European Journal of Operational Research 235 (2014) 697–708

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

Emergency response in natural disaster management: Allocation and scheduling of rescue units

Felix Wex [a], Guido Schryen [b,*], Stefan Feuerriegel [a], Dirk Neumann [a]

[a] Chair for Information Systems Research, University of Freiburg, Platz der Alten Synagoge, 79098 Freiburg, Germany
[b] Management Information Systems, University of Regensburg, Universitätsstr. 31, 93053 Regensburg, Germany

ARTICLE INFO

Article history:
Received 13 December 2012
Accepted 11 October 2013
Available online 25 October 2013

Keywords:
Decision support systems
Natural Disaster Management (NDM)
Heuristics
Assignment
Scheduling

ABSTRACT

Natural disasters, such as earthquakes, tsunamis and hurricanes, cause tremendous harm each year. In order to reduce casualties and economic losses during the response phase, rescue units must be allocated and scheduled efficiently. As this problem is one of the key issues in emergency response and has been addressed only rarely in literature, this paper develops a corresponding decision support model that minimizes the sum of completion times of incidents weighted by their severity. The presented problem is a generalization of the parallel-machine scheduling problem with unrelated machines, non-batch sequence-dependent setup times and a weighted sum of completion times – thus, it is NP-hard. Using literature on scheduling and routing, we propose and computationally compare several heuristics, including a Monte Carlo-based heuristic, the joint application of 8 construction heuristics and 5 improvement heuristics, and GRASP metaheuristics. Our results show that problem instances (with up to 40 incidents and 40 rescue units) can be solved in less than a second, with results being at most 10.9% up to 33.9% higher than optimal values. Compared to current best practice solutions, the overall harm can be reduced by up to 81.8%.

1. Introduction

Natural disasters, such as earthquakes, tsunamis, floods, hurricanes and volcanic eruptions, have caused tremendous harm in the past and continue to threaten infrastructure and millions of people each year. Of particular importance for the reduction of casualties and economic losses is the response phase in natural disaster management, during which a large number of geographically-dispersed incidents, such as fires and collapsed buildings, require immediate processing by rescue units in the presence of severe resource scarcities and time pressure. Thus, one of the most critical emergency response tasks (Comfort, Ko, & Zagorecki, 2004) is the efficient allocation and scheduling of rescue units. However, this challenge has been addressed in the literature only very rarely.

In this paper, we propose a decision support model for emergency operations centers that allocates available rescue units to emerging incidents and schedules the processing time of these incidents. The model is formulated as a binary quadratic optimization problem, where the objective minimizes the sum of completion times of incidents weighted by their severity. We refer to this problem as the *Rescue Unit Assignment and Scheduling Problem* (RUASP). Our decision problem is related to problems from both routing and scheduling. We show that our problem can be modeled as a (more complex) modification of both the Multiple Traveling Salesman Problem (mTSP) and the parallel-machine scheduling problem with unrelated machines, non-batch sequence-dependent setup times and a weighted sum of completion times as the objective function, classified as $R/ST_{SD}/\sum w_j C_j$ in the scheduling literature. Using this relationship, we prove that our problem is NP-hard.

However, the NP-hardness of the underlying problem opposes one of the imposed requirements that decisions – even in complex emergency situations – must be derived timely. Therefore, we propose, implement and computationally compare several heuristics for the allocation and scheduling of rescue units. More specifically, we use a Monte Carlo-based heuristic as well as joint applications of 8 construction heuristics and 5 improvement heuristics. In addition, we embed these combinations of construction and improvement heuristics into GRASP metaheuristics. Thus, our work contributes not only to the field of disaster management, but also to the optimization literature in general.

The remainder of this paper is structured as follows. Section 2 examines and presents relevant literature and reveals the research gap that our paper addresses. In Section 3, we suggest the RUASP problem and propose an appropriate optimization model. Because of the NP-hardness, Section 4 proposes several solution heuristics. Our computational experiments are presented in Section 5, which also discusses our results. We summarize our results in Section 6, and conclude with an outlook on future research directions.

* Corresponding author. Tel.: +49 941 9435634; fax: +49 941 9435635.
E-mail addresses: felix.wex@is.uni-freiburg.de (F. Wex), guido.schryen@wiwi.uni-regensburg.de (G. Schryen), stefan.feuerriegel@is.uni-freiburg.de (S. Feuerriegel), dirk.neumann@is.uni-freiburg.de (D. Neumann).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.10.029

2. Related work

In the literature on disaster management, challenges and activities are classified (Ajami & Fattahi, 2009; Altay & Green, 2006; IFRC, 2012) into the preparedness phase (period before the disaster), the *response phase* (period during and shortly after the disaster) and the *recovery phase* (period long time after the disaster). More specifically, the preparation phase addresses tasks related to planning, training, early warning (i.e. prediction) and the establishment of necessary emergency services (Gasparini, Manfredi, & Zschau, 2007; Nisha de Silva, 2001; Pollak, Falash, Ingraham, & Gottesman, 2004; Svensson, Holst, Lindquist, & Lindgren, 1996; UN/ISDR, 2005). The primary aims during the response phase are both rescue from immediate danger and stabilization of the condition of survivors. Tasks include relief, emergency shelter and settlement, emergency health, water and sanitation and tracing and restoring family links (IFRC, 2012). In the recovery phase, tasks are related to person finding, (ex-post) data analysis, intelligent infrastructure repair and the provision of various emergency services as well as resources in order to recover the most important infrastructure facilities (GAO, 2006; Salleem et al., 2008; Sherali, Carter, & Hobeika, 1991). According to Chen, Sharman, Rao, and Upadhyaya (2008), these phases are sometimes also arranged in a life cycle.

Regarding decision support, research streams (Airy, Mullen, & Yen, 2009; Comes et al., 2010; Lambert & Patterson, 2002; Reijers, Jansen-Vullers, Zur Muehlen, & Appl, 2007; Tamura, Yamamoto, Tomiyama, & Hatono, 2000) utilize methods from applied statistics and probability theory combined with mathematical programming approaches to establish novel codes of conduct and metrics that assist commanders in critical minutes of the decision-making process. In a first research stream, competitive mechanisms (e.g. auctions) and cooperative mechanisms (e.g. multi-criteria approaches) are developed and, in this context, Fiedrich, Gehbauer, and Rickers (2000) introduce the usage of optimization modeling. Second, another research direction follows guidelines from computational intelligence research (Leifler, 2008; van de Walle & Turoff, 2008) to bridge the gap between information system design principles and decision support process architectures. A third research stream uses empirical investigations of past decision-making conclusions to establish innovative courses of action (Faraj & Xiao, 2006). Fourth, research also focuses on the decision-making process based on either decentralized agents (Airy et al., 2009; Falasca, Zobel, & Fetter, 2009) or a centralized authority.

Researchers argue that distributed coordination (i.e. assignments and schedules) remains independent of failures of a single emergency operations center, communication bottlenecks evolve more seldom and loss minimization is achieved more easily. Regarding the latter, Rolland, Patterson, Ward, and Dodin (2010) promote centralized coordination by applying a mathematical programming model for scheduling distributed rescue units and the assignments of incidents to these. However, the suggested model uses time periods of fixed length, and does not account for the fact that incidents may have different levels of severity. As a remedy, Wex, Schryen, and Neumann (2011, 2012, 2013) suggest mathematical formulations and a Monte Carlo-based heuristic for the centralized scheduling and allocation of rescue units under certainty and under uncertainty, respectively.

This study focuses on decision support in operational management during the response phase of natural disaster management. To augment existing work, we develop and computationally validate a large set of heuristics for the decision support problem of centralized coordination of rescue units in terms of their schedules and assignments to incidents. We evaluate all heuristics against two benchmarks: best practice solutions and lower bounds of optimal solutions.

3. Optimization model

This section introduces the problem of scheduling rescue units and assigning them to incidents optimally after the occurrence of a disaster. We refer to this problem as the *Rescue Unit Assignment and Scheduling Problem* (RUASP).

3.1. Problem specification

The problem size is determined by the number of available rescue units m and the number of incidents n that needs to be processed. We consider situations in which the number of available rescue units is smaller than or equal to the number of incidents ($m \leqslant n$) as this ratio is typical in natural disasters. Furthermore, we account for the following properties.[1]

Property 1. Since not every rescue unit is able to process each incident, we account for both specific requirements of incidents and different capabilities of rescue units.

Property 2. Processing times are both incident-specific and unit-specific.

Property 3. Different rescue units need different travel times between the locations of incidents.

Property 4. The processing of an incident must not be interrupted (non-preemption).

Property 5. Each incident is assigned a weighting factor accounting for both casualties and damage induced over time. This weight is named factor of destruction or severity level. The sum of weighted completion times regarding the processing of incidents measures, as a proxy, the overall harm.

We illustrate the RUASP in Fig. 1, which shows a feasible solution of a problem instance with $m = 5$ units and $n = 12$ incidents. For each incident j, the level of severity (i.e. factor of destruction) is given by $w_j \in \{1, \ldots, 5\}$. The sample schedule considers the specific requirements (types) of incidents and the capabilities of rescue units. Here, the variable cap_{kj} equals 1 if and only if rescue unit k has the capability to process incident j.

The figure indicates that the problem to be solved is static and that all incidents, available rescue units and their characteristics are known. However, the decision support system updates its assignments continuously.[2] As a consequence, it seems realistic to assume that each instance does not exceed a moderately large size ($m, n \leqslant 40$), for which our heuristics can provide feasible solutions in timely manner.

3.2. Relationship to routing and scheduling problems

This section explores the relationship of RUASP to existing problems from both routing and scheduling.

[1] In order to provide decision support for realistic situations, we conducted interviews with associates from the German Federal Agency for Technical Relief (THW). These associates provided us with profound information on on-site coordination in the upright aftermath of the 2011 earthquake and tsunami in Japan.

[2] In practice, information is likely to be updated frequently so that assignment and scheduling decisions have to be refreshed based on the status quo of available information. We account for these dynamics by suggesting that the optimization model is applied in an iterative manner: if the decision makers determine to update the current scheduling and allocation plan based on new information, a new instance of the optimization problem with updated information is created. When solving this new instance, one needs to account for the fact that some of the known incidents have already been or are being processed. Accordingly, rescue units may have been already assigned and sent to incidents. In this case, it must be prohibited to assign busy rescue units until they will have finished their jobs (non-preemption). To sum up, a sequence of instances is generated and solved during the disaster response phase.

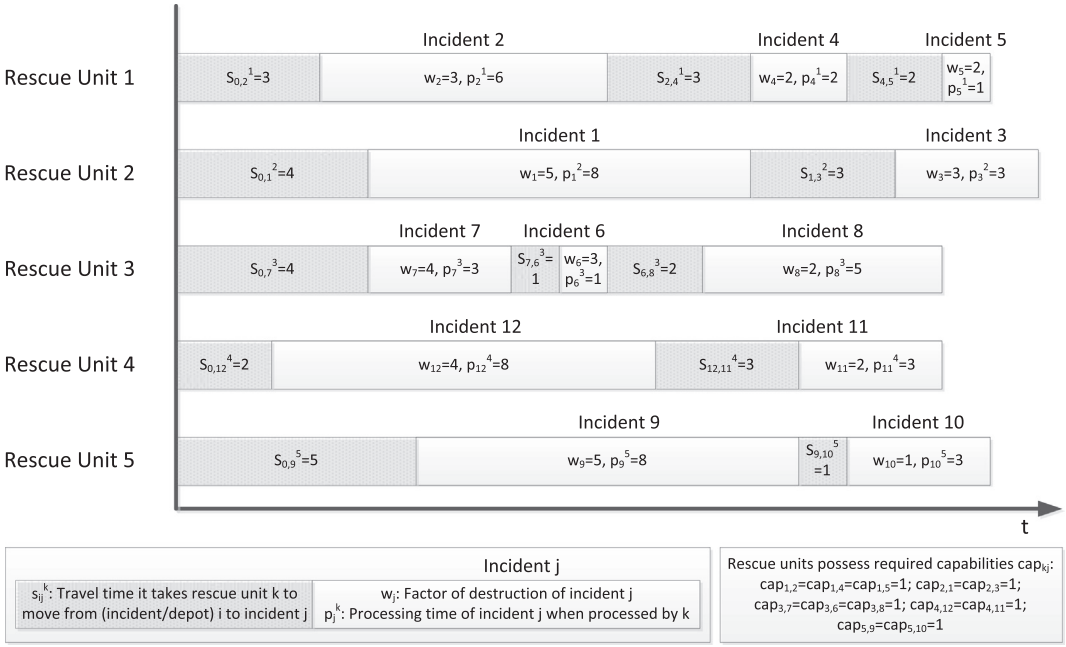

Fig. 1. Feasible solution for sample schedules and assignments with $m = 5$ units and $n = 12$ incidents.

In the routing domain, our problem is related to the multiple Traveling Salesman Problem (mTSP), which is a generalization of the TSP and a relaxation of the Vehicle Routing Problem (VRP) with the capacity restrictions removed (Bektas, 2006). To prove the relationship to mTSP, one needs to map rescue units to salesmen and incidents to cities/nodes while requiring that rescue units need to return to a central depot (given by a fictitious incident) with severity level 0. Furthermore, Property 1 (i.e. capabilities) is modeled by setting the corresponding mTSP decision variables to 0. While we can aggregate processing times and travel times in the RUASP to overall travel times, Properties 2 and 3 also require travel times in the mTSP to be salesman-specific. These properties can be modeled by providing salesmen-specific travel times between two cities. In addition to that, Property 4 (non-preemption) is inherently included in the mTSP. Altogether, this leads to the problem mTSP *with salesman-specific travel times*.

With regard to modeling this problem, it seems straightforward to extend existing mTSP models. In the literature, different mTSP models are suggested (Bektas, 2006). Among these models, only the flow based formulation, which uses three-index decision variables (for two cities and one salesman), can be easily modified to account for salesman-specific travel times. This extension requires leaving all constraints unchanged and substituting only the objective function coefficients c_{ij} by c_{ij}^k, with k being the index of the salesman and i as well as j being the indices of the cities.

Finally, Property 5 addresses the objective to minimize the sum of weighted completion times. However, a serious issue is caused by considering this property since the objective function in mTSP depends only on the edges traveled, but not on the order in which they are traveled. Considering also Property 5 leads to a mTSP *with salesman-specific travel times under minimizing the sum of weighted visiting times*. We are not aware of related research where a problem of this structure is addressed.

In the same manner as the mTSP, the VRP shares the issue caused by Property 5. Again, we are not aware of any VRP extension that allows modeling our problem. To sum up, the RUASP is related to both the mTSP and the more general VRP, but it is neither a specialization nor a relaxation of any of these problems. Consequently, neither an exact mTSP algorithm nor exact VRP can be regarded as an exact RUASP algorithm. However, as the sets of constraints of the mTSP (in the flow based formulation) and of the RUASP are equal, Section 4 adapts heuristics for the mTSP to the RUASP.

The RUASP is also related to problems in the scheduling literature. If we map rescue units to machines, incidents to jobs and travel times to setup times, then the RUASP is similar to the *parallel-machine scheduling problem with unrelated machines, non-batch sequence-dependent setup times, and a weighted sum of completion times as the objective*, classified as $R/ST_{SD}/\sum w_jC_j$ in the scheduling literature (Allahverdi, Ng, Cheng, & Kovalyov, 2008). The RUASP generalizes this scheduling problem which fulfills Properties 1, 2, 4 and 5 as the RUASP provides for machine-specific setup times between two jobs, while, in the scheduling problem, times depend only on the jobs. More precisely, the RUASP becomes a $R/ST_{SD}/\sum w_jC_j$ scheduling problem if setup times are machine-independent. Property 1 of the RUASP (i.e. capabilities) can be modeled by setting the corresponding decision variables to 0. With regard to the problem formulation of RUASP, any formulation of the scheduling problem $R/ST_{SD}/\sum w_jC_j$ may be used and modified so that Property 3 (different rescue units need different travel times between the locations of the incidents) holds.

However, according to the review by Allahverdi et al. (2008), there is only one publication adressing this scheduling problem (Weng, Lu, & Ren, 2001). While this paper suggests a recursive objective function, it specifies the constraints at high level only. Thus, their model formulation is too generic for our intention to suggest an optimization model. We suggest and computationally compare several heuristics based on Weng et al. (2001), which can be adapted to the RUASP (see Section 4).

3.3. Mathematical model

In this section, we propose an optimization model to find optimal schedules and assignments of rescue units to incidents. The

Table 1
Notation used in the mathematical model.

Input parameters	
n	Total number of incidents, with set $I = \{1, \dots, n\}$
m	Total number of rescue units, with set $K = \{1, \dots, m\}$
$w_j \in R^{\geqslant 0}$	Factor of destruction (severity level) of incident j
$p_j^k \in R^{\geqslant 0}$	Time required by rescue unit k to process incident j; ∞ if rescue unit k is incapable of addressing incident j
$s_{ij}^k \in R^{\geqslant 0}$	Travel time required by rescue unit k to move from incident i to incident j; if $i = 0$ then rescue unit k resides at its depot before traveling to incident j
$cap_{ki} \in \{0,1\}$	1 if rescue unit k is capable of addressing incident i; 0 otherwise
Decision variables	
$X_{ij}^k \in \{0,1\}$	1 if incident i is processed by rescue unit k immediately before processing incident j; 0 otherwise
$Y_{ij}^k \in \{0,1\}$	1 if incident i is processed by rescue unit k (at any time) before processing incident j; 0 otherwise

model is presented in a binary quadratic formulation.[3] The notation is given in Table 1.

The mathematical model can be written as

$$\min_{X_{ij}^k, Y_{ij}^k} \sum_{j=1}^{n} \left(w_j \sum_{i=0}^{n} \sum_{k=1}^{m} \left[p_i^k Y_{ij}^k + \left(p_j^k + s_{ij}^k \right) X_{ij}^k + Y_{ij}^k \left(\sum_{l=0}^{n} X_{li}^k s_{li}^k \right) \right] \right) \tag{O}$$

$$\text{s.t.} \sum_{i=0}^{n} \sum_{k=1}^{m} X_{ij}^k = 1, \quad j = 1, \dots, n, \tag{C1}$$

$$\sum_{j=1}^{n+1} \sum_{k=1}^{m} X_{ij}^k = 1, \quad i = 1, \dots, n, \tag{C2}$$

$$\sum_{j=1}^{n+1} X_{0j}^k = 1, \quad k = 1, \dots, m, \tag{C3}$$

$$\sum_{i=0}^{n} X_{i,n+1}^k = 1, \quad k = 1, \dots, m, \tag{C4}$$

$$Y_{il}^k + Y_{lj}^k - 1 \leqslant Y_{ij}^k, \quad i = 0, \dots, n; \quad j = 1, \dots, n+1; \quad k = 1, \dots, m; \quad l = 1, \dots, n, \tag{C5}$$

$$\sum_{i=0}^{n} X_{il}^k = \sum_{j=1}^{n+1} X_{lj}^k, \quad l = 1, \dots n; \quad k = 1, \dots m, \tag{C6}$$

$$X_{ij}^k \leqslant Y_{ij}^k, \quad i = 0, \dots, n; \quad j = 1, \dots, n+1; \quad k = 1, \dots, m, \tag{C7}$$

$$Y_{ii}^k = 0, \quad i = 0, \dots, n+1; \quad k = 1, \dots, m, \tag{C8}$$

$$Y_{ij}^k \leqslant cap_{ki}, \quad i = 1, \dots, n; \quad j = 1 \dots n+1; \quad k = 1, \dots, m, \tag{C9}$$

$$\sum_{l=1}^{n+1} X_{il}^k \geqslant Y_{ij}^k, \quad i = 0, \dots, n; \quad j = 1 \dots n+1; \quad k = 1, \dots, m, \tag{C10}$$

$$\sum_{l=0}^{n} X_{lj}^k \geqslant Y_{ij}^k, \quad i = 0, \dots, n; \quad j = 1 \dots n+1; \quad k = 1, \dots, m, \tag{C11}$$

$$X_{ij}^k, Y_{ij}^k \in \{0,1\}, \quad i = 0, \dots, n; \quad j = 1, \dots, n+1; \quad k = 1, \dots, m. \tag{C12}$$

The objective function (O) of the model minimizes the weighted sum of completion times over all incidents. In addition to the existing n incidents, we add two fictitious incidents given by 0 as the starting point (named depot) and $n+1$ as the ending point. These require no processing time ($p_0^k = p_{n+1}^k = 0$), but unit k needs a given setup time $s_{0j}^k \geqslant 0$ to move from its starting location to incident j. In addition to that, we set $s_{j(n+1)}^k = 0$ for all rescue units k. Let w_j denote the so-called factor of destruction of incident j. Consequently, the lower the factor of destruction, the less severe is the incident.

Constraint (C1) ensures that there is exactly one incident that is processed immediately before each of the n non-fictitious incidents. Similarly, Constraint (C2) ensures there is exactly one incident that is processed immediately after each of the n non-fictitious incidents. Constraints (C3) and (C4) guarantee that each rescue unit starts processing the fictitious incident 0 (the depot) and each rescue unit ends processing the fictitious incident $n+1$. Constraint (C5) accounts for the transitivity in predecessor relationships. If an immediate predecessor for a specific incident j exists, there has to be a successor as given by Constraint (C6). Constraint (C7) indicates that an immediate predecessor is also considered a general predecessor. Constraint (C8) prohibits a reflexive, direct or indirect predecessor relationship. Constraint (C9) ensures that rescue unit k is not assigned to incident i if k has not the capability to process i. Constraints (C10) and (C11) ensure that Y_{ij}^k is set to 0 if rescue unit k does not process incident i before incident j. Constraint (C12) makes the model a binary program. Each feasible solution of the minimization model represents valid schedules and assignments for all rescue units.

The above RUASP formulation can benefit from removing some variables and constraints depending on the particular problem instance. Using $cap_{ki} = 0$, it follows that $X_{ij}^k = Y_{ij}^k = 0$ for $j = 1, \dots n+1$. Thus, these variables can be removed from the model. Additionally, those constraints of (C5)–(C9) can be removed where $cap_{ki} = 0 \vee cap_{kj} = 0 \vee cap_{kl} = 0$. Apparently, the extent of model simplification depends on the number of capabilities rescue units have. However, for the sake of clarity, we do not explicitly integrate these simplifications in the above model.

With regard to computational complexity, it can be shown easily that the RUSAP is computationally intractable and NP-hard. The proof is included in the online appendix.

4. Heuristics for solving the rescue unit assignment and scheduling problem

Beyond proving NP-hardness of the RUASP (see online appendix), we used small up to moderately large instances with m, $n \leqslant 40$ to evaluate practical runtimes. Using a mixed integer nonlinear programming optimizer, more precisely, the Simple Branch and Bound solver in GAMS, we found that even small instances cannot be solved optimally in a practically reasonable time. As confirmed in interviews with the German Federal Agency of Technical Relief (THW), decision support in practice must be provided in less than 30 minutes. Therefore, we suggest several heuristics for solving the RUASP.

Greedy heuristic: This heuristic is applied in practice in emergency operations centers, usually in a manually-operated and non-automated decision-making process. We gained information on this heuristic through interviews with the THW. As this heuristic processes incidents in descending order of their level of severity, we refer to it as GREEDY heuristic.

Construction heuristics: We draw on the scheduling literature and adapt seven heuristics (Weng et al., 2001) proposed for solving the $R/ST_{SD}/\sum w_j C_j$ scheduling problem. We name the heuristics SCHED1 to SCHED7.

Improvement heuristics: Based on the routing literature, we adapt the classical 2-opt and 3-opt exchange procedure within a single rescue unit (Lin, 1965; Lin & Kernighan, 1973) as well as

[3] As noted in Section 3.2, problem formulations of the related mTSP and the $R/ST_{SD}/\sum w_j C_j$ scheduling problem are available, but, eventually, turned out to be not useful for modeling the RUASP. With regard to the mTSP, the RUASP requires an objective function in which the order of processed incidents is considered. We suggest such an objective function by introducing artificial decision variables that model predecessor relationships. As these variables are appropriate for easily adding subtour elimination constraints, we do not need to draw on the so-called MTZ-based subtour elimination constraints in the flow-based formulation (Bektas, 2006, p. 215). The other constraints included in the flow-based formulation are used in similar form.

multi-unit 2-opt and 3-opt, resulting in four heuristics. Furthermore, we suggest a load balancing heuristic.

GRASP metaheuristics: We integrate the previously mentioned construction and improvements heuristics into GRASP metaheuristics.

Monte Carlo-based heuristic: We propose a Monte Carlo-based heuristic in order to account for randomness in the search process.

With the exception of the Monte Carlo-based heuristic, the overall set of suggested heuristics can be divided into the set of 8 construction heuristics, which generate initial feasible solutions of RUASP instances, and 5 improvement heuristics, which iteratively generate new feasible solutions and test them for local optimality. Combining each of the construction heuristics with each of the improvement heuristics, we finally yield 40 composed heuristics, all of which are considered in our computational experiments.

In the remaining part of this section, we first describe the construction heuristics. Then, we suggest improvement heuristics before we illustrate GRASP metaheuristics and the Monte Carlo-based heuristic. We use the notations as introduced in Table 1.

4.1. Construction heuristics

The group of construction heuristics consists of the GREEDY approach used in practice and a set of construction heuristics originating from scheduling literature. Let τ_k denote the total processing and setup time for unit k in the corresponding iteration. The assignment α_k stores the last incident processed by unit k in the current iteration. The variable $\tilde{p}_i$ gives the average processing time needed for processing incident i by those units that are capable of i. Then, each heuristic returns $\sigma = (\sigma_1, \ldots, \sigma_m)$, which is a list of schedules for all m units.

4.1.1. Greedy heuristic

The GREEDY heuristic, which models best practice in emergency operations centers today, follows the idea that incidents are assigned to rescue units in descending order of the factor of destruction. Here, each incident j is assigned to a rescue unit k that is capable of processing incident j immediately while considering assignment history and updated travel times. The pseudoce of the GREEDY algorithm is described below.

1: Sort incidents in decreasing order of severity, $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n$, and set $C \leftarrow \{w_1, \ldots, w_n\}$.
2: Initialize the current completion time of each rescue unit, rescue units to start at the depot, the ordered list of incidents assigned to unit, i.e. $c_k \leftarrow 0, \quad \alpha_k \leftarrow 0, \quad \sigma_k \leftarrow \emptyset \quad \forall k \in K$.
3: **for** $\imath = 1$ **to** n **do**
4: Select incident $i \leftarrow \imath$ to be processed.
5: $K^* \leftarrow \{k \in K | cap_{ki} = 1\}$ are all units capable of processing incident.
6: **if** $K^* \neq \emptyset$ **then**
7: $unit \leftarrow \arg\min_{k \in K^*} \tau_k + s^k_{\alpha_k, i}$ chooses unit with lowest start time.
8: **else**
9: **return** unsuccessfully (no feasible assignment).
10: **end if**
11: Update $\tau_{unit} \leftarrow \tau_{unit} + s^{unit}_{\alpha_{unit}, i} + p^{unit}_i, \quad \alpha_{unit} \leftarrow i, \quad \sigma_{unit} \leftarrow \sigma_{unit} \cup \{i\}$.
12: **end for**
13: **return** $\sigma \leftarrow (\sigma_1, \ldots, \sigma_m)$ being the list of schedules.

Obviously, the greedy algorithm ignores the eventuality that it may not be optimal to process the most severe incidents first since processing times may also play a crucial role in the decision-making process.

Although the GREEDY heuristic proceeds dynamically through updating the availability and travel times of rescue units, it acts myopically in regard to the selection of the incident that is assigned next. For example, it may be sub-optimal regarding the overall harm (cmp. objective function (O)) to first assign to rescue unit k the most severe incident that has a comparably long processing time and, then, to assign to unit k the incident with the second largest factor of destruction and with a comparably short processing time. Apparently, the GREEDY heuristic may easily fail in providing *good* solutions to an instance of the RUASP. However, because of its simplicity, it provides solutions quickly and is applicable in practice even without computational support for small instances.

4.1.2. Scheduling heuristics

To consider a trade-off between severity and processing time, we adapt 7 heuristics for the scheduling problem $R/ST_{SD}/\sum w_j C_j$ as suggested by Weng et al. (2001).

The first heuristic differs from the greedy algorithm in two ways: (1) jobs are ordered based on the ratio of their processing time averaged over all units to the severity level. (2) The criterion for assigning incidents to units does not only consider the time required to travel to the location of the respective incident but also the time required to process the incident. In more detail, the algorithm named SCHED1 proceeds as follows.

1: Sort incidents by $\frac{\tilde{p}_1}{w_1} \geqslant \frac{\tilde{p}_2}{w_2} \geqslant \cdots \geqslant \frac{\tilde{p}_n}{w_n}$ with $\tilde{p}_i \leftarrow \frac{1}{m} \sum_{k \in \{K | cap_{ki} = 1\}} p^k_i$ being the average processing time of incident i, and set $C \leftarrow \left\{\frac{\tilde{p}_1}{w_1}, \ldots, \frac{\tilde{p}_n}{w_n}\right\}$.
2: Initialize the current completion time of each rescue unit, rescue units to start at the depot, the ordered list of incidents assigned to unit, i.e. $c_k \leftarrow 0, \quad \alpha_k \leftarrow 0, \quad \sigma_k \leftarrow \emptyset \quad \forall k \in K$.
3: **for** $\imath = 1$ **to** n **do**
4: Select incident $i \leftarrow \imath$ to be processed.
5: $K^* \leftarrow \{k \in K | cap_{ki} = 1\}$ are all units capable of processing incident.
6: **if** $K^* \neq \emptyset$ **then**
7: $unit \leftarrow \arg\min_{k \in K^*} \tau_k + s^k_{\alpha_k, i}$ chooses unit with start time.
8: **else**
9: **return** unsuccessfully (no feasible assignment).
10: **end if**
11: Update $\tau_{unit} \leftarrow \tau_{unit} + s^{unit}_{\alpha_{unit}, i} + p^{unit}_i, \quad \alpha_{unit} \leftarrow i, \quad \sigma_{unit} \leftarrow \sigma_{unit} \cup \{i\}$.
12: **end for**
13: **return** $\sigma \leftarrow (\sigma_1, \ldots, \sigma_m)$ being the list of schedules.

The second scheduling heuristic, namely SCHED2, differs from heuristic SCHED1 by assigning an incident to that rescue unit which has the lowest processing time. Thus, Step 7 is replaced as follows.

7: $unit \leftarrow \arg\min_{k \in K^*} p^k_i$ chooses unit with lowest average processing time.

Furthermore, the following algorithm SCHED3 considers processing times and travel times but ignores history. Hence, Step 7 looks as follows.

7: $unit \leftarrow \arg\min_{k \in K^*} s^{unit}_{\alpha_{unit},i} + p^k_i$ chooses unit with lowest sum of travel and average processing time.

Further heuristics named SCHED4, SCHED5 and SCHED6 are exactly the same as heuristics SCHED1, SCHED2 and SCHED3, respectively, except that, in Step 1, incidents are renumbered using their minimum processing time rather than using the average processing time:

1: Sort incidents by

$$\frac{\tilde{p}_1}{w_1} \geqslant \frac{\tilde{p}_2}{w_2} \geqslant \cdots \geqslant \frac{\tilde{p}_n}{w_n} \quad \text{with} \quad \tilde{p}_i \leftarrow \min_{k \in \{\kappa | cap_{\kappa i}=1\}} p^k_i$$

being the minimum processing time of incident i and set $C \leftarrow \left\{\frac{\tilde{p}_1}{w_1}, \ldots, \frac{\tilde{p}_n}{w_n}\right\}$.

This step requires that a minimum exists always. If a minimum does not exist, then the respective incident cannot be processed by any of the units and the instance has, thus, no feasible solution. In order to avoid drawbacks induced by pre-ordering incidents (as in algorithms SCHED1 to SCHED6), the following algorithm SCHED7 selects both incident and unit in the same step.

1: Initialize the current completion time of each rescue unit, rescue units to start at the depot, the ordered list of incidents assigned to unit, i.e.
$c_k \leftarrow 0, \quad \alpha_k \leftarrow 0, \quad \sigma_k \leftarrow \emptyset \quad \forall k \in K.$
2: Initialize list of incidents $I \leftarrow \{1, \ldots, n\}$.
3: Set $C \leftarrow \left\{\frac{\tau_k + s^k_{\alpha_k,i} + p^k_i}{w_i} \mid i \in I, k \in K\right\}$ and $c \leftarrow \min_{i \in I, k \in K} \frac{\tau_k + s^k_{\alpha_k,i} + p^k_i}{w_i}$.
4: **for** $\iota = 1$ **to** n **do**
5: Select incident $i^* \in I$ and unit $k^* \in K$ corresponding to c, i.e. here is the ratio of completion time to severity level minimal. If no minimum exists, stop unsuccessfully (no feasible assignment possible).
6: Update $I \leftarrow I \setminus \{i^*\}, \quad \tau_{k^*} \leftarrow \tau_{k^*} + s^{k^*}_{\alpha_{k^*},i^*} + p^{k^*}_{i^*}, \quad \alpha_{k^*} \leftarrow i, \quad \sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{i^*\}$.
7: Update $C \leftarrow \left\{\frac{\tau_k + s^k_{\alpha_k,i} + p^k_i}{w_i} \mid i \in I, k \in K\right\}$ and $c \leftarrow \min_{i \in I, k \in K} \frac{\tau_k + s^k_{\alpha_k,i} + p^k_i}{w_i}$.
8: **end for**
9: **return** $\sigma \leftarrow (\sigma_1, \ldots, \sigma_m)$ being the list of schedules.

4.2. Improvement heuristics

We consider heuristics for k-opt node exchanges originating from routing literature as well as load balancing as improvement heuristics.

4.2.1. Routing heuristics

In the routing literature, k-opt exchange procedures constitute improvement heuristics for solving the Traveling Salesman Problem (Lin, 1965; Lin & Kernighan, 1973), where in each iteration a k-opt exchange is applied until no further k-opt exchange leads to an improvement of the objective value (local optimum is reached). However, in our setting the exchange of 2 or 3 edges across units leads to infeasible solutions when (sequences of) incidents are assigned to units which are not capable of processing these incidents. Thus, we do not exchange edges but nodes (i.e. incidents) and refer to these moves as 2-nodes and 3-nodes exchange respectively. We apply these exchange procedures in two ways. First, a k-node exchange is applied inside the schedule of each rescue unit individually (named 2NSU with $k = 2$ and 3NSU with $k = 3$ respectively). Second, exchanges are applied across schedules of multiple rescue units (named 2NMU with $k = 2$ and 3NMU with $k = 3$ respectively). The procedures of the resulting four heuristics are shown in Figs. 2 and 3.

4.2.2. Load balancing heuristic

When queues of rescue units tend to get long in large-scale disaster scenarios, incidents at the end of the queue need to wait comparably long until being processed. This can result in excessively large harm (in terms of objective value). In order to avoid an extremely severe impact, we suggest a load balancing heuristic LOADBAL that aims at improving a current solution by reassigning the last incidents in a queue to the end of another queue. Let i_k be the last incident in the (ordered) list σ_k. Then, the LOADBAL heuristic proceeds as follows.

1: Initialize $\text{harm}(\sigma_k) \leftarrow \sum_{\varsigma \in \sigma_k} w_\varsigma \left(\sum_{\iota=1}^{\varsigma} s^k_{\iota-1,\iota} + p^k_\iota\right)$ to be the harm related to unit $k \in K$.
2: **repeat**
3: $k^* \leftarrow \arg\max_{k \in K} \text{harm}(\sigma_k)$ selects the unit k^* with the highest harm.
4: Select the unit k' for which the processing of incident i_{k^*} as the last incident of the queue results in the lowest additional harm, i.e.

$$k' \leftarrow \arg\min_{k \in \{\kappa \in K | cap_{\kappa,i_{k^*}}=1\}} \text{harm}(\sigma_k \cup \{i_{k^*}\}) - \text{harm}(\sigma_k).$$

5: Determine the reduction and the increase of harm caused by moving incident i_{k^*} from the queue of unit k^* to k', i.e.

$$\Delta\text{harm}_{k^*} \leftarrow \text{harm}(\sigma_{k^*}) - \text{harm}(\sigma_{k^*} \setminus \{i_{k^*}\}),$$
$$\Delta\text{harm}_{k'} \leftarrow \text{harm}(\sigma_{k'} \cup \{i_{k^*}\}) - \text{harm}(\sigma_{k'}).$$

6: **if** $\Delta\text{harm}_{k^*} - \Delta\text{harm}_{k'} > 0$ **then**
7: Create new solution with less harm by setting

$$\sigma_{k^*} \leftarrow \sigma_{k^*} \setminus \{i_{k^*}\}, \quad \text{harm}(\sigma_{k^*}) \leftarrow \text{harm}(\sigma_{k^*}) - \Delta\text{harm}_{k^*},$$
$$\sigma_{k'} \leftarrow \sigma_{k'} \cup \{i_{k^*}\}, \quad \text{harm}(\sigma_{k'}) \leftarrow \text{harm}(\sigma_{k'}) + \Delta\text{harm}_{k'}.$$

8: **end if**
9: **until** $\Delta\text{harm}_{k^*} - \Delta\text{harm}_{k'} \leqslant 0$

4.3. GRASP metaheuristics

Construction heuristics suffer from a shortcoming, i.e. they follow the same search path over and over. As a remedy, GRASP (greedy randomized adaptive search procedure) offers a possibility to diversify the solutions generated by the construction heuristic (Feo & Resende, 1995; Pitsoulis & Resende, 2002; Resende & Ribeiro, 2003). More precisely, GRASP is a multi-start metaheuristic for combinatorial problems in which each iteration consists of two phases: construction and local search. The construction phase uses a construction heuristic to create feasible solutions, whose neighborhood is searched using an improvement heuristic until a local minimum is found. The best overall solution is kept as the result. GRASP variants of algorithms GREEDY and SCHED1 to SCHED7 as construction heuristics are given by the following pseudocode.

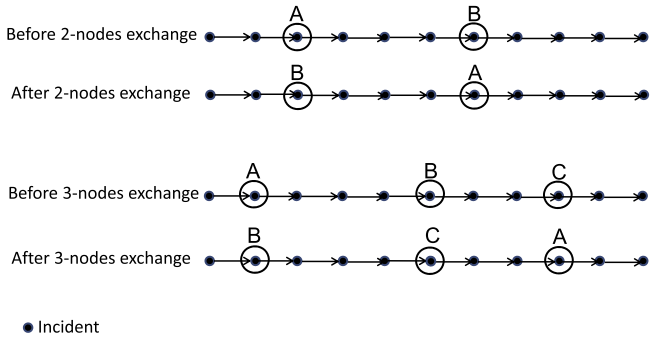

Fig. 2. Illustration of 2-nodes and 3-nodes exchange steps in a single unit.

1: Initialize $S \leftarrow \emptyset$.
2: **for** *iter* = 1 to $\mathcal{N}$ (max. iterations)
3: Perform greedy randomized construction by initializing candidate set C, i.e. perform initial steps in algorithms GREEDY and SCHED1 to SCHED7 respectively.
4: **for** $\iota = 1$ **to** n **do**
5: Compute $c_{\min} \leftarrow \min\{c|c \in C\}$ and $c_{\max} \leftarrow \max\{c|c \in C\}$.
6: RCL $\leftarrow \{c \in C|c \leqslant c_{\min} + \alpha(c_{\max} - c_{\min})\}$.
7: Select randomly a value $c \in$ RCL and let i be the corresponding incident.
8: Perform steps inside the loop in algorithms GREEDY or SCHED1 to SCHED7 without reassigning i.
9: Update $C \leftarrow C\backslash\{i\}$.
10: **end for**
11: Set $\sigma \leftarrow (\sigma_1, \ldots, \sigma_m)$ being the list of schedules.
12: Perform local search upon σ by one of the improvement heuristics giving σ'. Update list of solution by $S \leftarrow S \cup \{\sigma'\}$.
13: **end for**
14: **return** solution min S.

4.4. Monte Carlo-based heuristic

At last, we design a Monte Carlo-based heuristic to solve our problem for the following reasons. First, Monte Carlo simulation is flexible with regard to future extensions of the optimization model, such as co-allocation of rescue units and the consideration of informational uncertainty. Second, the complexity of the RUASP is high because of the many constraints and we assume that a Monte Carlo-based heuristic will not easily get stuck in a local optimum. In more complex scenarios, "*evaluation procedures rely a great deal on trial and error*" (Buxey, 1979, p. 566). In contrast, a Monte Carlo method overcomes this shortcoming.

The key idea of generating a feasible solution in our Monte Carlo-based heuristic is that incidents are iteratively scheduled in two stages. In stage one, an incident is assigned randomly to one of the D most appropriate rescue units where appropriateness is defined based on processing times. The motivation of this procedure is based on avoiding both (a) assignments of incidents to units that require an extremely long time for processing (thus, a parameter $D \in [0\%, 100\%]$ is used), and (b) myopic assignments of incidents to units that require the shortest processing time among all units (thus, randomness is included). In a second stage, the incident is inserted into the incident queue of the previously selected rescue unit. The criterion for determining the position of the new incident in the queue is based on a weighted ratio of the severity of incident w_i and the time $p_i^{k^*}$ it takes the selected rescue unit to process this incident. Each queue lists its incidents in descending order of $w_i/p_i^{k^*}$.

The Monte Carlo-based heuristic runs a fixed number of iterations with the Monte Carlo-based heuristic being the one with the lowest value found in all iterations. The Monte Carlo-based heuristic requires two input parameters: D and $\mathcal{N}$. $D \in [0\%, 100\%]$ is used for the selection of rescue units. The variable $\mathcal{N}$ is the number of feasible solutions generated; we set $D = 90\%$ and $\mathcal{N} = 500,000$ based on the results of pre-tests.[4] In more detail, the Monte Carlo-based heuristic MC proceeds as follows.

1: **for** *iter* = 1 **to** $\mathcal{N}$ (max. iterations) **do**
2: Initialize the cumulative processing time of each rescue unit, rescue units to start at the depot, the ordered list of incidents assigned to unit, i.e. curr_process_time(k) $\leftarrow 0$, $\alpha_k \leftarrow 0$, $\sigma_k \leftarrow \emptyset$ $\forall k \in K$.
3: **while** $I \neq \emptyset$ **do**
4: Select next incident $i \in I$ and update $I \leftarrow I\backslash\{i\}$.
5: $K^* \leftarrow \{k \in K|cap_{ki} = 1\}$ are all units capable of processing incident i.
6: **if** $K^* = \emptyset$ **then**
7: **return** unsuccessfully (no feasible assignment).
8: **end if**
9: Sort K^* in ascending order of curr_process_time and select randomly a rescue unit k^* with one of the D lowest values of curr_process_time of all rescue units in K^*.
10: Update $\tau_{k^*} \leftarrow \tau_{k^*} + s^{k^*}_{\alpha_{k^*}a_{st},i} + p_i^{k^*}$, $\alpha_{unit} \leftarrow i$.
11: curr_process_time(k^*) $\leftarrow$ curr_process_time(k^*) $+ p_i^{k^*}$.
12: Set $\sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{i\}$ and order σ_{k^*} in descending order of $w_i/p_i^{k^*}$.
13: **end while**
14: **end for**
15: **return** $\sigma = (\sigma_1, \ldots, \sigma_m)$ being the list of schedules.

5. Computational experiments

In our computational experiments, we evaluate the suggested heuristics against two benchmarks: (1) we compare the solutions of the heuristics with a lower bound of the optimal solution. We need to draw on lower bounds as finding optimal solutions even for moderately small instances turned out to be computationally infeasible. A gap between a solution found with a heuristic and the lower bound is an upper bound of the gap between the heuristic solution and the optimal solution. Thus, the determined gap underestimates the quality of the heuristic solutions. (2) We evaluate the solutions of all suggested heuristics regarding their improvement over the GREEDY heuristic, which represents best practice behavior of emergency operations centers, and, thus, it acts as a suitable benchmark. We first present our procedure to find an appropriate RUASP relaxation in order estimate lower bounds. Then, we explain the data generation for our experiments. Subsequently, we present and discuss results as well as runtimes.

5.1. Relaxation of the RUASP

We tried to find optimal solutions for the binary quadratic programming formulation of our problem using the Simple Branch and Bound solver (SBB) inside the software package GAMS. Even for small instances with 40 incidents and 40 rescue units, we are not able to find optimal solutions because of the NP-hardness of the RUASP. As a consequence, we derive appropriate relaxations of RUASP.

[4] In our simulations we did not find evidence that an increase in the number of iterations substantially improves the quality of solutions.

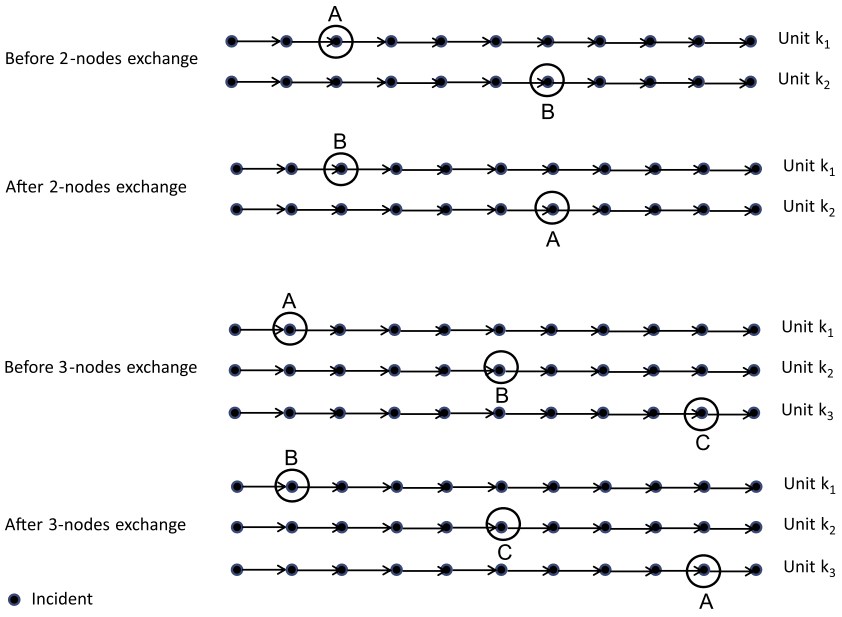

Fig. 3. Illustration of 2-nodes and 3-nodes exchange steps across units.

The computation of the lower bound is achieved by relaxing the binary constraints within the optimization model to $X_{ij}^k \in [0,1]$. We found this constraint relaxation most suitable because of the following reason: we examined and computationally tested each possibility to relax a constraint (for a scenario with 10 rescue units and 20 incidents) regarding its consequence for the mathematical model, the generation of schedules, the runtimes and the gap between the optimal solution of the original problem instance and the optimal solution of the relaxed problem instance. The relaxation of all but the binary constraints led to (a) unrealistic model extensions such as circular assignments or fragmentations of rescue units, (b) no significant enhancements concerning runtimes, and/or (c) an increase in the complexity of the whole model in terms of an exploding solution space or in terms of runtimes. The only suitable relaxation option was Constraint (C12), which has been found adequate for the calculation of lower bounds. The relaxation of the binary constraints of a model is a common method. Its application reduced runtimes substantially, while the gap between the optimal solution values of the problem with relaxation and the original problem turns out be low.

We used the CONOPT solver inside GAMS to solve this relaxed binary quadratic programming formulation of the optimization model. Runtimes for the largest instances (40 incidents and 40 rescue units) varied between 11 hour and 22 hour, which results in an average runtime of 15.6 hour. Even the calculation of the smallest scenarios with 10 incidents and 10 units took at least 2 seconds with an average of 207 seconds. The runtimes also indicate an exponential increase depending on the number of units and incidents. We also found that runtimes increase exponentially with both the number of units and incidents (significance of the overall model at the 0.1% level), with the number of incidents having a stronger impact.

5.2. Data generation

We designed the computational experiments based on interviews with associates of the THW. These associates were in direct contact to first search and rescue teams after the major earthquake in Japan in 2011. Hence, the generation of values for input parameters is given by Table 2. For our computational evaluation, we generated ten different instances for each scenario size. We limit the number of incidents and the number of rescue units to a maximum of 40 for three reasons: (1) our interviewees at the THW motivated these upper bounds to reflect practice given that a single rescue unit may consist of several members. (2) If a new situation makes it necessary to update old schedules and assignments, we assume that the new instance is unlikely to exceed these limits since some rescue units may have be assigned to a number of incidents already. (3) For implausible instances that consist of more than 40 rescue units and incidents, the available computing power was not sufficiently powerful to determine lower bounds of optimal solution values in reasonable times.

Looking at the situation after the 2011 disaster in Japan, we find that in urban areas where most of the incidents occur, travel times between incident locations are low compared to processing times. For example, it takes much more time to extinguish a house on fire or to stabilize a collapsed building than it takes a rescue unit to travel there. We consider this relationship by the mean values of the normal distributions for generating processing times and travel and setup times. Furthermore, the factor of destruction of an incident indicates the level of severity, as introduced by the U.S. Department of Homeland Security (2008): low (1), guarded (2), elevated (3), high (4), and severe (5) harm.

Table 2
Settings in randomly generated scenarios. Here, $U(\alpha,\beta,\gamma)$ is the discrete uniform distribution between α and β with step size γ.

Input parameter	Value, range or distribution
Number of rescue units	$m \in \{10,20,30,40\}$
Number of incidents	$n \in \{10,20,30,40\}$
Number of instances	10
Processing times	$p_j^k \sim N(20,10)$
Travel times	$s_{ij}^k \sim N(1,0.3)$
Factors of destruction	$w_j \in \{1,2,3,4,5\}$
Capabilities of rescue units	$A_k \sim U(1,1,4)$, $k \in K$
Capabilities required by incidents	$R_j \sim U(1,1,4)$, $j \in I$, $\text{cap}_{ki} = \begin{cases} 1, & \text{if } A_k = R_i; \\ 0, & \text{else} \end{cases}$
Number of iterations	500,000

Table 3

Mean results of composed heuristics in relation to lower bound solutions, i.e. $\mu\left(\frac{H_i}{LB}\right)$. Cells are colored according to this ratio (the brighter, the closer is the computed solution to the lower bound), whereas stars denote coefficient of variations (CV) with *** 0.03, ** 0.06, * 0.09.

Construction heuristic	Improvement heuristic	N	10	20		30			40			
		K	10	10	20	10	20	30	10	20	30	40
Greedy	—		2.631	2.125	3.283	2.547	2.714	3.993	2.926	3.104	3.734	4.515
	2Nsu		2.629	2.042	3.28	2.399	2.71	3.991	2.662	3.066	3.729	4.513
	2Nmu		2.629	2.038	3.28	2.373	2.71	3.991	2.619	3.064	3.729	4.513
	3Nsu		2.631	2.08	3.281	2.379	2.71	3.993	2.623	3.069	3.731	4.515
	3Nmu		1.439	1.268 *	1.407 *	1.317 **	1.283 **	1.388 **	1.479 **	1.368 *	1.347 **	1.465 **
	LoadBal		2.255	2.072	2.637	2.505	2.427	2.748	2.916	2.842	2.94	3.137
Sched1	—		1.176 **	1.276 *	1.252 *	1.376 **	1.24 **	1.259 **	1.591 **	1.306 **	1.25 **	1.32 **
	2Nsu		1.134 **	1.223 *	1.222 *	1.318 **	1.202 **	1.225 **	1.499 **	1.231 **	1.211 **	1.279 **
	2Nmu		1.134 **	1.217 *	1.219 *	1.31 **	1.202 **	1.224 **	1.474 **	1.225 **	1.209 **	1.279 **
	3Nsu		1.158 **	1.227 *	1.239 **	1.313 **	1.222 **	1.246 **	1.476 **	1.252 **	1.233 **	1.315 **
	3Nmu		1.124 ***	1.161 **	1.203 **	1.232 **	1.145 ***	1.203 ***	1.362 **	1.173 ***	1.178 ***	1.251 **
	LoadBal		1.176 **	1.276 *	1.252 *	1.376 **	1.24 **	1.259 **	1.591 **	1.306 **	1.25 **	1.32 **
Sched2	—		1.326	1.434	1.411 *	1.467	1.35 *	1.385	1.733	1.451 *	1.378 *	1.397 *
	2Nsu		1.251	1.356	1.335 *	1.394	1.281 **	1.309 *	1.639	1.358 *	1.31 **	1.327 **
	2Nmu		1.24	1.344	1.312 **	1.362 *	1.266 **	1.298 *	1.588	1.327 *	1.292 **	1.321 **
	3Nsu		1.254	1.345	1.324 **	1.363 *	1.28 **	1.326 *	1.59	1.331 *	1.299 **	1.329 **
	3Nmu		1.232	1.302	1.3 **	1.326 *	1.25 **	1.29 *	1.516	1.273 **	1.278 **	1.313 **
	LoadBal		1.285	1.359	1.341 *	1.405	1.276 **	1.311 *	1.686	1.357 *	1.292 **	1.331 **
Sched3	—		1.284	1.432	1.413 *	1.493	1.347 **	1.374	1.76	1.422 *	1.354 *	1.406 **
	2Nsu		1.232	1.359	1.331 *	1.426 *	1.278 **	1.296 *	1.667	1.329 *	1.285 **	1.328 **
	2Nmu		1.232	1.348	1.306 **	1.395 *	1.259 **	1.291 *	1.613	1.302 *	1.272 **	1.318 **
	3Nsu		1.246	1.35	1.321 **	1.394 *	1.272 **	1.316 *	1.613	1.312 *	1.282 **	1.333 **
	3Nmu		1.229 *	1.307	1.294 **	1.344 *	1.241 **	1.286 *	1.532	1.261 *	1.259 **	1.311 ***
	LoadBal		1.252 *	1.353	1.345 *	1.421 *	1.27 **	1.314 *	1.714	1.336 *	1.29 **	1.335 **
Sched4	—		1.172 *	1.266 *	1.216 **	1.362 **	1.212 **	1.234 **	1.623	1.279 **	1.264 *	1.269 **
	2Nsu		1.154 **	1.231 *	1.203 **	1.328 **	1.192 **	1.215 **	1.536	1.238 **	1.239 *	1.263 ***
	2Nmu		1.154 **	1.225 *	1.204 **	1.323 **	1.192 **	1.215 **	1.517	1.236 **	1.24 *	1.263 ***
	3Nsu		1.172 *	1.234 *	1.206 **	1.33 **	1.201 **	1.226 **	1.517	1.256 **	1.252 *	1.268 **
	3Nmu		1.137 **	1.162 **	1.195 **	1.226 **	1.159 **	1.2 ***	1.364 **	1.161 ***	1.19 **	1.251 ***
	LoadBal		1.172 *	1.266 *	1.216 **	1.362 **	1.212 **	1.234 **	1.623	1.279 **	1.264 *	1.269 **
Sched5	—		1.272	1.474	1.427 *	1.466	1.355 *	1.425	1.825	1.446	1.382 *	1.435 *
	2Nsu		1.24	1.357	1.317 *	1.384 *	1.277 **	1.318 *	1.671	1.347 *	1.297 **	1.327 **
	2Nmu		1.24	1.343	1.311 **	1.362 *	1.266 **	1.298 *	1.592	1.324 *	1.292 **	1.318 **
	3Nsu		1.248	1.357	1.348 *	1.384 *	1.275 **	1.328 *	1.587	1.331 *	1.3 **	1.357 **
	3Nmu		1.232	1.302	1.296 **	1.325 *	1.25 **	1.29 *	1.519	1.275 **	1.278 **	1.313 **
	LoadBal		1.235	1.368	1.321 **	1.395 *	1.285 *	1.309 *	1.778	1.329 *	1.256 **	1.333 *
Sched6	—		1.327	1.504	1.422 *	1.486	1.344 *	1.392	1.816	1.406	1.372 *	1.398 **
	2Nsu		1.269	1.383	1.312 *	1.401 *	1.274 **	1.294 *	1.671	1.312 *	1.288 **	1.307 ***
	2Nmu		1.268	1.362	1.306 **	1.382 *	1.264 **	1.281 **	1.593	1.293 *	1.278 **	1.301 ***
	3Nsu		1.276	1.376	1.343 **	1.402 *	1.271 **	1.312 *	1.588	1.295 *	1.284 **	1.329 **
	3Nmu		1.249	1.327	1.291 **	1.333 *	1.251 **	1.275 **	1.523	1.274 *	1.256 **	1.298 ***
	LoadBal		1.236	1.382	1.317 **	1.415 *	1.279 *	1.3 *	1.777	1.298 *	1.262 **	1.307 **
Sched7	—		1.119 ***	1.185 *	1.181 **	1.236 **	1.141 ***	1.198 ***	1.387 **	1.162 **	1.185 ***	1.229 ***
	2Nsu		1.119 ***	1.184 *	1.18 **	1.228 **	1.138 ***	1.197 ***	1.371 **	1.158 **	1.182 ***	1.229 ***
	2Nmu		1.119 ***	1.184 *	1.18 **	1.227 **	1.138 ***	1.196 ***	1.367 **	1.157 **	1.182 ***	1.229 ***
	3Nsu		1.119 ***	1.184 *	1.18 **	1.227 **	1.138 ***	1.196 ***	1.367 **	1.157 **	1.182 ***	1.229 ***
	3Nmu		1.109 ***	1.143 **	1.175 **	1.212 **	1.124 ***	1.193 ***	1.339 **	1.147 **	1.169 ***	1.228 ***
	LoadBal		1.119 ***	1.185 *	1.181 **	1.236 **	1.141 ***	1.198 ***	1.387 **	1.162 **	1.185 ***	1.229 ***
MC	—		1.118	1.41	1.926	1.683	2.092	2.681	1.92	2.378	2.971	3.453

Hence, we select a discrete uniform distribution for the severity levels.

The number of capabilities of rescue units was set to five. These account for policemen, fire brigades, paramedics, search and rescue units, and special casualty access teams. This discrete distinction of units' types and skills is based on and yet extending the classification of The New South Wales Government (2007). Incidents require exactly one of these differently skilled rescue units. The ratio of capabilities and the personnel required at an incident is generated randomly using a discrete uniform distribution.

The selection of the above parametric distributions has several reasons: (a) we found that real-world scenarios match such settings and (b) because of the individual variance of the selected distributions, the proposed heuristics are tested under unfavorable conditions. Other ranges of parameter values and distributions

did not result in significant deviations neither in the generated schedules nor in the assignments.

5.3. Data evaluation

We now evaluate the results of the suggested heuristics. We used the numerical computing environment MATLAB for implementation and simulation. We consider scenarios consisting of 10, 20, 30 and 40 incidents and units, with the number of units being lower than or equal to the number of incidents. For each problem instance, ten instances are randomly generated and solved by all heuristics. For each instance size and heuristic, we average the ratios of the heuristic solution H_i to the lower bound LB yielding averaged ratios $\mu\left(\frac{H_i}{LB}\right)$. Thus, the smaller the ratio, the closer is the heuristic solution to the lower bound. In addition, we calculate the respective averages when applying the GREEDY heuristic without any improvement heuristic, which represents current best practice. All results are shown in Table 3. The best practice results are given in the top row and originate from the GREEDY algorithm along with no improvement heuristic. The evaluation of Table 3 suggests the following findings[5]:

(1) All compositions of construction and improvement heuristic improve best practice results given by the GREEDY algorithm for each of the instance sizes. All results are significant at the .01 level (p-value of t-test).
(2) Results of the GREEDY algorithm are improved by each of the improving heuristic for each of the instance sizes (.01 level of significance).
(3) Choosing SCHED7 as construction heuristic in combination with any of the improvement heuristics leads to superior results compared to other combinations of construction and improvement heuristics. We found statistical evidence at the .05 level with the exception of only a few comparisons.
(4) Mean ratios of all except GREEDY-based composite heuristics tend to be well below 1.5. Results in terms of ratios become worse with large problem scenarios. Compositions consisting of the GREEDY heuristic lead to mean outcomes of between 1.268 and 4.515.
(5) The relative performance of the MC heuristic (bottom row) is highly volatile. It seems to be a good choice when scenarios are of small size, whereas results become worse with increasing size of the solution space. Another observation is that, for all instance sizes, MC dominates (at the .01 level) both the GREEDY heuristic and the joint application of the GREEDY heuristic and any of the heuristics 2NSU, 2NMU and 3NSU.
(6) In general, we identify composite heuristics using 3NMU performing best. In 247 out of 320 statistical comparisons, 3NMU-based algorithms performed better (at the .05 level) than composite heuristics without 3NMU.
(7) Improvement heuristics are able to improve the solutions provided by any construction heuristic. This holds for 1532 out of 3200 comparisons (at the 0.1 level).
(8) The application of the GRASP metaheuristics showed mixed results (cf. online appendix). Compared to the classical counterparts, which apply the same combination of construction and improvement heuristic, results are better in only some cases.
(9) Depending on the instance size, the best solution values achieved by the heuristics are at most 10.9% up to 33.9% higher than the lower bound, with the best results often provided by combinations which use SCHED7.

5.4. Runtimes

As solutions of RUASP instances need to be found within minutes in real natural disasters, acceptably low runtimes of the suggested heuristics are crucial for their practical usage. Runtimes of all heuristics, except those involving the 3NMU heuristic or the MC heuristic, were below one second for all instances of all sizes. The 3NMU heuristic required up to 20 seconds in instances of largest size (40 incidents and 40 rescue units) and is thus applicable in practice, too.

In contrast, runtimes of the MC are linear in the number of iterations and, as our results show, also depend on the instance size. Using 500,000 iterations in each of the runs, we found statistical evidence that the runtimes of the MC heuristic grow linearly with both rescue units and incidents, while the number of incidents has a slightly stronger impact. A detailed analysis is given in the online appendix. As average runtimes vary between 3.45 minutes for small instances (10 units and 10 incidents) and 18.26 minutes for large instances (40 units and 40 rescue units), the applicability of the MC heuristic depends on the instance size, on the number of iterations, rescue units and incidents, and on the available computing resources.

As shown above, GRASP metaheuristics show a possible path to improve solutions of both construction and improvement heuristics. These metaheuristics diversify the search paths and, consequently, require significantly more computation time. The average runtimes account for 38.03 seconds, but can get as high as 25.89 minutes. When integrating 3NMU inside GRASP in particular, average runtimes even rise to 187.63 seconds across all instances. Increasing the number of iterations inside GRASP also boosts runtimes, but without improving solutions.

5.5. Discussion

Our results show that the current best practice behavior in emergency response situations can be substantially improved by applying heuristics. As most improvements can be achieved in less than a second (only in a few cases, the computation time spans several minutes), our heuristics are well applicable in practice. As the RUASP generalizes the *parallel-machine scheduling problem with unrelated machines, non-batch sequence-dependent setup times and a weighted sum of completion times as the objective*, our algorithms can also be applied to this well-known class of scheduling problems.

Although our tested instances do not have more than 40 incidents and 40 rescue units, this limitation in size is of no substantial practical relevance for two reasons: first, our algorithms are likely to process instances of much larger size than 40 incidents and 40 rescue units in less than a minute. The limitation of size in our computations is rooted in the high computation times required to determine *good* bounds. Furthermore, additional computing power can be used to solve larger instances. Second, as we argued above, larger instance sizes of the RUASP are unlikely to occur as instances are generated and solved iteratively.

The benefit of having an optimization model and automated decision support available is obvious: the proposed decision support provides assistance to the decision makers in situations characterized by a high level of complexity and high time pressure. However, we would like to stress that the application of any of the proposed heuristics is intended to enhance human-based decision making and to offer decision support timely to decision makers; it is not intended to substitute the actual decisions of practitioners.

6. Conclusion and outlook

In this paper, we address the Rescue Unit Scheduling and Assignment Problem (RUASP), which is a key issue in emergency response management. Our contributions are as follows. We derive

a binary quadratic optimization model of the problem. Considering literature on scheduling and routing, we propose a Monte Carlo-based heuristic, eight construction heuristics, five improvement heuristics and GRASP metaheuristics. Then, we computationally evaluate and compare these heuristics. In addition to that, we evaluate the heuristics against the current best practice behavior and against lower bounds of optimal solutions. We found that the RUASP can be solved for instances with up to 40 incidents and 40 rescue units in less than a second, with the solution values being at most 10.9% up to 33.9% larger than the optimal value. While comparing heuristic solution values with lower bounds is particularly relevant for theoretical analysis, comparing heuristic solution values with the values found by the GREEDY heuristic is relevant for the disaster management domain because the GREEDY heuristic represents current best practice behavior. According to our results, our algorithms are capable of generating schedules which reduce the overall harm caused by the GREEDY heuristic to at least 42.0% and to at most 81.8%. This level of harm reduction is considerably large. This can help decrease casualties and economic losses substantially.

Some future research directions may enhance the applicability of our optimization model: (1) The integration of performance degradation and preemptive scheduling can be beneficial. Performance degradation becomes apparent when rescue units lose some of their vigor caused by the duration of their deployment and the constant pressure to save lives over time. (2) Time windows during which incidents need to be processed seem also adequate in emergency response settings. For example, time windows are of particular importance when humans are buried alive and need to be saved quickly. (3) Another interesting stream would be to analyze collaboration between rescue units and the coordination of autonomous agents. (4) We admit that a deterministic model in the envisaged application in disaster relief is questionable when information on incidents, including the level of severity, processing times and travel times, are not precisely known. While some information may be modeled stochastically based on historical data, other information is often described and assessed by humans, where linguistic estimations are common. In such cases, fuzzy set theory is a useful approach to model uncertainty. Future research needs to clarify when to use which type of uncertainty, how distribution functions and fuzzy membership functions can be modeled, and how resulting models can be solved.

Acknowledgements

We are grateful to the editor and the anonymous reviewers, who all provided many valuable comments which helped improve the paper.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejor.2013.10.029.

References

Airy, G., Mullen, T., & Yen, J. (2009). Market based adaptive resource allocation for distributed rescue teams. In J. Landgren & S. Jul (Eds.), *Proceedings of the 6th conference on information systems for crisis response and management (ISCRAM 2009)*. Gothenburg, Sweden.

Ajami, S., & Fattahi, M. (2009). The role of earthquake information management systems (EIMSs) in reducing destruction: A comparative study of Japan, Turkey and Iran. *Disaster Prevention and Management, 18*, 150–161.

Allahverdi, A., Ng, C., Cheng, T., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*, 985–1032.

Altay, N., & Green, W. G. III, (2006). OR/MS research in disaster operations management. *European Journal of Operational Research, 175*, 475–493.

Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega, 34*, 209–219.

Buxey, G. M. (1979). The vehicle scheduling problem and Monte Carlo simulation. *The Journal of the Operational Research Society, 30*, 563–573.

Chen, R., Sharman, R., Rao, H. R., & Upadhyaya, S. J. (2008). Coordination in emergency response management. *Communications of the ACM, 51*, 66–73.

Comes, T., Conrado, C., Hiete, M., Kamermans, M., Pavlin, G., & Wijngaards, N. (2010). An intelligent decision support system for decision making under uncertainty in distributed reasoning frameworks. In S. French, B. Tomaszewski & C. Zobel (Eds.), *Proceedings of the 7th international conference on information systems for crisis response and management (ISCRAM 2010)*. Seattle, USA.

Comfort, L. K., Ko, K., & Zagorecki, A. (2004). Coordination in rapidly evolving disaster response systems the role of information. *American Behavioral Scientist, 48*, 295–313.

Falasca, M., Zobel, C. W., & Fetter, G. M. (2009). An optimization model for humanitarian relief volunteer management. In J. Landgren & S. Jul (Eds.), *Proceedings of the 6th conference on information systems for crisis response and management (ISCRAM 2009)*. Gothenburg, Sweden.

Faraj, S., & Xiao, Y. (2006). Coordination in fast-response organizations. *Management Science, 52*, 1155–1169.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*, 109–133.

Fiedrich, F., Gehbauer, F., & Rickers, U. (2000). Optimized resource allocation for emergency response after earthquake disasters. *Safety Science, 35*, 41–57.

GAO (2006). *Disaster relief: Governmentwide framework needed to collect and consolidate information to report on billions in federal funding for the 2005 gulf coast hurricanes: GAO-06-834*.

Gasparini, P., Manfredi, G., & Zschau, J. (Eds.) (2007). *Earthquake early warning systems*. Berlin, Heidelberg: Springer.

IFRC (2012). *Disaster management - IFRC*.

Lambert, J. H., & Patterson, C. E. (2002). Prioritization of schedule dependencies in hurricane recovery of transportation agency. *Journal of Infrastructure Systems, 8*, 103–111.

Leifler, O. (2008). Combining technical and human-centered strategies for decision support in command and control: The ComPlan approach. In F. Fiedrich & B. van de Walle (Eds.), *Proceedings of the 5th conference on information systems for crisis response and management (ISCRAM 2008)*. Washington, DC.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal, 44*, 2245–2269.

Lin, S., & Kernighan, B. (1973). An effective Heuristic algorithm for the traveling salesman problem. *Operations Research, 21*, 498–516.

Nisha de Silva, F. (2001). Providing spatial decision support for evacuation planning: A challenge in integrating technologies. *Disaster Prevention and Management, 10*, 11–20.

Pitsoulis, L., & Resende, M. G. C. (2002). Greedy randomized adaptive search procedures. In P. M. Pardalos & M. G. C. Resende (Eds.), *Handbook of applied optimization* (pp. 168–181). New York: Oxford University Press.

Pollak, E., Falash, M., Ingraham, L., & Gottesman, V. (2004). Operational analysis framework for emergency operations center preparedness training. In *Proceedings of the 36th conference on winter simulation*. Piscataway, NJ: IEEE.

Reijers, H. A., Jansen-Vullers, M. H., Zur Muehlen, M., & Appl, W. (2007). Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications. In G. Alonso, P. Dadam & M. Rosemann (Eds.), *Proceedings of the 5th international conference on business process management (BPM 2007)*. Berlin, New York: Springer volume 4714 of Lecture Notes in Computer Science.

Resende, M. G. C., & Ribeiro, C. C., (2003). Greedy randomized adaptive search procedures. In: F. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics. US international series in operations research & management science* (Vol. 57, pp. 219–249). Boston: Springer.

Rolland, E., Patterson, R., Ward, K., & Dodin, B. (2010). Decision support for disaster management. *Operations Management Research, 3*, 68–79.

Saleem, K., Luis, S., Deng, Y., Chen, S.-C., Hristidis, V., & Li, T. (2008). Towards a business continuity information network for rapid disaster recovery. In S. A. Chun, M. Janssen, & J. R. Gil-García (Eds.), *Proceedings of the 2008 international conference on digital Government Research ACM international conference proceeding series*.

Sherali, H. D., Carter, T. B., & Hobeika, A. G. (1991). A location–allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B: Methodological, 25*, 439–452.

Svensson, A., Holst, J., Lindquist, R., & Lindgren, G. (1996). Optimal prediction of catastrophes in autoregressive moving-average processes. *Journal of Time Series Analysis, 17*, 511–531.

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*, 461–468.

UN/ISDR (2005). *Hyogo framework for action: building the resilience of nations and communities to disasters*.

U.S. Department of Homeland Security (2008). *Homeland security advisory system–guidance for federal departments and agencies*. Available from https://www.dhs.gov/files/programs/gc_1156876241477.shtm. (Accessed December 08, 2011).

van de Walle, B., & Turoff, M. (2008). Decision support for emergency situations. In F. Burstein & C. Holsapple (Eds.), *Handbook on decision support systems 2 international handbooks on information systems* (pp. 39–63). Berlin, Heidelberg: Springer.

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*, 215–226.

Wex, F., Schryen, G., & Neumann, D. (2011). Intelligent decision support for centralized coordination during emergency response. In M. A. Santos, L. Sousa & E. Portela (Eds.), *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM 2011)*.

Wex, F., Schryen, G., & Neumann, D. (2012). Operational emergency response under informational uncertainty: A fuzzy optimization model for scheduling and allocating rescue units. In L. Rothkrantz, J. Ristvej & Z. Franco (Eds.), *Proceedings of the 9th international conference on information systems for crisis response and management (ISCRAM 2012)*.

Wex, F., Schryen, G., & Neumann, D. (2013). Decision modeling for assignments of collaborative rescue units during emergency response. In *Proceedings of the 46th Hawaii international conference on system science*. Wailea, HI: IEEE Computer Society Press.