	1.8 (0.7)	47.9 (41.2)	95.1 (83.2)	1454.7 (1137.8)

number of relocation moves that can be performed in the local search ( $\theta$ ), and the number of times that the two phases of the algorithm are iterated ( $\tau$ ) (i.e., the number of feasible solutions that are generated initially).

In a statistical experiment using a subset of the instances in  $S_1$  we determine the significance of the effect of the three parameters on the quality of the solutions and computing time. To this end, we use three different values for the parameters  $\lambda$  (25 percent, 50 percent and 75 percent) and  $\theta$  (10, 50 and 100), and two values for the parameter  $\tau$  (1 and 5). The analysis of variance (ANOVA) indicates that the parameters  $\theta$  and  $\tau$  have significant effect on the quality of the solution and computing time. Thus, when  $\theta$  and  $\tau$  increase, the value of the objective function decreases while the computing time increases. This is an expected behavior since the increasing values of  $\theta$  and  $\tau$  correspond to the increased number of solutions considered by our heuristic. However, we observe only marginal improvement in the quality of the solutions found as this parameter increases beyond the values of  $\theta = 100$  and  $\tau = 5$  (with significant increase in run time), and therefore they are not considered in our study. Given the good performance of our heuristic solution method, we do not believe that the increased run time is worth the corresponding improvement. At the same time, parameter  $\lambda$  does not have significant effect on the IGRCP performance. However, the computing times for the most difficult instances decrease when the value of  $\lambda$  increases. Based on the results of the designed experiment, the parameters  $\lambda$ ,  $\theta$  and  $\tau$  are set to 50 percent, 100 and 5, respectively.

5.3.2. Numerical results for the set of small instances,  $S_1$

To evaluate the performance of our IGRCP algorithm, we solve the 300 small instances (set  $S_1$ ) and compare the solutions found by IGRCP with those obtained using the DP approach. Due to the random nature of our algorithm, we perform 30 independent runs of IGRCP with random start over the set of instances in  $S_1$ ; we call each of these runs a *repetition*. The IGRCP performance is assessed based on three metrics: (1) algorithm computation run time, (2) the quality of the found solutions, and (3) consistency of the algorithm performance among multiple repetitions.

The average computing time for IGRCP algorithm limited to the 225 (out of 300) instances that were solved to optimality by DP model was less than 2.34 seconds. When the analysis is extended to the complete set of 300 instances the average computing time increases considerably. In addition, the computing time increases for larger values of  $\alpha$  and smaller values of  $\beta$ , i.e., when the network is severely affected by the disaster and the maximum tolerable increase in the shortest path from a node  $i$  to the depot to be considered accessible is small. Note, however, that the effect of the parameter  $\alpha$  is significantly larger than the effect of the parameter  $\beta$ . Tables 5 and 6 present the average and standard deviation for computing times for each combination of the number of nodes and values of the parameters  $\alpha$  and  $\beta$ , respectively, aggregated over 30 repetitions.

To evaluate the quality of the solutions found by IGRCP we compare its performance to the optimal solutions we obtain with the DP algorithm for 225 out of 300 instances. (The 30 repetitions for each instance results in a total of 6750 IGRCP solutions available for analysis.) Our algorithm finds the optimal solutions for 6264 out of 6750

**Table 6**

Average (and standard deviation) CPU time (in seconds) with respect to the instance size and values of  $\beta$  ( $S_1$  set of instances).

No. nodes	$\beta$			
	5	10	25	50
21	3.2 (4.5)	2.8 (3.9)	1.9 (2.1)	1.3 (1.2)
26	3.8 (4.6)	2.9 (3.5)	1.8 (2.0)	1.4 (1.4)
31	31.1 (60.2)	19.6 (35.4)	12.1 (26.8)	6.5 (12.1)
36	101.8 (199.8)	82.5 (158.1)	43.3 (85.7)	23.6 (44.6)
41	464.2 (930.3)	455.0 (960.0)	282.0 (652.0)	79.2 (161.7)

**Table 7**

CV (percent) with respect to the instance size and values of  $\alpha$  and  $\beta$  ( $S_1$  set of instances).

No. nodes	$\alpha$					$\beta$			
	5	10	25	30	50	5	10	25	50
21	0.0	0.0	1.0	1.2	4.6	0.9	1.4	0.9	2.2
26	0.0	0.0	0.7	1.1	2.5	0.9	0.7	0.4	1.5
31	0.0	0.0	0.7	2.5	8.3	1.4	1.9	1.9	4.0
36	0.0	0.0	1.1	3.2	4.1	1.1	1.9	1.6	2.3
41	0.0	0.0	2.0	2.4	5.1	1.8	1.4	2.4	1.9

comparisons, or for 92.8 percent of the cases. For each of the 225 instances, at least one of the 30 repetitions always finds an optimal solution. The average optimality gap for the cases that did not find an optimal solution is 6.5 percent and for the 82.8 percent of cases the gap is less than 10.0 percent.

To evaluate whether our IGRCP approach consistently finds good solutions among multiple repetitions, we analyze the variability of the solutions found over the 30 repetitions. We compute the coefficient of variation (CV) for the objective functions values, defined as the ratio of the standard deviation over the mean. Averaged over all 300 instances, the CV is 1.6 percent. Table 7 shows the average coefficient of variation with respect to the instance size and values of  $\alpha$  and  $\beta$ . Results show that the variability of the solutions obtained by our approach is predominantly affected by the damage level ( $\alpha$ ) of the network. In addition, when analyzing the gap between the best solution found for each of 300 instances and the solutions of each repetition, we observe that for almost 80 percent (7150 out of 9000) of the runs our algorithm generates the same best solution. For 15 percent to 20 percent of the remaining cases (1300 out of 9000), the difference between the best solution found and the solution of the specific repetition is less than 10 percent.

5.3.3. Numerical results for the set of medium and large instances,  $S_2$

We use instance set  $S_2$  to evaluate the performance of our algorithm when solving medium and large size instances. Note that DP is unable to solve these instance within the 24 hours deadline. For each of the instances we run the IGRCP algorithm with parameters  $\lambda$ ,  $\theta$  and  $\tau$  set to 50 percent, 100 and 5, respectively. Based on the favorable results of the IGRCP consistency performance analysis in Section 5.3.2 and increased computational demand of the larger instances, we restrict the algorithm run to a single repetition for each problem instance. Table 8 presents the average computing time for the combinations of number of nodes and  $\alpha$  values for which all the instances were solved in less than 10 hours. Parameter  $\alpha$ , the percentage of damaged network, has a higher impact on the computing time than the number of nodes. In addition, our results show that parameter  $\beta$ , the maximum tolerable percentage in which the path connecting each node  $i$  to the depot is allowed to increase, also affects the computing time such that when the value of  $\beta$  increases the computing time decreases. However, this effect is less significant than the one generated by the value of parameter  $\alpha$  and the number of nodes.

**Table 8**

CPU time (in seconds) with respect to the instance size and values of  $\alpha$  ( $S_2$  set of instances).

No. nodes	$\alpha$				
	5	10	25	30	50
61	1.50	2.77	173.93	331.95	5708.91
81	2.53	8.59	615.72	1531.73	22,907.68
101	5.06	31.79	1624.17	–	–
201	10.76	355.83	–	–	–
301	373.18	5818.85	–	–	–
401	1684.75	13,447.40	–	–	–

**Table 9**

Average number of iterations in which the best solution is found with respect to the instance size and values of  $\alpha$  ( $S_2$  set of instances).

No. nodes	$\alpha$				
	5	10	25	30	50
61	1.0	1.0	2.8	1.9	3.3
81	1.3	1.2	2.0	3.3	2.8
101	1.3	1.5	2.5	–	–
201	1.0	2.1	–	–	–
301	1.5	2.3	–	–	–
401	1.0	2.2	–	–	–

Table 9 presents the average number of iterations it took IGRCPC algorithm to find the best solution, by analyzing when among the five iterations it found the solution. We observe that as the value of  $\alpha$  increases, the number of iterations also increases. That is, finding a good solution is more difficult for the instances in which the network is severely damaged.

5.4. Solution analysis and insights

In this section, we analyze the structure of the solutions found with the goal to validate our optimization model and obtain some managerial insights. The discussion in this section is restricted to the solutions found by the IGRCPC algorithm for instance set  $S_1$ . While not all solutions found using IGRCPC are guaranteed to be optimal (see Section 5.3), an analysis of the subset of  $S_1$  instances for which IGRCPC does find an optimal solution yields analogous results.

5.4.1. Comparison to myopic heuristic

As discussed earlier, one important distinction between our model and the existing literature is the fact that we enforce predecessor constraints and ensure feasible repair schedules, where a damaged node cannot be repaired until the crew has repaired all the damaged nodes on a path between that node and the depot. The effect of these constraints is especially important when the crew needs to repair a sequence of nodes before a large demand can be connected. In such situations, repairing any of the intermediate nodes will have a small effect on the objective function value at best, and good solution strategies can only be found by considering several repairs into the future. The importance of this strategy is demonstrated by comparing our solutions to those found by a greedy heuristic that myopically selects the next node to repair without considering the effect of future repairs (we call such heuristic “myopic”). To this end, we considered 280 out of 300 instances in  $S_1$  for which two or more nodes needed to be repaired to obtain full accessibility.

For 127 out of 280 instances, IGRCPC found a better solution than the myopic heuristic, with an average difference between the two solutions of 54.52 percent and a maximum difference of almost 300 percent. The myopic heuristic found a better solution in 8 cases, with an average difference of 1.64 percent and a maximum difference of 3.19 percent. For the instances in which IGRCPC found a better solu-

**Table 10**

Average percentage of nodes repaired for various  $\alpha$  values.

Values of $\alpha$ (percent)	5	10	25	30	50
Nodes repaired (percent)	37.24	28.01	29.72	29.69	31.82

**Table 11**

Average percentage of nodes repaired for various  $\beta$  values.

Values of $\beta$ (percent)	5	10	25	50
Nodes repaired (percent)	39.88	34.81	26.02	19.55

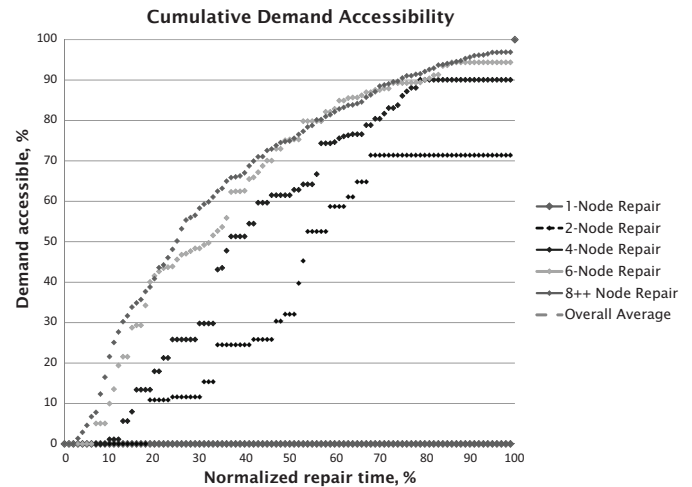


Fig. 3. Aggregate and overall averaged cumulative demand accessibility versus normalized time.

tion, the myopic heuristic solution always repairs a larger number of nodes. On average, the myopic heuristic repairs 58.78 percent of the damaged nodes while our IGRCPC algorithm repairs only 34.04 percent. Moreover, for the instances with 10 or more damaged nodes, the myopic heuristic repairs, on average, 18.1 nodes while the IGRCPC algorithm repairs 8.5 nodes. This demonstrates that failing to look ahead and, by doing so, ignoring the effects of repairing a given node on future demand accessibility generally results in a repair operation that takes much more time and consumes more resources, by repairing more roads, than strictly necessary.

5.4.2. Solution structure and managerial insights

The first important observation we make from a policy and planning perspective is that only a fraction of the damaged roads (i.e., repair nodes) need to be repaired to ensure accessibility for the entire network (see Tables 10 and 11). As expected, that percentage decreases as  $\beta$  values increase, since this corresponds to more relaxed requirements for a demand node to be accessible. The correlation with  $\alpha$  values is not as clear. While there is an overall positive trend for the percentage of repaired nodes, instances with  $\alpha = 5$  percent correspond to a particularly high number of repaired nodes. This can be explained by an observation that for such low value of  $\alpha$  a number of instances have only one or two damaged nodes, and the solution requires to repair all of them.

By analyzing the cumulative demand that becomes accessible over the duration of the repair operation we observe that, on average, the percentage of accessible demand follows a law of diminishing returns. Fig. 3 illustrates how cumulative demand accessibility changes over the course of the entire repair operation, for different numbers of nodes repaired. The figure graphs demand averaged over all  $S_1$  instances and a few examples of demand averaged over subsets of instances with the same number of nodes repaired. To account for different repair times between the instances, the x-axis corresponds to

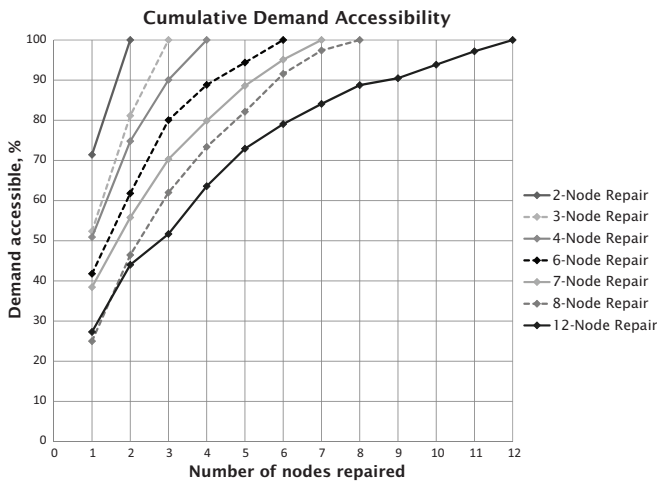


Fig. 4. Aggregate cumulative demand accessibility versus number of nodes repaired.

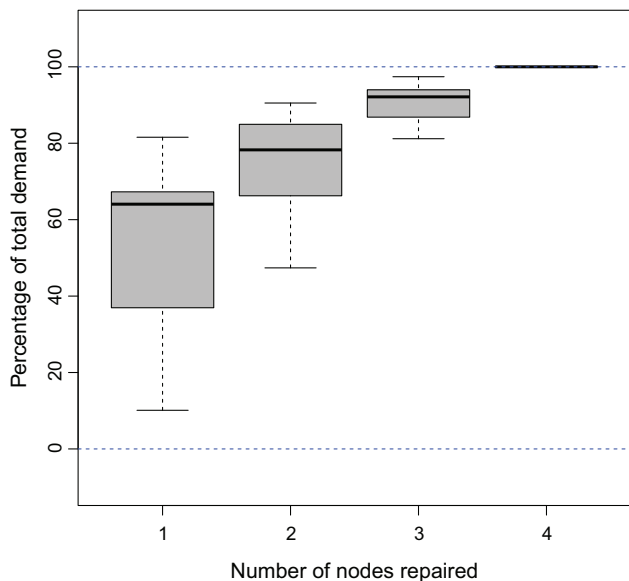


Fig. 5. Boxplot of cumulative demand accessibility for all instances with 4 nodes repaired.

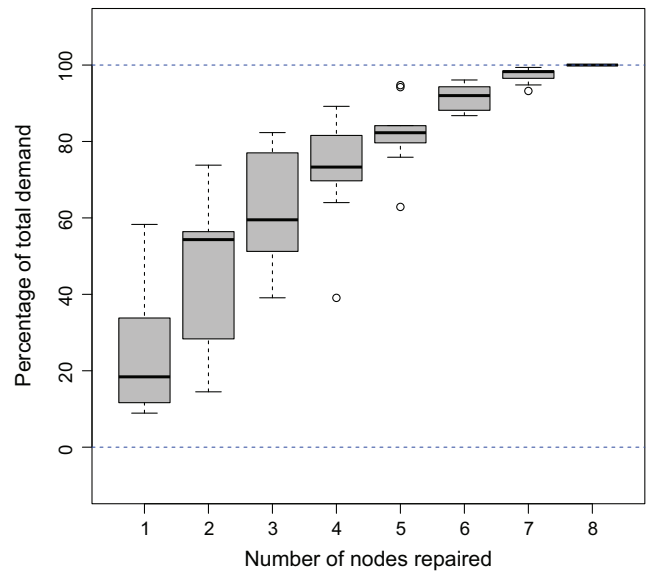


Fig. 6. Boxplot of cumulative demand accessibility for all instances with 8 nodes repaired.

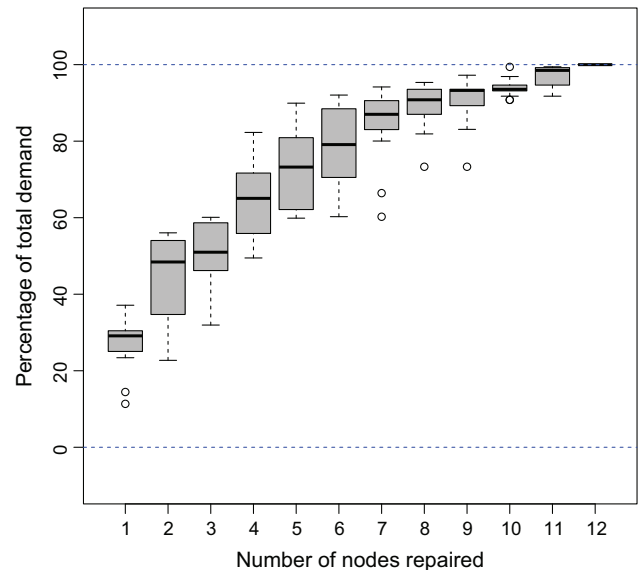


Fig. 7. Boxplot of cumulative demand accessibility for all instances with 12 nodes repaired.

normalized repair time. From the graphs, we observe an initial delay until the first set of demand nodes become accessible, which corresponds to the travel and repair times of the crew to complete its first assignment. Then, on average, the percentage of marginal demand connected decreases with the passing of time and with each additional repair of a node, resulting in a concave cumulative demand curve. We make the same observation in Fig. 4, illustrating accessible demand versus number of nodes repaired. Towards the end of the repair operation (Fig. 3), the curves become flat, corresponding to the repair crew traveling farther away to repair the subsequent node or repairing multiple nodes before the next improvement in demand accessibility. At the very end, we observe a jump in percent of accessible demand, which is explained by a considerable amount of demand that becomes accessible after the last repair is carried out by the crew at the completion of the repair operation.

While, on average, the rule of diminishing returns is clearly illustrated, there is a lot of variation from instance to instance with regards to the percentage demand connected by each consecutive node repaired. This fact highlights the importance of solving the optimization model for each individual instance, and further explains poor performance of the myopic heuristic. Boxplots in Figs. 5–7 illus-

trate the distribution of cumulative demand accessible after each repaired node for all instances that require four, eight and twelve nodes to be repaired, respectively. While all instances connect a progressively higher percentage of demand as more nodes are repaired, we observe high variance earlier in the operation. Fig. 8 illustrates individual examples of the cumulative demand accessibility over the duration of the repair operation. Here, we see that Examples 1, 3, 5 and 6 have a significantly lower percentage of demand connected early on in comparison to Examples 2 and 4. At the same time, Examples 2, 3, 4 and 6 illustrate instances where multiple nodes need to be repaired before the next improvement in network accessibility. More specifically, in case of Example 3 the sixth node repaired does not directly contribute to the cumulative accessible demand, but its repair is necessary in order for the seventh repaired node to connect a substantial percentage of demand. Examples 4 and 6 illustrate the cases where multiple consecutive repairs are required before the next improvement in accessible demand. This diverse set of

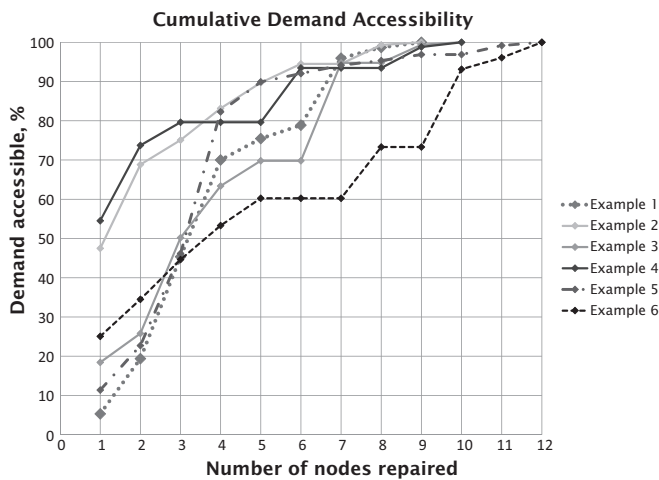


Fig. 8. Individual examples of cumulative demand accessibility.

examples highlights the importance of finding a close-to-optimal solution of each individual problem instance.

## 6. Conclusion and future research

In this paper, we study the problem of emergency repair of a rural road network that has been damaged by the occurrence of a natural disaster. We address the scheduling and routing of a repair crew, starting from a single depot, while optimizing accessibility to the towns and villages that demand humanitarian relief. This is an important problem given the impact that a functioning road network has on providing the opportune relief to the affected population. This paper extends the traditional network upgrading problems by considering not only the selection of the edges to be repaired or upgraded, but also the time dependency and crew routing associated with the required repair work.

While the NRCSR was developed mainly to be used as a support tool in the initial response stage, our problem setting and solution methods could be directly extended to the later recovery stage, as long as the time scale of travel times and repair times are equivalent. Problems such as connecting remote villages to a central medical facility would be applicable here, where moving the repair equipment from one node to the next requires significant time that is comparable in scale to the actual repair time.

We develop a dynamic programming model and discuss efficient implementation techniques that allows us to find optimal solutions for a range of instance sizes of the problem. However, the DP approach limits the size of the road networks that can be solved within a 24-hour time limit. This fact motivated the development of a heuristic (IGRCP) that we develop for our problem. We run extensive computational experiments to analyze the difficulty of solving instances of the NRCSR depending on the values of three distinct parameters. Two of the parameters are beyond the control of the decision maker: the number of nodes or population centers in a network and the level of network damage caused by the disaster, which is denoted by parameter  $\alpha$  and determines the number of edges requiring repair. The third parameter,  $\beta$ , is the maximum tolerable percentage by which a path connecting each demand node  $i$  to the depot is allowed to increase in order to be considered accessible. This parameter can be controlled by the decision maker and depends on the specifics of each setting.

Our results show that the factor that has the highest impact on the difficulty of solving an NRCSR instance is the  $\alpha$  parameter. That is, the most severely affected networks correspond to the most difficult instances to solve. This is mainly due to the fact that the NRCSR is, in part, a combinatorial problem for which a solution is represented by a permutation of the damaged nodes that have to be repaired. There-

fore, as the level of network damages increases, so does the number of nodes that have to be repaired, which, in turn, increases the number of possible permutations to be considered. An instance of the NRCSR also becomes more difficult to solve when the size of the network increases. However, the effect of this parameter is not as significant as it is for parameter  $\alpha$ . Finally, when the value of the parameter  $\beta$  increases, the time required to solve each instance decreases. This phenomenon is explained by the fact that increasing the value of  $\beta$  implies that the decision maker is willing to accept a longer path between a demand node and the depot in order to consider that demand point accessible.

When comparing the difficulty of solving an instance of the NRCSR using the dynamic programming model versus the IGRCP algorithm, the results show that the two algorithms experience similar trends in the levels of difficulty. That is, the instances that are easily solved by the DP are also easily solved by the IGRCP. Similarly, for the instances that could not be solved to optimality by DP within the time limit, the computational time of the IGRCP algorithm is also higher. However, the DP method performs better than the IGRCP for instances with few nodes and a small percentage of network damage ( $\alpha$ ), while the IGRCP performs better than the DP when both the size of the instance network and the level of damage increase. We use the IGRCP algorithm to solve instances containing up to 401 nodes, and the results confirm that parameter  $\alpha$  strongly correlates with the level of difficulty of an NRCSR instance.

Analysis of the solution structure provides important managerial insights for our problem. We observe that only a fraction of damaged nodes (approximately 30 percent, on average) needs to be repaired to establish accessibility for the entire network. In addition, while, on average, each subsequent repair of a damaged node follows a law of diminishing return, that property does not necessarily hold true for all instances. This fact, along with poor performance of the tested myopic heuristic, establish the need for optimization of the repair crew schedule and rout for each individual instance. This is especially the case when a sequence of multiple nodes need to be repaired before observing improvement in the network accessibility.

In this paper, we have designed an objective function to accurately capture and evaluate the accessibility measure, since it is closely tied to the performance of our solution strategies. An interesting open research direction to pursue in future work is to study alternative measures and objective functions for estimating network accessibility. Moreover, the efficiency of the IGRCP algorithm can be potentially improved by restricting the complete exploration of the current solution neighborhood, at each iteration, and instead, focusing on specific regions of the feasible solution space. Note, however, that from the perspective of the quality of the solutions, the suggested measures might not have any positive effect on the value of the objective function itself, just the algorithm run time.

From our extensive study of the NRCSR, we are able to identify problem properties that can be useful in extending this work to more complex and realistic settings. The first promising research direction is to extend this problem to more than one repair crew and more than one depot node. This extended problem would model real-life situations where different organizations or relief entities are involved in the emergency reconstruction operations of the region's road network. In addition, the collaboration and coordination aspects of such problem settings would have to be considered. For example, one would have to answer the question of what happens when two or more crews are able to work simultaneously on repairing a single damaged node. Another interesting extension that emerges from considering multiple depots is the distribution allocation decisions. In such cases, one has to consider the limited amount of resources available at each depot, and the repairing decisions are restricted by the need of fulfilling the relief demand of the towns that are made accessible.

## Appendix A. List of symbols

$\mathcal{V}$	Set of nodes $\{0, 1, \dots, n + 1\}$ ; node 0 is the depot; node $n + 1$ is a dummy node; $\mathcal{V} = \{0\} \cup \mathcal{V}_d \cup \mathcal{V}_r \cup \{n + 1\}$
$\mathcal{V}_d$	Set of demand nodes; $\mathcal{V}_d \subset \mathcal{V}$
$\mathcal{V}_r$	Set of damaged nodes requiring repair; $\mathcal{V}_r \subset \mathcal{V}$
$\mathcal{E}$	Set of edges or roads
$e_{ij}$	Edge connecting nodes $i$ and $j$ ; $e_{ij} \in \mathcal{E}$
$\mathcal{E}(k)$	Set of arcs adjacent to node $k \in \mathcal{V}$ , $\mathcal{E}(k) \subset \mathcal{E}$
$w_i$	Relief demand of node $i \in \mathcal{V}$
$t_{ij}$	Time required to travel between nodes $i$ and $j$ on edge $e_{ij}$
$s_j$	Time required to repair damaged node $j \in \mathcal{V}_r$
$d_{ij}$	Length of edge $e_{ij}$ connecting nodes $i$ and $j$ , i.e., distance between nodes $i$ and $j$
$d_e$	Length of edge $e$
$D_i$	Maximum acceptable distance from node $i \in \mathcal{V}_d$ to the depot in order for $i$ to be accessible to it

## References

- Altay, N., & Green, W. (2006). OR/MS research in disaster operations management. *European Journal of Operational Research*, 175(1), 475–493.
- Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6), 621–636.
- Beamon, B., & Balcik, B. (2008). Performance measurement in humanitarian relief chains. *International Journal of Public Sector Management*, 21(1), 4–25.
- Benson, C., Twigg, J., & Myers, M. (2001). NGO initiatives in risk reduction: An overview. *Disasters*, 25(3), 199–215.
- Besiou, M., Stapleton, O., & Van Wassenhove, L. (2011). System dynamics for humanitarian operations. *Journal of Humanitarian Logistics and Supply Chain Management*, 1(1), 78–103.
- Brown, G. G., & Vassiliou, A. L. (1993). Optimizing disaster relief: Real-time operational and tactical decision support. *Naval Research Logistics (NRL)*, 40(1), 1–23.
- Cavdaroglu, B., Hammel, E., Mitchell, J. E., Sharkey, T. C., & Wallace, W. A. (2013). Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. *Annals of Operations Research*, 203(1), 279–294.
- Chen, Y., & Tzeng, G. (1999). A fuzzy multi-objective model for reconstructing post-earthquake road-network by genetic algorithm. *International Journal of Fuzzy Systems*, 1(2), 85–95.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.
- Ergun, O., Karakus, G., Keskinocak, P., Swann, J., & Villarreal, M. (2010). Operations research to improve disaster supply chain management. In *Wiley encyclopedia of operations research and management science, number 2*. John Wiley & Sons, Inc..
- Feng, C., & Wang, T. (2003). Highway emergency rehabilitation scheduling in post-earthquake 72 hours. *Journal of the Eastern Asia Society for Transportation Studies*, 5, 3276–3285.
- Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2), 109–133.
- Goemans, M. X., & Williamson, D. P. (1995). A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2), 296–317.
- Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics (NRL)*, 34(3), 307–318.
- Guha-Sapir, D., Vos, F., Below, R., & Ponserre, S. (2011). Annual disaster statistical review. (Technical report). Centre for Research on the Epidemiology of Disasters (CRED), Institute of Health and Society (IRSS), and Université Catholique de Louvain.
- Holguín-Veras, J., Pérez, N., Jaller, M., Wassenhove, L. N. V., & Aros-Vera, F. (2013). On the appropriate objective function for post-disaster humanitarian logistics models. *Journal of Operations Management*, 31(5), 262–280.
- Karlaftis, M., Kepaptsoglou, K., & Lambropoulos, S. (2007). Fund allocation for transportation network recovery following natural disasters. *Journal of Urban Planning and Development*, 133(1), 82–89.
- Klingman, D., Napier, A., & Stutz, J. (1974). NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5), 814–821.
- Kovács, G., & Spens, K. (2011). Trends and developments in humanitarian logistics – A gap analysis. *International Journal of Physical Distribution & Logistics Management*, 41(1), 32–45.
- Kunz, N., & Reiner, G. (2012). A meta-analysis of humanitarian logistics research. *Journal of Humanitarian Logistics and Supply Chain Management*, 2(2), 116–147.
- Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3), 345–358.
- Liberatore, F., Ortuño, M., Tirado, G., Vitoriano, B., & Scaparra, M. (2014). A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. *Computers & Operations Research*, 42(0), 3–13.
- Matisziw, T. C., Murray, A. T., & Grubestic, T. H. (2010). Strategic network restoration. *Networks and Spatial Economics*, 10(3), 345–361.
- Maya Duque, P., Coene, S., Goos, P., Sörensen, K., & Spieksma, F. (2013). The accessibility arc upgrading problem. *European Journal of Operational Research*, 224(3), 458–465.
- Maya Duque, P. A., Dolinskaya, I. S., & Sorensen, K. (2014). Network repair crew scheduling and routing for emergency relief distribution problem. Working Paper 14-03, Northwestern University, Department of Industrial Engineering and Management Sciences.
- Maya Duque, P. A., & Sorensen, K. (2011). A grasp metaheuristic to improve accessibility after a disaster. *OR Spectrum*, 33(3), 525–542.
- PAHO (2000). *Natural disasters: Protecting the public's health*. Pan American Health Organization.
- Pedraza Martinez, A. J., Stapleton, O., & Van Wassenhove, L. N. (2010a). Using OR to support humanitarian operations: Learning from the Haiti earthquake. INSEAD Faculty & Research Working Paper 2010/47/TOM/ISIC.
- Sheu, J.-B. (2007). Challenges of emergency logistics management. *Transportation Research Part E: Logistics and Transportation Review*, 43(6), 655–659.
- Stilp, K., Ergun, O., & Kekinocak, P. (2012). Managing debris collection and disposal operations. (Technical report). Center for Humanitarian Logistics, Georgia Institute of Technology.
- Toth, P., & Vigo, D. (2002). *The vehicle routing problem: 9*. Siam.
- Trunick, P. (2005). Delivering relief to tsunami victims. *Logistics Today*, 46(2), 1–9.
- Tuzun Aksu, D., & Ozdamar, L. (2014). A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61, 56–67.
- Van Wassenhove, L. (2003). New interesting POM cases from Europe. *POMS Chronicle*, 10, 19.
- Van Wassenhove, L., Martinez, A., & Stapleton, O. (2010). An analysis of the relief supply chain in the first week after the haiti earthquake. INSEAD Humanitarian Research Group.
- Yan, S., & Shih, Y. (2009). Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers & Operations Research*, 36(6), 2049–2065.
- Yan, S., & Shih, Y.-L. (2012). An ant colony system-based hybrid algorithm for an emergency roadway repair time-space network flow problem. *Transportmetrica*, 8(5), 361–386.