European Journal of Operational Research 272 (2019) 352–363

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

An exact branch-and-price algorithm for scheduling rescue units during disaster response

Gerhard Rauchecker[a,*], Guido Schryen[b]

[a] Department of Management Information Systems, University of Regensburg, Universitaetsstrasse 31, Regensburg 93053, Germany
[b] Department of Business Information Systems, Paderborn University, Warburger Strasse 100, Paderborn 33098, Germany

ARTICLE INFO

Article history:
Received 15 July 2017
Accepted 5 June 2018
Available online 15 June 2018

Keywords:
OR in disaster relief
Disaster operations management
Scheduling
Branch-and-price,

ABSTRACT

In disaster operations management, a challenging task for rescue organizations occurs when they have to assign and schedule their rescue units to emerging incidents under time pressure in order to reduce the overall resulting harm. Of particular importance in practical scenarios is the need to consider collaboration of rescue units. This task has hardly been addressed in the literature. We contribute to both modeling and solving this problem by (1) conceptualizing the situation as a type of scheduling problem, (2) modeling it as a binary linear minimization problem, (3) suggesting a branch-and-price algorithm, which can serve as both an exact and heuristic solution procedure, and (4) conducting computational experiments – including a sensitivity analysis of the effects of exogenous model parameters on execution times and objective value improvements over a heuristic suggested in the literature – for different practical disaster scenarios. The results of our computational experiments show that most problem instances of practically feasible size can be solved to optimality within ten minutes. Furthermore, even when our algorithm is terminated once the first feasible solution has been found, this solution is in almost all cases competitive to the optimal solution and substantially better than the solution obtained by the best known algorithm from the literature. This performance of our branch-and-price algorithm enables rescue organizations to apply our procedure in practice, even when the time for decision making is limited to a few minutes. By addressing a very general type of scheduling problem, our approach applies to various scheduling situations.

1. Introduction

Managing natural and man-made disasters, such as earthquakes, floods, droughts, or industrial accidents, has become an important issue in today's world. According to the International Federation of Red Cross and Red Crescent Societies (IFRC), there have been 6699 reported disasters in the decade between 2003 and 2012 with more than 1.1 million people killed and financial losses of more than US$ 1.5 trillion (IFRC, 2013). One of the most severe natural disaster ever is the 2011 Tohoku-oki earthquake, tsunami, and nuclear accident in Japan with an estimated US$ 211 billion of direct damage in addition to 19,000 fatalities (Kajitani, Chang, & Tatano, 2013). Although the list could be continued, these statistics suffice to show the importance of constantly refining disaster operations management to reduce the impact of disasters on humankind.

Disaster operations management (DOM) has received considerable attention in the OR and MS literature, see Green and Kolesar (2004), Altay and Green (2006), and Galindo and Batta (2013) for an overview. Tasks in DOM can be classified into four main phases: mitigation, preparedness, response, and recovery. One of the most critical tasks in DOM is decision support for disaster operations centers during disaster response and in particular the scheduling of rescue units to process disaster incidents (Wex, Schryen, Feuerriegel, & Neumann, 2014). We study this problem taking into account several real-world properties: (i) each rescue unit may have multiple capabilities, such as medical care, fire extinguishing, and search-and-rescue, while each incident may require several of these capabilities. When not all of the required capabilities of an incident can be provided by a single rescue unit, the collaboration of several rescue units is necessary. Collaboration can occur in different forms, including what we call *tight* and *loose* collaboration. While the former requires that all rescue units are available before they can start their operation, the latter allows rescue units to work independently. For example, when an incident requires the capabilities of both firemen and medical staff, firemen can and should start rescuing buried people although medical staff

* Corresponding author.
E-mail addresses: gerhard.rauchecker@ur.de (G. Rauchecker), schryen@posteo.de (G. Schryen).

https://doi.org/10.1016/j.ejor.2018.06.010
0377-2217/

is still not available. In our manuscript, we consider loose collaboration as also done in previous work (e.g., Schryen, Rauchecker, & Comes, 2015; Wex, Schryen, & Neumann, 2013). (ii) The processing of incidents is non-preemptive, (iii) processing times and travel times are incident- and unit-dependent, and (iv) each incident has a specific severity level. Since disaster incidents are time-critical, especially when human lives are in danger, Rolland, Patterson, Ward, and Dodin (2010) suggest to use completion times of incidents as a proxy for overall harm. Building on this approach and accounting for the nature of loose collaboration (when rescue units required by an incident can start processing this incident independently), we minimize the weighted sum of completion times, where weights are incident-specific and where each completion time refers to a particular pair of rescue unit k and incident j and denotes the time at which unit k has finished its processing of incident j. We refer to this problem as *Disaster Response Scheduling Problem* (DRSP).

Even though this type of problem is highly relevant in practical contexts, it has rarely been investigated in the OR literature. For example, Wex et al. (2014) present heuristics for a specialization of DRSP in which incidents have only a single requirement, making collaboration of rescue units obsolete. Another special case of DRSP was investigated by Rolland et al. (2010), who introduced meta-heuristics for settings in which incidents do not have specific severity levels. Wex et al. (2013) and Schryen et al. (2015) show that DRSP itself is NP-hard in the strong sense. They introduce heuristics and evaluate the quality of their solutions using lower bounds obtained by an integer quadratic program relaxation. Bodaghi and Ekambaram (2016) present a mixed-integer linear program, using a commercial solver to find the optimal solution for one small DRSP instance with four rescue units as a case study. However, their case study instance does not involve multi-capability rescue units, although this could be accounted for by their model. To the best of our knowledge, no further algorithms for solving the DRSP have been suggested in the literature.

We close this research gap by (1) formulating DRSP as a binary linear program, (2) developing a novel branch-and-price (b&p) algorithm to solve the proposed mathematical program optimally, and (3) conducting computational experiments to assess the performance of the proposed algorithm. We evaluate execution times of the algorithm and compare the solutions obtained by our b&p algorithm to solutions returned by the SCHED heuristic suggested by Schryen et al. (2015), which is currently the best performing algorithm for DRSP in the literature. We show that our b&p algorithm enables decision makers in disaster operation centers to improve the quality of their scheduling decisions substantially. Consequently, this helps decreasing both casualties and economic losses.

The DRSP represents a very general form of scheduling problems. It subsumes non-preemptive scheduling on identical/uniform/unrelated parallel machines with the objective function being the (weighted) sum of completion times. In addition, it accounts for sequence-dependent setup times (Allahverdi, 2015). As a consequence, our b&p algorithm can be used for solving many types of scheduling problems.

The remainder of the paper is structured as follows: Section 2 presents and discusses relevant literature. In Section 3, we formulate DRSP as a binary linear optimization model. In Section 4, we present our b&p algorithm to solve the proposed model exactly. We evaluate the b&p algorithm in computational experiments in Section 5. The results of the experiments are discussed in Section 6 before we finally conclude in Section 7.

2. Related work

The four phases of disaster operations management are widely considered as mitigation, preparedness, response, and recovery (Altay & Green, 2006; Galindo & Batta, 2013) and are often arranged as a life cycle. Mitigation tasks include activities for reducing the long-term risk of a disaster (Paul & Hariharan, 2012; Tamura, Yamamoto, Tomiyama, & Hatono, 2000). The preparedness phase includes all activities performed before a disaster that aim at providing a more efficient processing of tasks once the disaster strikes (Albores & Shaw, 2008; Salmerón & Apte, 2010). While mitigation and preparedness refer to the time before a disaster, response phase activities take place in the immediate aftermath of a disaster. The main objective here is the deployment of vital resources to affected people (Fiedrich, Gehbauer, & Rickers, 2000; Lodree & Taskin, 2009). Finally, the recovery stage includes tasks that restore the normal functioning of the community (Liberatore, Ortuño, Tirado, Vitoriano, & Scaparra, 2014; Sahebjamnia, Torabi, & Mansouri, 2015).

During the response phase, in which our investigated problem is situated, researchers offer a variety of methods to support decisions. These include mathematical programming, probability theory and statistics, simulation, and decision theory to name only a few (Simpson & Hancock, 2009). As outlined in the introduction, the decision problem DRSP under investigation deals with scheduling rescue units to process a set of disaster incidents. These incidents in particular may have multiple requirements, thus enabling rescue unit collaboration. Rolland et al. (2010) have published a paper which investigates the scheduling of (loosely) collaborative rescue units. They model the situation as a resource-constrained project scheduling problem and present two meta-heuristics for its solution. However, their setting does not account for different severity levels of incidents.

Wex et al. (2013) model DRSP as a quadratic binary program and present a heuristic for its solution. However, their approach is a crude probabilistic exploration of the feasible solution space. Schryen et al. (2015) present a more sophisticated heuristic for DRSP based on scheduling theory. The authors evaluate their approach against a heuristic modeling best-practice behavior and against lower bounds of a quadratic programming relaxation. Bodaghi and Ekambaram (2016) present a mixed-integer linear program for DRSP and calculate the optimal solution for a small case study instance with four rescue units using a commercial solver. Although their model could account for it, their case study instance does not involve multi-capability rescue units. To the best of our knowledge, no further algorithms for solving DRSP have been suggested in the literature.

To develop an exact algorithm for DRSP, we draw upon connections to the closely related field of machine scheduling, see Brucker (2007); Pinedo (2016), and Rabadi (2016) for an overview. DRSP is a generalization of the *Rescue Unit Assignment and Scheduling Problem* (RUASP) in which each incident has only a single requirement, which makes collaboration obsolete. Wex et al. (2014) show that RUASP is a scheduling problem on unrelated parallel machines with sequence- and machine-dependent setup times and a weighted sum of completion times as objective function. Using the three-field notation by Graham, Lawler, Lenstra, and Rinnooy Kan (1979), RUASP can be classified as $R/s_{ijk}/\sum w_j C_j$, which in turn is a generalization of the machine scheduling problem $R/s_{ij}/\sum w_j C_j$ in which setup times are not machine-dependent. According to the extensive literature reviews by Allahverdi, Gupta, and Aldowaisan (1999), Allahverdi, Ng, Cheng, and Kovalyov (2008) and Allahverdi (2015), all research articles on these two machine scheduling problems focus on heuristics due to a lack of efficiency in solving proposed mathematical programming formulations exactly (Arnaout, Rabadi, & Mun, 2006; Chen, 2015; Rauchecker & Schryen, 2015; Tsai & Tseng, 2007; Weng, Lu, & Ren, 2001; Wex et al., 2014).

Regarding a generalization of the $R/s_{ij}/\sum w_j C_j$ problem, Lopes, Alvelos, and Lopes (2014) and Lopes and de Carvalho (2007) present a branch-and-price (b&p) algorithm for the

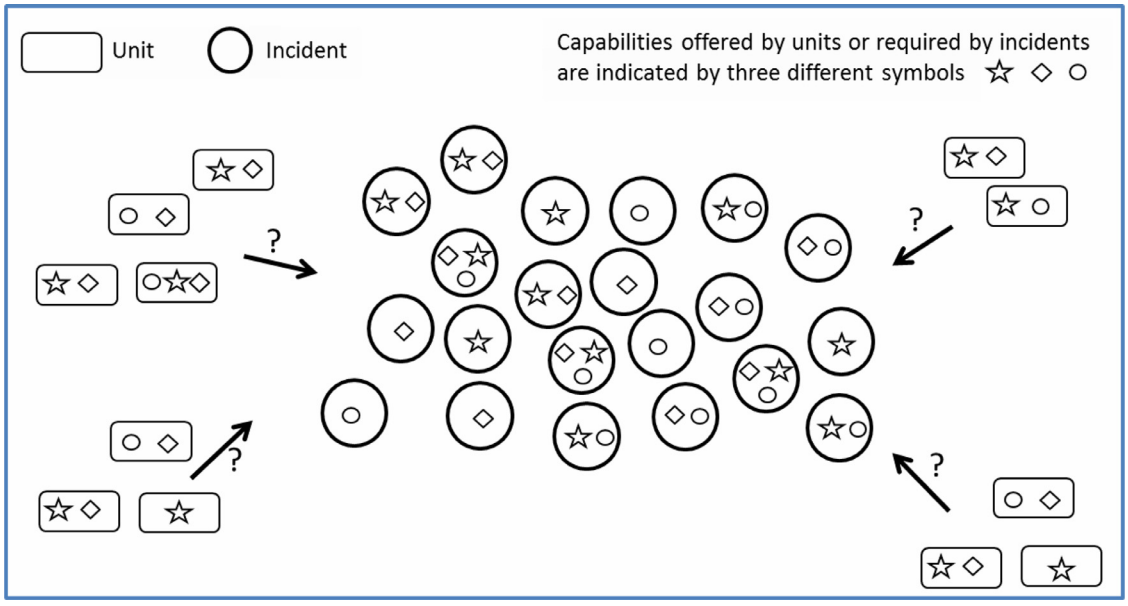

Fig. 1. Sample scenario for the disaster response scheduling problem (DRSP) with $n = 21$ incidents, $m = 12$ rescue units, and $r = 3$ capabilities.

$R/s_{ij}/\sum w_jT_j$ problem in which jobs have due dates and the objective is to minimize the sum of weighted tardiness penalties. This scheduling problem is a generalization of $R/s_{ij}/\sum w_jC_j$ (the problems coincide when all due dates are 0) but not a generalization of RUASP or DRSP in which setup-times are machine-dependent. Consequently, their b&p algorithm for $R/s_{ij}/\sum w_jT_j$ cannot be applied to DRSP. However, we extend their approach to develop a novel b&p algorithm for DRSP in this paper.

3. Optimization model

In this section, we suggest a mathematical formulation as an optimization model for DRSP. A set $\{1,\dots,n\}$ of n disaster incidents has to be processed by a set $\{1,\dots,m\}$ of m rescue units. Each rescue unit may offer different capabilities and each incident may require multiple capabilities. A sample scenario for DRSP is given in Fig. 1. The set of possible requirements/capabilities is represented by $\{1,\dots,r\}$. Set $cap_{kq} = 1$ when unit k offers capability q (0 otherwise) and $req_{jq} = 1$ when incident j requires capability q (0 otherwise). A rescue unit is only eligible for processing one or more of an incident's requirements if it offers the respective capabilities. For an incident j, let M_j denote the set of rescue units that are capable of processing at least one requirement of j.

Let p_j^k be the processing time of a unit k for an incident j and let s_{ij}^k be the travel time of a unit k between the locations of incidents i and j. The time required by unit k to reach the location of incident j from its current position (e.g., a depot) is represented by s_{0j}^k.[1] Furthermore, we denote by w_j the weight of an incident j, which corresponds to its severity level. The processing of an incident by a rescue unit is non-preemptive. Using this notation, a sample schedule for DRSP is shown in Fig. 2. In order to determine the overall harm, we calculate the weighted sum of completion times, where completion time refers to a particular pair of rescue unit and incident. Whenever a rescue unit finishes its processing of an incident j, the current time (weighted with w_j) is added to the objective function and the unit can move on to the next incident. In the example in Fig. 2, unit 1 contributes $(3+6)\cdot 3+(9+2+3)\cdot 2+(14+2+4)\cdot 2 = 95$ to the objective function while the contributions of unit 2 and unit 3 are 55 and 41, respectively. This leads to a weighted sum of completion times (i.e., the objective function value) of 191.

The essence and motivation of our objective function lies in our approach to consider loose collaboration. In this setting, different rescue units do not have to process the requirements of an incident at the same time. For each rescue unit that processes an incident i, it holds that the harm resulting from a delayed processing increases with the extent of this delay. These characteristics provide the rationale to add, for each incident i, the completion times of all rescue units that process incident i to the objective function, rather than considering the maximum of completion times among all rescue units that process incident i, for example. In particular, in scenarios where a certain capability required by several incidents can be found at only one rescue unit k^*, some of these incidents may need to wait for rescue unit k^* much longer than for other rescue units, which can process parts of the incident much earlier. Considering the maximum of completion times of all rescue units that process a particular incident would ignore the harm-reducing effects of all rescue units that process parts of the incident earlier than rescue unit k^*. This issue does not occur when using the sum of completion times, since there is an incentive that each requirement is processed as soon as possible. Further discussion on our objective function and its comparison with the *max completion time* objective function is provided in Appendix A. In summary, we argue that the *max completion time* objective function might be more suitable for tight collaboration (where units have to jointly process different requirements at the same time) but the *sum of completion times* objective function is suitable for the loose collaboration setting that we consider in DRSP.

In the literature, DRSP has been modeled by binary programs which use decision variables X_{ij}^k, indicating whether an incident i is processed directly before incident j on unit k (Bodaghi & Ekambaram, 2016; Schryen et al., 2015; Wex et al., 2013). However, algorithms based on this modeling approach are practically inefficient and corresponding papers do not report optimal solutions even for medium-sized instances. Therefore, we present a novel formulation in which the decision variables indicate whether an entire schedule is used for a rescue unit or not.

A schedule $\omega = (j_1,\dots,j_h)$, with $1 \le h$ (or $h = 0$ if ω is the empty schedule), is defined as a tuple of pairwise different

[1] Consequently, we can view $i = 0$ as an artificial incident modeling the current position of the rescue unit. Using the term *incident*, however, we refer to a regular disaster incident $j \in \{1,\dots,n\}$ throughout the paper unless otherwise stated.

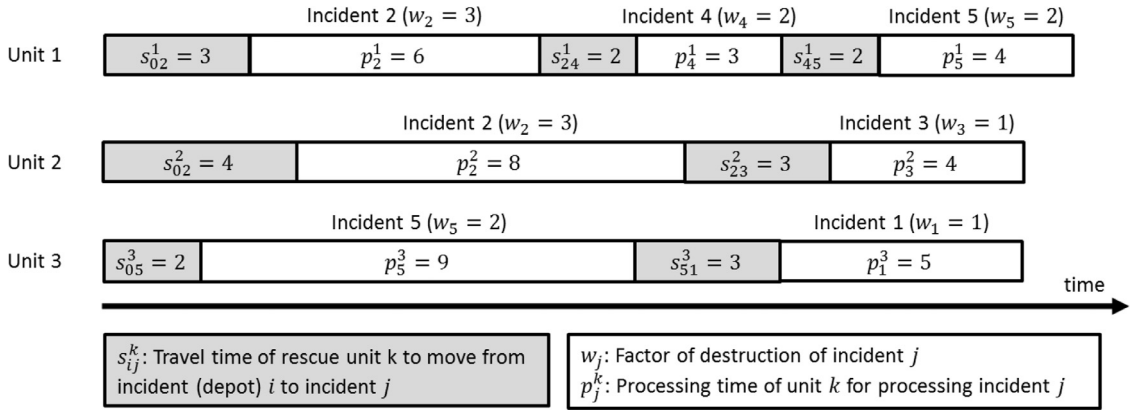

Fig. 2. Sample schedule with $n = 5$ incidents and $m = 3$ rescue units.

Table 1
Notation for the mathematical formulation.

Notation	Description
$j = 1, \ldots, n$	Disaster incidents
$k = 1, \ldots, m$	Rescue units
$q = 1, \ldots, r$	Requirements/capabilities
cap_{kq}	Binary capability indicator (k offers q or not)
req_{jq}	Binary requirement indicator (j requires q or not)
M_j	Set of units capable of processing j
w_j	Weight of j
p^k_j	Time required by k to process j
s^k_{ij}	Time required by k to travel from i to j
$\omega \in \Omega^k$	Feasible schedules on k
c^k_ω	Weighted sum of completion times of ω on k
$a_{j\omega}$	Binary occurrence indicator (ω contains j or not)
x^k_ω	Binary decision variable (ω used on k or not)

incidents $j_1, \ldots, j_h$. A schedule $\omega = (j_1, \ldots, j_h)$ is feasible on a unit k if and only if $k \in M_{j_l}$ for all $l = 1, \ldots, h$. The tuple represents the order in which the incidents are processed by rescue unit k. The set of all feasible schedules on unit k is denoted by Ω^k. The weighted sum of completion times c^k_ω of a schedule $\omega = (j_1, \ldots, j_h)$ on a unit k is be defined as

$$c^k_\omega := \sum_{l=1}^{h} w_{j_l} \cdot \left(\sum_{g=1}^{l} s^k_{j_{g-1} j_g} + p^k_{j_g} \right). \tag{1}$$

Let $a_{j\omega} \in \mathbb{Z}$ be the binary parameter which indicates how often incident j is contained in schedule ω. For each unit k and each schedule $\omega \in \Omega^k$, we introduce a binary decision variable x^k_ω being 1 if ω is used for k and 0 otherwise. This allows for the following binary linear programming formulation for DRSP (cf. Table 1 for an overview on the notation):

$$\min \quad \sum_{k=1}^{m} \sum_{\omega \in \Omega^k} c^k_\omega \cdot x^k_\omega \tag{BinLP}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{\omega \in \Omega^k} cap_{kq} \cdot a_{j\omega} \cdot x^k_\omega \geq req_{jq} \qquad \forall j = 1, \ldots, n; q = 1, \ldots, r \tag{2}$$

$$\sum_{\omega \in \Omega^k} x^k_\omega = 1 \qquad \forall k = 1, \ldots, m \tag{3}$$

$$x^k_\omega \in \{0, 1\} \qquad \forall k = 1, \ldots, m; \omega \in \Omega^k \tag{4}$$

The objective function of the minimization model (BinLP) is the weighted sum of completion times of all schedules that are used on the rescue units. Constraint set (2) ensures that each requirement of each incident is processed by a suitable rescue unit. Constraint set (3) assures that exactly one (possibly empty) schedule is used for each rescue unit. The binary constraints (4) guarantee that each schedule is either fully used or not used (no fractional usage of schedules).

Algorithm 1 Branch-and-price algorithm for DRSP.

1: solve linear relaxation of root node (BinLP) using column generation
2: initialize set of active nodes
3: **repeat**
4: select an active node for branching
5: branch on selected node by constructing two child nodes
6: solve child nodes' linear relaxations using column generation
7: update set of active nodes based on new information
8: **until** set of active nodes is empty

4. Branch-and-price algorithm for DRSP

In this section, we develop an exact branch-and-price (b&p) algorithm for solving DRSP instances. A b&p algorithm, which has originally been conceptualized by Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance (1998), is a specific form of a branch-and-bound (b&b) algorithm in which all linear relaxations are solved using column generation. This, in turn, was originally introduced by Dantzig and Wolfe (1960) in the context of Dantzig-Wolfe decomposition. The macro structure of our b&p algorithm is presented in Algorithm 1. Lines 2, 3, 7, and 8 occur in every b&b algorithm and do not require further explanation. All other lines are clarified in detail in the remainder of this section.

4.1. Solving the linear relaxation of the root node

First, we present a method to solve the linear relaxation of the root node (line 1 in Algorithm 1). When solving a DRSP instance, the root node of the b&b tree is given by model (BinLP). Consequently, when we relax the binary constraints (4) to $0 \leq x^k_\omega \leq 1$ for all units k and schedules $\omega \in \Omega^k$, the root node's linear relaxation is given as follows:

$$\min \sum_{k=1}^{m} \sum_{\omega \in \Omega^k} c^k_\omega \cdot x^k_\omega \tag{BinLP – LR}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{\omega \in \Omega^k} cap_{kq} \cdot a_{j\omega} \cdot x_\omega^k \geq req_{jq} \qquad \forall j = 1, \ldots, n; q = 1, \ldots, r \tag{5}$$

$$\sum_{\omega \in \Omega^k} x_\omega^k = 1 \qquad \forall k = 1, \ldots, m \tag{6}$$

$$x_\omega^k \geq 0 \qquad \forall k = 1, \ldots, m; \omega \in \Omega^k \tag{7}$$

The restriction $x_\omega^k \leq 1$ is already implied by the combination of (6) and (7). For solving (BinLP-LR), we use column generation, which is applied in general to solve continuous LPs with a large number of variables and a small number of constraints. The idea behind column generation is to solve a series of restricted LPs instead of the large original LP. First, an initial restricted LP is solved in which only a small feasible subset of variables (also called columns) is considered. Based on this solution, columns with negative reduced costs are added to the restricted LP before it is solved again. This is repeated until no more columns with negative reduced costs exist, which implies that the optimal solution of the current restricted LP is also optimal for the original LP with all remaining variables set to zero (Lübbecke & Desrosiers, 2005). In the following, we apply column generation to solve (BinLP-LR) by specifying (i) the set of variables considered in the initial restricted LP and (ii) a method on how to find variables with negative reduced costs.

The set of variables for the initial restricted LP is obtained by a solution heuristic for DRSP. We use the SCHED algorithm suggested by Schryen et al. (2015). Further, let (π, σ) denote the optimal dual solution of a restricted LP, i.e., π_{jq} is the dual variable corresponding to the pair (j, q) in constraint (5) and σ_k is the dual variable corresponding to unit k in constraint (6). The reduced cost of a variable x_ω^k with respect to the optimal dual solution of the restricted LP is defined as follows:

$$r_\omega^k := c_\omega^k - \sum_{j=1}^{n} \sum_{q=1}^{r} cap_{kq} \cdot a_{j\omega} \cdot \pi_{jq} - \sigma_k. \tag{8}$$

Finding variables with least reduced costs is equivalent to solving the so called *pricing problem*

$$r^* := \min_{k=1,\ldots,m} \min_{\omega \in \Omega^k} c_\omega^k - \sum_{j=1}^{n} \sum_{q=1}^{r} cap_{kq} \cdot a_{j\omega} \cdot \pi_{jq} - \sigma_k. \tag{PP}$$

We adapt a dynamic programming algorithm – originally formulated by Lopes and de Carvalho (2007) in a machine scheduling context – in order to obtain a solution for the pricing problem. A detailed description is provided in Appendix B. The algorithm requires two assumptions: (i) travel times must fulfill the triangle inequality, i.e., $s_{i_1 i_2}^k \leq s_{i_1 i_3}^k + s_{i_3 i_2}^k$ for all incidents i_1, i_2, i_3 and rescue units k and (ii) schedules must be allowed to contain incidents multiple times. The triangle inequality for travel times can be guaranteed by viewing travel times s_{ij}^k as the time which unit k requires for traveling along its shortest time path between the locations of i and j. To allow schedules to contain incidents multiple times, we enlarge the sets Ω^k in both (BinLP) and (BinLP-LR) accordingly. Consequently, $a_{j\omega}$ is not binary anymore. In order to keep the sets Ω^k finite, we restrict the maximum makespan of a schedule to $n \cdot (\max_{i,j,k} s_{ij}^k + \max_{j,k} p_j^k)$ which guarantees that the optimal solution of (BinLP) is still contained in the sets Ω^k. These modifications are only required for the dynamic programming algorithm to work and they do not affect the optimal solution of (BinLP). The reason is that the triangle inequality for travel times assures that in an optimal solution for (BinLP), each incident is processed at most once by the same unit since a schedule $\omega \in \Omega^k$ that processes an incident multiple times has always a higher weighted sum of completion times c_ω^k than the schedule resulting when all duplicates are removed.

4.2. Node selection and branching strategy

In the following, we explain our strategy for selecting an active node to branch on (line 4 of Algorithm 1). We use a hybrid strategy whose two elements are last-in-first-out (LIFO) and best-lower-bound-first (BLBF). A LIFO strategy selects the active node for branching that has been created most recently. A BLBF strategy selects the active node with the lowest optimal solution of its linear relaxation for branching.

At the beginning, we use the LIFO search strategy, which is suitable for finding a good feasible solution for the current problem instance early. This corresponds to a b&b node having an integer optimal solution for its linear relaxation. After having found such a feasible solution for the current problem instance, we switch to the BLBF search strategy, which is most suitable for finding an optimal solution and for finally proving its optimality.

For branching on a selected node (line 5 of Algorithm 1), we use so called flow variables X_{ij}^k, as this is common for b&p algorithms in unrelated parallel machine scheduling (Lopes et al., 2014; Lopes & de Carvalho, 2007). These variables are defined as

$$X_{ij}^k = \sum_{\omega \in \Omega^k} \delta_{ij\omega} \cdot x_\omega^k \tag{9}$$

for every pair of incidents $i = 0, \ldots, n$ and $j = 1, \ldots, n$ and every unit $k = 1, \ldots, m$, where $\{x_\omega^k | k = 1, \ldots, m; \omega \in \Omega^k\}$ is an optimal solution of the selected node's linear relaxation. The integer parameter $\delta_{ij\omega}$ indicates how often the sequence $i \to j$ is contained in schedule ω (the sequence $0 \to j$ translates to *j is the first incident in ω*). If x_ω^k is binary for all units k and schedules $\omega \in \Omega^k$, then X_{ij}^k indicates how often incident i is processed directly before incident j by unit k.[2] We branch along the edge (i^*, j^*, k^*) where (i) $X_{i^* j^*}^{k^*}$ is closest to 0.5, i.e.,

$$(i^*, j^*, k^*) = \arg\min_{i,j,k} |X_{ij}^k - 0.5|, \tag{10}$$

and (ii) (i^*, j^*, k^*) has not been used for any branching leading to the current node.

For branching along the edge (i^*, j^*, k^*), we introduce node-specific sets P_j^k of possible predecessors for all incidents j and units k. At the root node, we initialize $P_j^k = \emptyset$ if k does not have any capabilities required by j (i.e., $k \notin M_j$). Otherwise, we set P_j^k as the set of all incidents that require one of the capabilities of unit k and add the artificial incident 0. Constructing two child nodes from the currently selected node is then conducted by modifying the predecessor sets P_j^k of the currently selected node for all incidents j and units k, resulting in node-specific predecessor sets P_j^k for the two child nodes.

For the first child node, i^* is simply removed from the possible predecessors of j^* on k^*, i.e., $P_{j^*}^{k^*} -= \{i^*\}$. This implies that i^* cannot be processed directly before j^* on k^* anymore. For the second child node, i^* is set to be the only possible predecessor of j^* on k^* and is removed from all other predecessor sets on k^*, i.e., we set $P_{j^*}^{k^*} = \{i^*\}$ and $P_j^{k^*} -= \{i^*\}$ for all $j \neq j^*$. In addition, we set $P_j^k -= \{j^*\}$ for all units $k \neq k^*$ which cannot serve any requirement of j^* that k^* cannot already serve. The latter applies analogously to i^* if $i^* \neq 0$.

Setting $\Omega^k = \{\omega = (j_1, \ldots, j_h) | j_{l-1} \in P_{j_l}^k \text{ for all } 1 < l \leq h\}$ for each child node, however, is not sufficient to force i^* to be processed directly before j^* on k^* in all schedules of the second child

[2] Processing incident 0 directly before incident j on unit k means that *incident j is processed first on unit k.*

node, since the presence of multiple capabilities per rescue unit may still enable the processing of all requirements of j^* by units $k \neq k^*$. To circumvent this issue, we introduce for all $k = 1, \ldots, m$ the sets $\mathcal{E}^k$ of all edges (i, j) with the property that the current node-to-construct (either first or second child node) emanates from constructing the second child node along the edge (i, j, k) at some point of its branching history. Then we define

$$\Omega^k = \{\omega = (j_1, \ldots, j_h) \mid j_{l-1} \in P^k_{j_l} \text{ for all } 1 < l \leq h \text{ and } i \to j \text{ is included in } \omega \text{ for all } (i, j) \in \mathcal{E}^k\}. \quad (11)$$

for all $k = 1, \ldots, m$. In particular, the sequence $i^* \to j^*$ is forced to be contained in every feasible schedule on k^* in the second child node. Conclusively, this guarantees $X^{k^*}_{i^* j^*} = 0$ on the first child node and $X^{k^*}_{i^* j^*} \geq 1$ on the second child node.

4.3. Solving child nodes' linear relaxations

As we have seen in the previous subsection, all nodes of the b&b tree are of the form (BinLP) – only differing in node-specific sets Ω^k. Consequently, all linear relaxations are of the form (BinLP-LR). Therefore, the column generation procedure to solve an arbitrary node's linear relaxation (line 6 of Algorithm 1) is similar to the procedure presented in Section 4.1, hence we only outline the differences.

The initial set of variables for the restricted LP is obtained by taking all variables from the final restricted LP of the parent node and penalizing those columns that are not feasible anymore due to branching (i.e., setting $c^k_\omega = \infty$ if $\omega \notin \Omega^k$).

We also need to take incomplete schedules into consideration. These incomplete schedules cannot be filtered out during column generation since the dynamic programming algorithm (see Appendix B for details) cannot be detained from constructing them. However, we can ignore them by not adding them to the current restricted LP. Consequently, an optimal solution of model (BinLP-LR) is found when all columns with negative reduced costs are incomplete. This completes the description of Algorithm 1. We prove the following result about its exactness in Appendix C.

Theorem 1. *The presented branch-and-price algorithm is an exact procedure for solving DRSP instances.*

5. Computational experiments for the b&p algorithm

In this section, we evaluate the execution times of the suggested b&p algorithm and determine the improvement of the b&p solutions over the solutions generated by the SCHED heuristic suggested by Schryen et al. (2015). Our hardware setup is an Intel Westmere X5675 CPU with a clock frequency of 3.07 gigahertz and 96 gigabytes RAM. We coded the algorithm in C++ on Linux CentOS 7.3. For the solution of the restricted LPs during column generation, we used Gurobi 7.

5.1. Data generation

To reflect the diversity of real-world disasters, we generate instances for four different scenarios. First, we discriminate between situations in which rescue units are either specialized, i.e., they have a low number of capabilities, or in which they are non-specialized, i.e., the number of capabilities per rescue unit is high. We distinguish between eight different capabilities (based on interviews with practitioners) that are listed in Table 2. Second, we differentiate between situations in which travel times are low compared to processing times (low travel intensity) or high compared to processing times (high travel intensity). There are several factors that influence travel intensity, which can vary substantially between different disasters. These factors include distances between locations of incidents, traffic density and congestions, or the difficulty of (and thereby time required for) processing incidents. For example, disasters in urban and rural areas may differ substantially in these regards. Combining the two dimensions described above, we yield four different scenarios, which account for diversity regarding both external factors (e.g., traffic conditions) and internal factors (e.g., specialization of rescue units).

For each of the four scenarios, we investigate different instance sizes in which the number of incidents n and rescue units m varies between 10 and 40 (with $m \leq n$ since resources are scarce in disaster situations). This range is realistic in real world disasters for two reasons (Schryen et al., 2015). First, there are multiple disaster operations centers (DOCs) in a large-scale disaster and this decentralized structure implies moderate numbers of rescue units that have to be scheduled by each DOC. Second, real-world disasters are highly dynamic situations. New (requirements of) incidents can occur, some (requirements of) incidents may already have been processed successfully, and available rescue units and their capabilities are likely to change over time. These dynamics can be considered by solving a sequence of different small- or medium-sized problem instances instead of one single large problem instance. A more detailed discussion is provided in Appendix D.

For each of the four scenarios, we also vary the probability p_{req} with which a particular incident requires a specific capability between 10% and 30%. We distinguish between the eight capabilities presented in Table 2. Applying these probability values leads to probabilities between 32.8% and 79.0% that a specific incident has at least two requirements, which makes (loose) collaboration necessary. Details are presented in Appendix E.

For each instance size and each requirement probability in each of the four scenarios, we randomly generate ten different instances. The details of our data generation process are presented in Table 2; while scenario-independent parameters are listed in the upper part, scenario-specific parameters are contained in the lower part of the table. The severity level of an incident is a uniformly drawn integer between 1 and 5 according to the five-step scale *low, guarded, elevated, high, and severe* of the former U.S. Homeland Security Advisory System (Behunin, 2004). The processing times are drawn from a normal distribution with mean value 100 and a standard deviation of 50. The high coefficient of variation (0.5) accounts for processing times that vary substantially in chaotic disaster situations. The processing times are rounded to integers for algorithmic reasons (cf. dynamic programming algorithm in Appendix B). This integer requirement does not affect the applicability of our approach, since times are prone to estimations and therefore not precise in disaster response.[3] To this point, all parameters are scenario-independent.

For each unit, the probability p_{cap} of having a specific capability is scenario-specific: We use $p_{cap} = 20\%$ when we investigate specialized rescue units, which leads to slightly less than two capabilities per rescue unit on average (the theoretical mean is approximately 1.92 based on the formulas in Appendix E). To model non-specialized rescue units, we doubled the probability to $p_{cap} = 40\%$, which rises the average number of capabilities per rescue unit to more than three (the theoretical mean is approximately 3.25 based on the formulas in Appendix E). This substantial increase enables us to gain insights into the effect of unit specialization on the performance of our b&p algorithm.

Finally, we explain the scenario-specific generation of travel times, the intention of which is to cover both low and high travel intensities. To accomplish realistic travel times, their generation is

[3] The mean value of the processing times (and therefore implicitly their precision) cannot be increased arbitrarily since the execution time of the dynamic programming algorithm depends on upper bounds for the makespan of feasible solutions for DRSP. This makespan increases with increasing processing times.

Table 2
Details of data generation.

Input parameter	Value, range, distribution
Number of incidents	$n \in \{10, 20, 30, 40\}$
Number of rescue units	$m \in \{10, 20, 30, 40\}$, $m \leq n$
Number of capabilities/requirements	8 (policemen, fire brigades, paramedics, search and rescue, debris removal, infrastructure preservation, logistics teams, special casualty access teams)
Probability of having a particular requirement	$p_{req} \in \{10\%, 15\%, 20\%, 25\%, 30\%\}$
Number of instances per p_{req} and instance size	10
Severity levels	$w_j \sim U(1, 5, 1)$
Processing times	$p_j^k \sim N(100, 50)$
Grid size for incident positioning	100×100
Speed of rescue units	$speed_k \sim U(8, 16, 1)$
Scenario *specialized/low intensity*	
Probability of having a particular capability	$p_{cap} = 20\%$
Travel intensity factor	$TIF = 1.0$
Scenario *specialized/high intensity*	
Probability of having a particular capability	$p_{cap} = 20\%$
Travel intensity factor	$TIF = 4.25$
Scenario *non-specialized/low intensity*	
Probability of having a particular capability	$p_{cap} = 40\%$
Travel intensity factor	$TIF = 1.0$
Scenario *non-specialized/high intensity*	
Probability of having a particular capability	$p_{cap} = 40\%$
Travel intensity factor	$TIF = 4.25$

based on a coordinate grid which represents a fictitious discretized version of real maps in disaster applications. For each pair of incidents (i, j) on the grid, the travel time of each unit k between the positions of incidents i and j can be calculated as the distance between i and j divided by the speed of unit k. In our data generation, each incident is placed on a 100×100 grid by uniformly drawing its x and y coordinates. After that, the euclidean distance between all pairs of incidents is calculated and then divided by the speed of unit k (which is a uniformly drawn integer between 8 and 16). This time is further scaled with a travel intensity factor of $TIF = 1.0$ for low travel intensity and $TIF = 4.25$ for high travel intensity and finally rounded up to obtain integer values for s_{ij}^k. The grid size, speed distributions, and travel intensity factors are selected in a way that this results in expected travel times of 5.1 for low travel intensity and 19.9 for high travel intensity.[4] Since the processing time distribution has a fixed mean of 100, this leads to a high ratio of mean processing times to mean travel times for low travel intensity (approx. 20) and a low corresponding ratio for high travel intensity (approx. 5).

In total, we generate and solve 2000 instances (four scenarios, ten instance sizes and five requirement probabilities per scenario, and ten instances per instance size and requirement probability).

5.2. Results

The results for our computational experiments are presented in this section. Fig. 3 displays the average execution times of our b&p algorithm for $p_{req} = 20\%$.[5] The corresponding average execution times before a first integer solution is found show a similar pattern and can be obtained from Tables F.13–F.16 in Appendix F. These execution times are important for practitioners when due to time pressure the b&p algorithm cannot be executed completely but is terminated once a feasible solution for the current DRSP instance (referred to as *first integer (FI) solution*) is found.[6] In this case, the b&p algorithm serves as a heuristic. The execution times for both the exact and heuristic version of our b&p algorithm are also reflected in the number of nodes that are explored during the algorithm (see Tables F.13–F.16 in Appendix F).

Further, we compare the objective values of the SCHED heuristic (suggested by Schryen et al., 2015) to the objective values of both the optimal solution and the FI solution found by the b&p algorithm. This comparison allows us to identify the levels of improvements over the SCHED heuristic achieved when the b&p algorithm is executed as an exact or as a heuristic solution procedure. Fig. 4 displays the average ratios of objective values of the SCHED solution to objective values of the optimal solution for $p_{req} = 20\%$.[7] For example, a ratio of 1.309 indicates that the objective value of the SCHED solution exceeds the objective value of the optimal solution by 30.9% on average. The respective ratios for the excess of SCHED solutions over the FI solutions show a similar pattern and can be obtained from Tables F.13–F.16 in Appendix F. Detailed statistics on all results presented in the figures can also be retrieved from these tables.

In order to measure the effect sizes of exogenous parameter values (including requirement probability) on the execution time of the branch-and-price algorithm, we conducted a sensitivity analysis, using the following regression model with a logarithmically transformed dependent execution time variable:

$$ln(EXEC\_TIME) = \beta_0 + \beta_1 \cdot n + \beta_2 \cdot \frac{n}{m} + \beta_3 \cdot p_{req} + \beta_4 \cdot p_{cap} + \beta_5 \cdot TIF + \varepsilon \quad (12)$$

Table 3 shows the results of the regression, which are based on data provided in Tables F.5–F.24 in Appendix F. From the original 2000 instances, we excluded those 26 instances where no optimal

[4] The expected travel times are calculated via enumerating all combinations of incidents pairs (i, j) with $i \neq j$ and unit speeds $speed_k \in \{8, 9, \ldots, 16\}$. All of these combinations are equally likely to be generated.

[5] Due to space limitation, we present results only for $p_{req} = 20\%$. Results for values other than 20% are shown in Appendix F.

[6] It needs to be noted that the SCHED heuristic is used to find a set of feasible columns for the initial restricted LP during column generation at the root node (cf. Section 4.1). Although this SCHED solution is feasible for DRSP, we do not refer to it as FI solution. Using the term FI solution, we rather refer to the first feasible solution that is obtained by further exploring the b&b tree. Such a feasible solution is found whenever the linear relaxation of a b&b node has an integer optimal solution.

[7] Due to space limitation, we present results only for $p_{req} = 20\%$. Results for values other than 20% are shown in Appendix F.

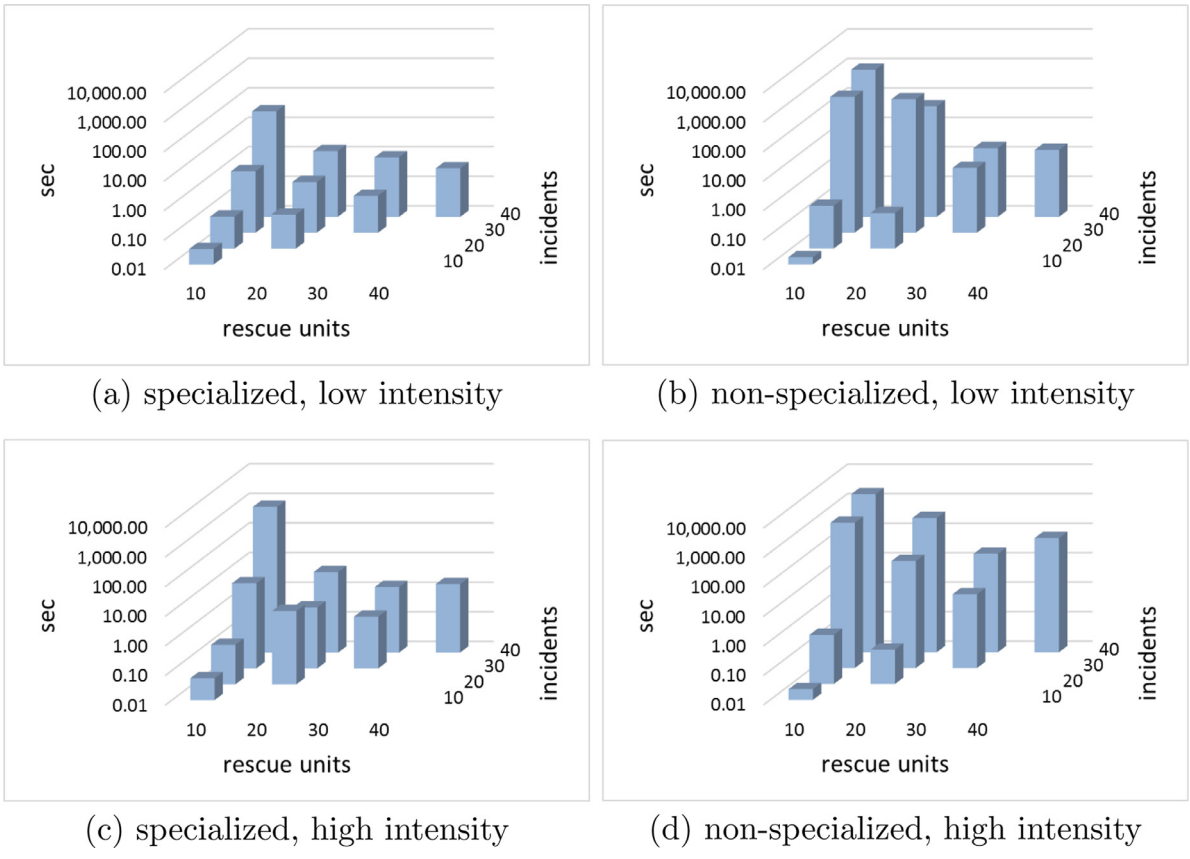

Fig. 3. Average execution times of the b&p algorithm.

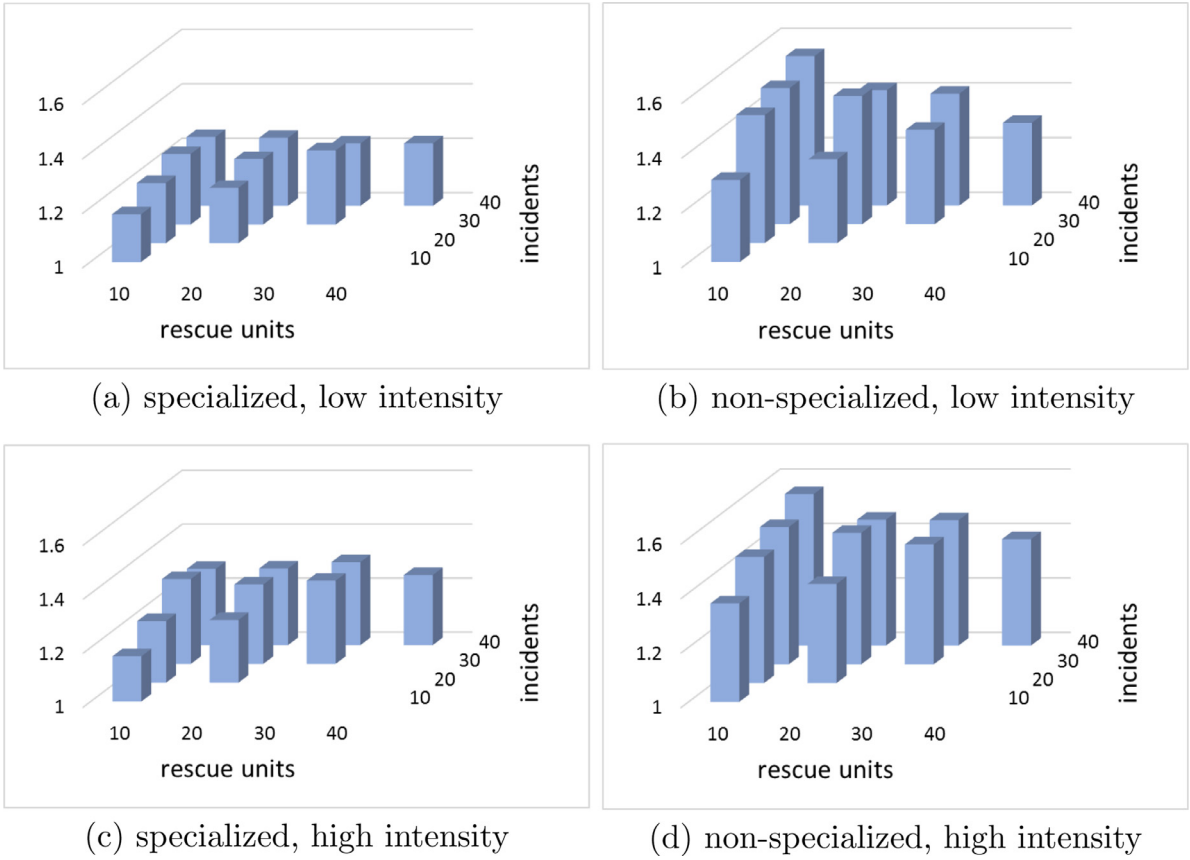

Fig. 4. Average ratio of SCHED objective value to optimal objective value.

Table 3
Results for regression on execution time.

Effect	Estimate (Std. error)	t value (Significance)
Number of incidents (n)	0.13 (0.00)	35.38 ***
Ratio of incidents to units (n/m)	0.88 (0.04)	21.82 ***
Requirement probability (p_{req})	15.08 (0.49)	30.99 ***
Capability probability (p_{cap})	2.80 (0.34)	8.21 ***
Travel intensity factor (TIF)	0.17 (0.02)	8.25 ***
N	1875	
R squared	0.64	

Notes. Model includes an intercept. ***significant at 0.1%.

Table 4
Results for regression on ratio SCHED/OPT (values of objective function).

Effect	Estimate (Std. error)	t value (Significance)
Number of incidents (n)	0.00 (0.00)	10.71 ***
Ratio of incidents to units (n/m)	0.02 (0.00)	6.35 ***
Requirement probability (p_{req})	1.11 (0.04)	30.60 ***
Capability probability (p_{cap})	0.67 (0.03)	26.57 ***
Travel intensity factor (TIF)	0.01 (0.00)	6.92 ***
N	1875	
R squared	0.49	

Notes. Model includes an intercept. ***significant at 0.1%.

solution has been calculated after 48 hours. Furthermore, we removed outliers (upper 5%) in order to avoid skewed regression coefficients.

The same type of regression was conducted with the time to find an FI solution as the dependent variable. The results are very similar and can be obtained from Table G.25 in Appendix G.

We also conducted a sensitivity analysis with the ratio $SCHED/OPT$ of the objective value of the SCHED solution to the objective value of the optimal solution. Here, we used the following linear regression model:

$$SCHED/OPT = \beta_0 + \beta_1 \cdot n + \beta_2 \cdot \frac{n}{m} + \beta_3 \cdot p_{req} + \beta_4 \cdot p_{cap} + \beta_5 \cdot TIF + \varepsilon \quad (13)$$

Table 4 presents the results of the regression. We again removed outliers as described above.

The results for the same type of regression on the ratio $SCHED/FI$ of the objective values of the SCHED solutions to the objective values of the FI solutions are again very similar and can be obtained from Table G.26 in Appendix G.

6. Discussion

We discuss the results of our computational experiments in this section. We analyze the efficiency of the b&p algorithm before we interpret its effectiveness. Both subsections begin with a detailed discussion of the results for $p_{req} = 20\%$ before we interpret the results of our sensitivity analysis in terms of effect sizes of exogenous variables, including requirement probabilities, and make predictions on the performance of our b&p algorithm for variations in input data. Finally, we discuss managerial implications of our experiments in a separate subsection.

6.1. Efficiency of the b&p algorithm

Discussion of results. Fig. 3 shows that the average execution time of our b&p algorithm varies between zero seconds and approximately 40 minutes. When a scenario and an instance size is fixed, execution times for the ten randomly generated instances can be volatile with some coefficients of variation being close to 3.0, see Tables F.13–F.16. Within each scenario, the execution times mainly depend on two factors. First, when the number of incidents or rescue units is fixed, the execution time tends to rise with an increasing ratio $\frac{n}{m}$ of incidents to rescue units. Using a logarithmic scale on the y-axis of Fig. 3, we can see that even small changes in this ratio can cause an exponential increase in execution times. For example, in the scenario with $n = 40$ non-specialized rescue units in low travel intensity situations, there is an average execution time of approximately two seconds when $m \in \{30, 40\}$, which increases to 56 seconds when $m = 20$ and to even 988 seconds when m is reduced to 10. This increase of execution times is rooted in the expanded workloads of the rescue units as well as in the resulting challenges to not only assign incidents to rescue units but also to schedule the incidents that are assigned to a particular rescue unit.

Second, when the ratio $\frac{n}{m}$ is fixed, the execution times tend to increase with an ascending number of incidents. For example, when rising the instance size from $n = 20$ and $m = 10$ to $n = 40$ and $m = 20$, the logarithmic scale shows that the increase in execution time is up to almost three magnitudes ($\cdot 10^3$); for example, it increases from 0.5 seconds to 356 seconds in the scenario with non-specialized rescue units and high travel intensity. This effect in not surprising as the solution space expands with increasing values of n.

Comparing the four scenarios, execution times for scenarios with specialized rescue units tend to be smaller than for scenarios with non-specialized rescue units (when low/high travel intensity is fixed). This difference can be more than two magnitudes. For example, for $n = 30$ incidents and $m = 10$ rescue units in a low travel intensity situation, the execution times rise from 1.2 seconds for specialized rescue units to 407 seconds for non-specialized rescue units. Facing high travel intensity, the respective execution times rise from 7.4 seconds to 848 seconds . This is a result of having more feasible allocations and therefore a larger feasible solution space when rescue units have more capabilities. Furthermore, execution times for high travel intensity scenarios tend to be higher than for low travel intensity scenarios (when specialized/non-specialized unit setting is fixed). This increment can be more than one magnitude. In the situation with $n = 40$ incidents and $m = 10$ specialized rescue units, for example, we have execution times of 38 seconds when there is low travel intensity and 828 seconds when there is high travel intensity. The execution times for $m = 40$ non-specialized rescue units and $n = 40$ incidents rise from 1.9 seconds when there is low travel intensity to 74 seconds when there is high travel intensity. This is caused by a less effective bounding since the number of nodes that are explored during the b&p algorithm increases correspondingly (see Tables F.13–F.16).

The execution times until an FI solution is found are substantially lower than the execution times of the entire b&p algorithm (see Tables F.13–F.16). These times are especially important for practitioners when the time for decision making is scarce. Over all scenarios and instance sizes, the highest execution time (averaged over ten instances) to find an FI solution find an FI solution is 34.2 seconds. Although being substantially lower, the execution times to find an FI solution follow the same patterns regarding the influence of the instance size as described above for the execution times for finding an optimal solution. However, execution times for finding an FI solution are independent of the specific scenario.

Sensitivity analysis. To analyze the sensitivity of execution times on changes in exogenous parameters and to make predictions for variations in input data, we use the results presented in Table 3. The regression model has a good fit of $R^2 = 0.64$, which means that 64% of the variance in execution times can be explained by the five exogenous parameters that we include as independent variables in our regression model (12). Furthermore, the effect sizes of all independent variables are highly significant.

When interpreting effect sizes of parameters on execution times, the logarithmic scale of execution times in the regression needs to be considered. For example, when the size of n is increased by 30 (e.g., by moving from the setting $n = m = 10$ to the setting $n = m = 40$), then the natural logarithm of the execution time increases by $30 \cdot 0.13$; i.e., the execution time increases with the factor of $e^{30 \cdot 0.13} \approx 49.4$. When the ratio of incidents to units is increased by 3 (e.g., by moving from the setting $n = m = 40$ to the setting $n = 40,\ m = 10$), then the execution time increases with the factor of $e^{3 \cdot 0.88} \approx 14.0$. It should be noticed that, when moving from the setting $n = m = 10$ to the setting $n = 40,\ m = 10$, the two effects discussed above occur contemporaneously; i.e., the execution time is approximately increased by the products of both factors (≈ 692).

Regarding the probability of having a particular requirement (p_{req}), we used values that differ by five percent points (10%, 15%, 20%, 25%, 30%). The regression results indicate that an increase by 5 percent points leads to an increase of execution time by the factor $e^{0.05 \cdot 15.08} \approx 2.1$; i.e., the execution time is approximately doubled.

The regression results also allow to compare the effects sizes of the level of specialization of rescue units and the travel intensity of the overall situation, both of which determine the type of scenario. The level of specialization is operationalized by the probability with which a specific rescue unit has a particular capability. We used two values ($p_{cap} = 20\%$ and $p_{cap} = 40\%$), and the impact on execution times from increasing p_{cap} is the factor $e^{0.2 \cdot 2.8} \approx 1.8$; i.e., execution times almost double when rescue units change their characteristics from *specialized* to *non-specialized*. Regarding travel intensity, the impact of changing travel intensity from *low* to *high* is given by the factor $e^{(4.25 - -1) \cdot 0.17} \approx 1.7$; i.e., the execution time is again almost doubled. However, it should be noted that we used, for both the level of specialization of rescue units and the travel intensity, only two values each so that the regression coefficients should be interpreted with caution.

For those variables where we have investigated more than two different values in our computational experiments (i.e., number of incidents n, ratio of incidents to units $\frac{n}{m}$, and requirement probability p_{req}), the results of the regression can be used to make predictions of execution times which are outside our computational scope. We give an example for the variable p_{req}. When we increase the requirement probability from $p_{req} = 30\%$ to $p_{req} = 40\%$, which is an increase by 0.1, we can expect the average execution times (ceteris paribus) to be increased by the factor $e^{0.1 \cdot 15.08} \approx 4.5$. When we fix the scenario to *specialized rescue units and low travel intensity* and the instance size to $n = 40$ and $m = 20$, for example, then we can expect the average execution time to rise from 98.1 seconds for $p_{req} = 30\%$ (this value can be obtained from Table F.21) to $4.5 \cdot 98.1\text{seconds} \approx 441s$ for $p_{req} = 40\%$.

We also conducted a regression on the time to find an FI solution, the results for which are presented in Table G.25. It shows that the times to find an FI solution depend on changes in exogenous model parameters in a similar way than the execution times of the entire b&p algorithm – with the exception that the specialization of rescue units has no significant influence.

6.2. Effectiveness of the b&p algorithm

In this subsection, we discuss the improvements of our exact b&p algorithm over the heuristic SCHED suggested by Schryen et al. (2015). We also discuss the quality of solutions when our b&p algorithm is executed as a heuristic by terminating upon finding an FI solution.

Discussion of results. From Fig. 4, we can see that the SCHED objective values exceed the optimal objective values obtained by our b&p algorithm by between 16.7% and 55.6% on average with low coefficients of variation; i.e., for fixed instance sizes and scenarios the levels of improvements are robust over instances. This shows that the solutions returned by our b&p algorithm substantially improve the solutions returned by the SCHED heuristic in all tested scenarios. When rescue units are specialized, the average excess of the SCHED objective values over the optimal objective values is almost constant, especially when $n > 20$. In scenarios with non-specialized rescue units, the average excess is higher and more volatile, i.e., it depends on instance sizes. The reason for both the higher excess and volatility is the fact that an increasing number of feasible allocations in non-specialized scenarios makes the solution space larger. Therefore, the heuristic approach of SCHED becomes less effective. It is also notable that the average SCHED excess is almost the same for high travel intensity and for low travel intensity.

The average ratios *SCHED*/*FI* of the objective value of the SCHED solution to the objective value of the FI solution show the same patterns regarding the influence of the instance size and the specific scenario on *SCHED*/*FI* as described above for the ratios *SCHED*/*OPT*, see Tables F.13-F.16 for details. Furthermore, the average ratios *SCHED*/*FI* and *SCHED*/*OPT* almost coincide when a particular scenario, instance size, and requirement probability are fixed. This implies that the FI solution is highly effective. Indeed, the average ratio *FI*/*OPT* of the objective value of the FI solution to the optimal objective value is between 0.0% and 6.5% with low coefficients of variation over instances of the same size and scenario, see Tables F.13-F.16 for details. Furthermore, the ratio *FI*/*OPT* was less than 5% in all but 20 instances (out of 400). In the most difficult single instance with an execution time of almost four hours, the FI solution exceeds the objective value of the optimal solution by only 1.6% and was found after 17.8 seconds, which is a small fraction of the full execution time.

Sensitivity analysis. The results of the sensitivity analysis regarding the effect sizes of our parameters on the extent with which our algorithm improves the objective values obtained from applying the heuristic suggested by Schryen et al. (2015) (i.e., the ratio *SCHED*/*OPT*) can be interpreted analogously to our analysis of execution times. As Table 4 reveals, considerable effects only occur regarding the requirement probability p_{req} and the capability probability p_{cap}; i.e., relative improvements of objective values achieved through our algorithm considerably increase only when the probability of having a specific requirement increases or when rescue units become more/less specialized. The regression on the average ratios *SCHED*/*FI* shows very similar results (for details see Table G.26) and are therefore not further discussed.

6.3. Managerial implications

Our computational experiments are of high relevance to the disaster operations management of rescue organizations when they need to assign and schedule their collaborating rescue units to emerging incidents under time pressure in order to reduce the overall resulting harm. According to our interviews with managers of rescue organizations, they need to make their decisions during ten minutes.

Our results show that for almost 94% of the 2000 tested problem instances, our b&p algorithm provides optimal solutions during the requested time window of ten minutes and substantially outperforms a heuristic suggested in the literature in terms of the resulting harm in almost all instances. Furthermore, first feasible solutions are found by the b&p algorithm in even considerably less time, with only five out of 2000 instances remaining without a first feasible solution during ten minutes. In almost all instances these

first feasible solutions are competitive to the optimum. These computational results make our b&p algorithm highly appealing for use in decision support systems for command & control units.

Extensive sensitivity analysis of execution times reveal statistically significant effect sizes of exogenous model parameters. Most influencing is the extent of parameters which determine the intensity of required collaboration of rescue units. The identification of effect sizes allows reliable predictions on execution times of the proposed b&p algorithm when all model parameters are set (explained variance of execution times is about 64%). Such predictions are useful for decision makers in the command & control board when due to time pressure they need to decide on whether and when the b&p algorithm should be aborted, accepting the best found solution so far.

7. Conclusion

In this paper, we address a challenge that occurs during the response phase of disaster operations management. In this phase, rescue organizations have to assign and schedule their rescue units to emerging incidents under time pressure in order to reduce the overall resulting harm. We refer to this problem as the *Disaster Response Scheduling Problem* (DRSP). We account for the practical need that the processing of incidents requires different capabilities, thereby making (loose) collaboration of rescue units necessary. We contribute to both modeling and solving this problem by (1) conceptualizing the situation as a generalization of a parallel machine scheduling problem, (2) modeling DRSP as a binary linear minimization problem, (3) suggesting a branch-and-price algorithm, which can serve as both an exact and heuristic solution procedure, and (4) conducting computational experiments – including a sensitivity analysis of the effects of exogenous model parameters on execution times and objective value improvements over a heuristic suggested in the literature – for different practical disaster scenarios.

The results of our computational experiments show that most problem instances of practically feasible size (number of incidents and number of rescue units are not larger than 40) can be solved to optimality in less than ten minutes. The optimal solutions substantially improve solutions found by the SCHED heuristic by Schryen et al. (2015), which is the best DRSP heuristic that we could find in the literature, in terms of weighted sum of completion times, which can be seen as a proxy for the overall harm in disaster situations. When time is scarce and decision makers have to coordinate rescue units before the algorithm terminates, they can abort the execution and rely on the best found integer solution, which is always feasible for DRSP. Even the first found integer solution is competitive to the optimal solution in terms of objective value and substantially better than the SCHED solution in almost all instances. A first integer solution was found within ten minutes in all but five instances. This makes our algorithm not only applicable in practice but also superior to existing algorithms in terms of harm reduction.

Since DRSP is a very general scheduling problem, our b&p algorithm can also be applied to a variety of more specialized scheduling problems, including non-preemptive scheduling on unrelated parallel machines with sequence-dependent setup times and a weighted sum of completion times as objective function ($R/s_{ij}/\sum w_j C_j$) and the Rescue Unit Assignment and Scheduling Problem ($R/s_{ijk}/\sum w_j C_j$). For both scheduling problems, only heuristic procedures have been proposed in the literature.

We envision further avenues for research. From a model perspective, preemption can be considered when rescue units may interrupt the processing of incidents. Also, time windows may be integrated in the model. Furthermore, uncertainty of data may be modeled by developing, e.g., stochastic versions of the model. From a validation perspective, our algorithm should be evaluated based on real data, which have not been available to us. From a computational perspective, our branch-and-price algorithm can be parallelized and executed in parallel computing environments, such as high-performance clusters.

Acknowledgments

This research has been funded by the Federal Ministry of Education and Research of Germany in the framework of KUBAS (project number 13N13942).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2018.06.010.

References

Albores, P., & Shaw, D. (2008). Government preparedness: Using simulation to prepare for a terrorist attack. *Computers & Operations Research, 35*(6), 1924–1943. doi:10.1016/j.cor.2006.09.021.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378. doi:10.1016/j.ejor.2015.04.004.

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, 27*(2), 219–239. doi:10.1016/S0305-0483(98)00042-5.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*(3), 985–1032. doi:10.1016/j.ejor.2006.06.060.

Altay, N., & Green, W. G. (2006). OR/MS research in disaster operations management. *European Journal of Operational Research, 175*(1), 475–493. doi:10.1016/j.ejor.2005.05.016.

Arnaout, J.-P., Rabadi, G., & Mun, J. H. (2006). A dynamic heuristic for the stochastic unrelated parallel machine scheduling problem. *International Journal of Operations Research, 3*(2), 136–143.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329. doi:10.1287/opre.46.3.316.

Behunin, S. A. (2004). *Homeland security advisory system*. Naval Postgraduate School, Monterey, California. Ph.D. thesis..

Bodaghi, B., & Ekambaram, P. (2016). An optimization model for scheduling emergency operations with multiple teams. In *Proceedings of the 2016 international conference on industrial engineering and operations management (IEOM)* (pp. 436–442). Detroit, MI, USA

Brucker, P. (2007). *Scheduling algorithms* (5th). Springer Berlin Heidelberg. doi:10.1007/978-3-540-69516-5.

Chen, J.-F. (2015). Unrelated parallel-machine scheduling to minimize total weighted completion time. *Journal of Intelligent Manufacturing, 26*(6), 1099–1112. doi:10.1007/s10845-013-0842-y.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research, 8*(1), 101–111. doi:10.1287/opre.8.1.101.

Fiedrich, F., Gehbauer, F., & Rickers, U. (2000). Optimized resource allocation for emergency response after earthquake disasters. *Safety Science, 35*(1), 41–57. doi:10.1016/S0925-7535(00)00021-7.

Galindo, G., & Batta, R. (2013). Review of recent developments in OR/MS research in disaster operations management. *European Journal of Operational Research, 230*(2), 201–211. doi:10.1016/j.ejor.2013.01.039.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326. doi:10.1016/S0167-5060(08)70356-X.

Green, L. V., & Kolesar, P. J. (2004). Improving emergency responsiveness with management science. *Management Science, 50*(8), 1001–1014. doi:10.1287/mnsc.1040.0253.

IFRC (2013). World disasters report 2013. http://www.ifrc.org/PageFiles/134658/WDR%202013%20complete.pdf (accessed on 25 May 2017)

Kajitani, Y., Chang, S. E., & Tatano, H. (2013). Economic impacts of the 2011 Tohoku-Oki earthquake and tsunami. *Earthquake Spectra, 29*(s1), 457–478. doi:10.1193/1.4000108.

Liberatore, F., Ortuño, M. T., Tirado, G., Vitoriano, B., & Scaparra, M. P. (2014). A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. *Computers & Operations Research, 42*, 3–13. doi:10.1016/j.cor.2012.03.019.

Lodree, E. J., & Taskin, S. (2009). Supply chain planning for hurricane response with wind speed information updates. *Computers & Operations Research, 36*(1), 2–15. doi:10.1016/j.cor.2007.09.003.

Lopes, M., Alvelos, F., & Lopes, H. (2014). Improving branch-and-price for parallel machine scheduling. In *Proceedings of the Forteenth international conference on computational science and its applications (ICCSA)*, Guimaraes, Portugal (pp. 290–300). doi:10.1007/978-3-319-09129-7_22.

Lopes, M. J. P., & de Carvalho, J. M. V. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research, 176*(3), 1508–1527. doi:10.1016/j.ejor.2005.11.001.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research, 53*(6), 1007–1023. doi:10.1287/opre.1050.0234.

Paul, J. A., & Hariharan, G. (2012). Location-allocation planning of stockpiles for effective disaster mitigation. *Annals of Operations Research, 196*(1), 469–490. doi:10.1007/s10479-011-1052-7.

Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems* (5th). Springer International Publishing. doi:10.1007/978-3-319-26580-3.

Rabadi, G. (Ed.). (2016). *Heuristics, meta-heuristics and approximate methods in planning and scheduling* (1st). Springer International Publishing. doi:10.1007/978-3-319-26024-2.

Rauchecker, G., & Schryen, G. (2015). High-performance computing for scheduling decision support: a parallel depth-first search heuristic. In *Proceedings of the Twenty-sixth Australasian conference on information systems*. Adelaide, Australia. Paper 7

Rolland, E., Patterson, R. A., Ward, K., & Dodin, B. (2010). Decision support for disaster management. *Operations Management Research, 3*(1–2), 68–79. doi:10.1007/s12063-010-0028-0.

Sahebjamnia, N., Torabi, S. A., & Mansouri, S. A. (2015). Integrated business continuity and disaster recovery planning: towards organizational resilience. *European Journal of Operational Research, 242*(1), 261–273. doi:10.1016/j.ejor.2014.09.055.

Salmerón, J., & Apte, A. (2010). Stochastic optimization for natural disaster asset prepositioning. *Production and Operations Management, 19*(5), 561–574. doi:10.1111/j.1937-5956.2009.01119.x.

Schryen, G., Rauchecker, G., & Comes, T. (2015). Resource planning in disaster response: aecision support models and methodologies. *Business & Information Systems Engineering, 7*(2), 1–17. doi:10.1007/s12599-015-0381-5.

Simpson, N., & Hancock, P. (2009). Fifty years of operational research and emergency response. *Journal of the Operational Research Society, 60*(1), 126–139. doi:10.1057/jors.2009.3.

Tamura, H., Yamamoto, K., Tomiyama, S., & Hatono, I. (2000). Modeling and analysis of decision making problem for mitigating natural disaster risks. *European Journal of Operational Research, 122*(2), 461–468. doi:10.1016/S0377-2217(99)00247-7.

Tsai, C., & Tseng, C. (2007). Unrelated parallel-machines scheduling with constrained resources and sequence-dependent setup time. In *Proceedings of the thirty–seventh international conference on computers and industrial engineering (CIE)* (pp. 20–23). Alexandria, Egypt

Weng, M. X., Lu, J., & Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics, 70*(3), 215–226. doi:10.1016/S0925-5273(00)00066-9.

Wex, F., Schryen, G., Feuerriegel, S., & Neumann, D. (2014). Emergency response in natural disaster management: Allocation and scheduling of rescue units. *European Journal of Operational Research, 235*(3), 697–708. doi:10.1016/j.ejor.2013.10.029.

Wex, F., Schryen, G., & Neumann, D. (2013). Assignments of collaborative rescue units during emergency response. *International Journal of Information Systems for Crisis Response and Management, 5*(4), 63–80. doi:10.4018/ijiscram.2013100104.