European Journal of Operational Research 285 (2020) 712–727

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

The median routing problem for simultaneous planning of emergency response and non-emergency jobs

Dylan Huizing[a,*], Guido Schäfer[a,b], Rob D. van der Mei[c,d], Sandjai Bhulai[d]

[a] Networks and Optimization Group, Centrum Wiskunde & Informatica, P.O. Box 94079, Amsterdam 1090 GB, The Netherlands
[b] Department of Econometrics and Operations Research, School of Business and Economics, Vrije Universiteit Amsterdam, Amsterdam 1081 HV, The Netherlands
[c] Stochastics Group, Centrum Wiskunde & Informatica, P.O. Box 94079, Amsterdam 1090 GB, The Netherlands
[d] Faculty of Science, Vrije Universiteit Amsterdam, Amsterdam, 1081 HV, The Netherlands

ARTICLE INFO

Article history:
Received 11 June 2019
Accepted 3 February 2020
Available online 10 February 2020

Keywords:
Location
Routing
Emergency logistics
Combined planning

ABSTRACT

This paper studies a setting in emergency logistics where emergency responders must also perform a set of known, non-emergency jobs in the network when there are no active emergencies going on. These jobs typically have a preventive function, and allow the responders to use their idle time much more productively than in the current standard. When an emergency occurs, the nearest responder must abandon whatever job he or she is doing and go to the emergency. This leads to the optimisation problem of timetabling jobs and moving responders over a discrete network such that the expected emergency response time remains minimal. Our model, the Median Routing Problem, addresses this complex problem by minimising the expected response time to the next emergency and allowing for re-solving after this. We describe a mixed-integer linear program and a number of increasingly refined heuristics for this problem. We created a large set of benchmark instances, both from real-life case study data and from a generator. On the real-life case study instances, the best performing heuristic finds on average a solution only 3.4% away from optimal in a few seconds. We propose an explanation for the success of this heuristic, with the most pivotal conclusion being the importance of solving the underlying *p*-Medians Problem.

1. Introduction

Optimal positioning, in applications such as firefighting and ambulance management, is an important part of emergency logistics research (Caunhye, Nie, & Pokharel, 2012). In classical models and settings, emergency responders are expected to remain at a base station until an emergency occurs, and an optimal or near-optimal set of waiting positions is determined (Brotcorne, Laporte, & Semet, 2003; Owen & Daskin, 1998; Plane & Hendrick, 1977).

However, in some real-world applications, it may be interesting to assign incident-preventing activities or other scheduled activities to responders when there are no active emergencies to resolve. For instance, one could want to assign routine railway inspections to idle railway emergency responders, or to combine scheduled ambulance transport with emergency response, or to plan routine patrols of the police force such that good emergency coverage is guaranteed. This would allow the emergency responders to spend their idle time much more effectively, namely by proactively preventing emergencies, rather than by waiting at a base station.

Combining an 'emergency response fleet' with a 'maintenance fleet' is challenging, but worthwhile (Kiechle, Doerner, Gendreau, & Hartl, 2009): this would yield more manpower for the scheduled work, as well as a larger pool of emergency responders. When an emergency happens, however, the nearest agent should abort whatever task he or she is doing and hurry towards the emergency. It would be undesirable if an emergency occurs, but due to poor planning, all responders are performing a task at some far-off location.

This gives rise to an interesting Operations Research challenge: how can we schedule these preventive tasks in the network, such that we can guarantee a good spread of emergency responders over the day and minimise the average response time to potential emergencies? Answering this question should provide a worthwhile contribution to the field of emergency logistics.

If one were to only care about minimising response time to emergencies, one could find an optimal distribution of agents over

* Corresponding author.
E-mail addresses: dylan.huizing@cwi.nl, dylanhuizing@gmail.com (D. Huizing), g.schaefer@cwi.nl (G. Schäfer), r.d.van.der.mei@cwi.nl (R.D. van der Mei), s.bhulai@vu.nl (S. Bhulai).

https://doi.org/10.1016/j.ejor.2020.02.002
0377-2217/

the network by solving a *p-Medians Problem* (P-MED) (Daskin & Maass, 2015). In P-MED, we have a finite set of nodes, each with some non-negative weight, and a symmetric distance matrix between these nodes. The goal is to select exactly p of these nodes which will act as 'facilities'. Each other node is then connected to its nearest facility, and contributes a cost equal to the distance to that facility times the node's weight. P-MED is the problem of finding the p nodes for which this total weighted distance is minimal. In this context, it would give the emergency responders optimal locations from which to anticipate emergencies.

If one were to only care about processing many preventive tasks in the network in the span of one work shift, one could find the fastest routing of agents over jobs by solving a *Distance-Constrained Vehicle Routing Problem* (DVRP) (Laporte, Desrochers, & Nobert, 1984). DVRP is almost identical to the classical *Vehicle Routing Problem* (VRP), except route lengths may not exceed some parameter T. More specifically, suppose we have again a finite set of nodes with a symmetric distance matrix. One node is called the 'depot'; the others are called the 'customers'. Each customer has a non-negative amount of processing time. Given a fixed number of vehicles p, a feasible solution to DVRP consists of exactly p tours that start and end at the depot, such that every client is visited exactly once, and such that every tour has a sum of traversed distances and processing times that does not exceed T. DVRP is the problem of finding the p feasible tours with the smallest total distance travelled. In this context, it would give the fastest routing of the agents over the preventive tasks such that everyone is back by the end of the shift.

The optimal solutions to these two problems would by definition be conflicting: the former would have the agents standing still, while the latter would have them move around with no explicit regard to emergency response times. In this context, however, one cares about *both* minimal response time and efficient task processing: one seeks to route agents over all given jobs, as in DVRP, but with the objective to minimise emergency response time, as in P-MED. This problem is therefore also interesting in that it lies on an unexpected boundary between two well-studied Operations Research problems, namely P-MED and DVRP.

Despite these motivations from academia and industry, the literature review in Section 2 suggests that this combined planning problem has received little attention. Therefore, this article proposes the *Median Routing Problem* (MRP) as a mathematical model for scheduling preventive tasks while minimising emergency response time, and proposes methods to find optimal or near-optimal solutions to this problem quickly.

The contribution of this paper is twofold. Firstly, we propose a mathematical model for this planning problem. This model allows for discretisation of continuous response time, is suited to deal with online re-planning when emergencies occur, and can be solved with mixed-integer linear programming. Secondly, we propose a heuristic for this model, that for real-life benchmark instances needs only 4.5 seconds to find solutions that are on average 3.4% away from optimal. This heuristic has an unusual approach, in that it decomposes the decisions into several NP-hard subproblems, but these NP-hard decisions are so much compressed that they can be solved within seconds in the benchmark instances. We propose an explanation for the success of this heuristic by comparing it with related heuristics under variation of instance parameters.

The remainder of this paper is structured as follows. In Section 2, we review literature concerning related problems. In Section 3, we give a rigorous problem definition, including its complexity. In Section 4, we describe a solution method and several heuristics. In Section 5, we detail the experimental setup in which these methods are compared. In Section 6, we present the results of computational experiments with some observations. In Section 7, we present our conclusions.

2. Related literature

Vast literature exists on solving and approximating P-MED, going back at least as early as the work of ReVelle and Swain (1970). Daskin and Maass (2015) have provided a recent overview of solution methods, construction and improvement algorithms and metaheuristics for P-MED. Among these, they provide a mixed-integer linear program. They remark that the variables describing median selection must be binary, but that the variables describing the assignment of nodes to medians may be left continuous. We will exploit a similar result in Section 4.1. If the distance matrix satisfies the triangle inequality, P-MED can be approximated to within a factor of $6\frac{2}{3}$ due to an LP-rounding result by Charikar, Guha, Tardos, and Shmoys (2002). This approximation factor has since been improved by Arya et al. Arya et al. (2004) using a local search method with swaps. They approximate P-MED to a factor $3+2/k$, where k is the number of swaps allowed to be made simultaneously.

Laporte reviewed a number of exact and approximate algorithms for DVRP (Laporte, 1992). Almoustafa, Hanafi, and Mladenović (2013) solved large instances of the variant with asymmetric travel costs using a modified branch-and-bound procedure with random tie-breaking. If we wish to minimise the number of vehicles needed rather than the travel costs, Nagarajan and Ravi (2012) provide a 2-approximation on tree metrics and an $(\mathcal{O}(\log \frac{1}{\varepsilon}), 1+\varepsilon)$-bicriteria approximation algorithm on general metrics.

Broader surveys of VRP variants were done by Eksioglu, Vural, and Reisman (2009),Toth and Vigo (2014) and Joubert (2007). In particular, in the Dynamic Vehicle Routing Problem (Pillac, Gendreau, Guéret, & Medaglia, 2013), the customers to be visited may appear during execution of the routes, and the decision maker is tasked with making a route over the known customers and to adjust them whenever new customers appear. This is similar, in some sense, to emergencies occurring and requiring a rescheduling. A key difference is that new Dynamic VRP customers can be incorporated into existing routes at any point, whereas emergencies demand immediate response. The inherent uncertainty in Dynamic VRP is dealt with in several ways, including Multiple Scenario Approaches (Pillac, Guéret, & Medaglia, 2012), a-priori routes (van Ee & Sitters, 2014; Zhang, Ohlmann, & Thomas, 2014), rolling horizon approaches (Jaillet, Bard, Huang, & Dror, 2002; Palma-Behnke et al., 2013) and rollout policies (Goodson, Ohlmann, & Thomas, 2013).

Ichoua, Gendreau, and Potvin (2000) use Tabu Search to minimise a weighted sum of travelled distance and lateness to both known and dynamically revealed jobs. One could adapt this to the problem at hand by seeing the revealed jobs as 'emergencies', and assigning zero weight to travelled distance and lateness to known jobs. However, their model would then prescribe that any feasible solution is optimal, as long as it responds to revealed jobs as quickly as possible.

Our research also considers situations where agents must be routed over jobs, but their start and end locations are not the same, despite this being a typical assumption in VRP variants. Therefore, our research makes use of the *(s,t)-path Travelling Salesman Problem* (path-TSP) (Hoogeveen, 1991). In path-TSP, we again observe a finite node set with a symmetric distance matrix. One node is called the 'start node' s; one other node is called the 'end node' t. Path-TSP is the problem of finding the shortest path that starts at s, ends at t, and visits each node exactly once. It can be solved by means of the mixed-integer linear program in Appendix C, or approximated by the method of Zenklusen (2019).

The problem at hand bears a strong resemblance to the k-Server Problem (Koutsoupias, 2009). In the k-Server Problem, requests appear dynamically in a metric space, and whenever this happens, a decision-maker must immediately decide which of k servers to send towards the request and how to reposition the rest. The goal is to minimise the total amount of distance travelled. Typically, probabilities for where requests may appear are not known, and researchers have focused on finding algorithms with small competitive ratio against someone who knows completely when and where the requests will appear.

Farahani and Hekmatfar (2009) describe a number of different facility location problems and concepts. In the Dynamic p-Median problems reviewed by Owen and Daskin (1998), locations may close and reopen in different time periods to satisfy period-dependent demands. In the Capacitated Mobile Facility Location Problem (Raghavan, Sahin, & Salman, 2019), initial facility locations have already been chosen, but one may relocate facilities against distance-dependent costs. The goal is to minimise a weighted sum of these facility relocation costs and the subsequent cost to serve all clients. The Location-Routing Problem, reviewed also by Drexl and Schneider (2015), is concerned with simultaneously deciding delivery routes and the facility locations from which they spring: the goal is to minimise the distance travelled between jobs.

Bertsimas and Van Ryzin (1993) study a dynamic Travelling Repairman Problem, where multiple agents may move freely over the Euclidean plane and must respond to dynamically revealed service requests as soon as possible. They describe policies with costs that are provably within a constant factor of the optimal policy costs.

For ambulances, the combined planning of emergency response and non-emergency patient transportation has received some attention. Kergosien, Gendreau, Ruiz, and Soriano (2014) study when and from which hospital to temporarily expend emergency ambulances on non-emergency transportation, as do van den Berg and Van Essen (2019). They seek to minimise the temporary loss in emergency response coverage when performing non-emergency transportation. In contrast, Kiechle et al. (2009) study a problem where emergencies are responded to by the nearest empty ambulance, including the ones performing non-emergency transportation. Their analysis is mostly focused on comparing whether it is better to arrive at the next job as early as possible or as late as possible.

In other fields, research has been done into combined maintenance-routing, which studies how to jointly determine when to perform maintenance and how to route between maintenance jobs. The maintenance schedule affects the routes, but the routes may also affect the maintenance schedule, depending on the piece of research. Most of the maintenance-routing literature in air transportation (Başdere & Bilge, 2014; Gopalan & Talluri, 1998; Haouari, Shao, & Sherali, 2012; Sarac, Batta, & Rump, 2006; Talluri, 1998) and train transportation (Maróti & Kroon, 2005; Penicka, Strupchanska, & Bjørner, 2003) seems to focus on how to execute a required transportation schedule with a set of vehicles, while ensuring that these vehicles are routed over maintenance stations regularly. Cohn and Barnhart (2003) include the subsequent crew scheduling into the optimisation as well.

López-Santana, Akhavan-Tabatabaei, Dieulle, Labadie, and Medaglia (2016) study a combined maintenance-routing problem in the oil and gas industry where the goal is to determine the expected optimal times and frequencies at which to perform maintenance, balancing the fixed cost of performing maintenance against the expected cost incurred when a machine breaks down and remains unrepaired until its next maintenance moment. Aside from determining the optimal times and frequencies in a maintenance planning phase (by optimising over continuous, non-linear functions numerically), they also try to fit feasible repairmen routes on this (using mixed-integer linear programming on a space-time network) in a routing phase, and iterate between the two until some stopping criterion is reached. Fontecha et al. (2020) improve upon this work in two notable ways. Firstly, they expand the model to allow for re-planning after breakdowns. Secondly, they replace the computation method with a more scalable version: that is, they remove the need to iterate between the two phases and they replace the mixed-integer linear program by a matheuristic. They then apply this to case studies in a large-scale sewage cleaning application. Irawan, Ouelhadj, Jones, Stålhane, and Sperstad (2017) study a maintenance-routing problem for offshore wind farms, that more closely resembles a Vehicle Routing Problem with Pick-up and Delivery. Inspired by this similarity, they solve it using a Dantzig-Wolfe decomposition method.

Some work has been done in coordinating several unmanned vehicles to provide joint 'coverage' over a region (Agarwal, Hiot, Joo, & Nghia, 2007; Doitsidis et al., 2012; Shu, Wang, Lin, Liu, & Zhou, 2013; Wang & Hussein, 2007). They focus on mapping out the entire area once with mobile camera's, rather than providing 'emergency coverage' whilst performing jobs in the region.

We conclude that many similar problems have been studied, but that each differs fundamentally from the problem at hand, and that a new model is required.

Table 1
Notation for the median routing problem.

Set		Description
$\mathcal{A}$		The set of agents
$\mathcal{J}$		The set of jobs
$\mathcal{V}$		The set of nodes
$\mathcal{T}$		The set of time-steps, $\mathcal{T} = \{0, 1, \ldots, T\}$
$\mathcal{V}_P$		The set of nodes where incidents may occur ($\mathcal{V}_P \subseteq \mathcal{V}$)
$\mathcal{V}_v$		The neighbourhood of $v \in \mathcal{V}$
Parameter	Domain	Description
S_a	$\mathcal{V}$	The start location of agent $a \in \mathcal{A}$
E_a	$\mathcal{V}$	The end location of agent $a \in \mathcal{A}$
L_j	$\mathcal{V}$	The node where job $j \in \mathcal{J}$ is located
Q_j	$\mathbb{Z}_{\geq 0}$	The number of time-steps job $j \in \mathcal{J}$ takes
P_v	(0,1]	The probability that the next emergency happens at node $v \in \mathcal{V}_P$
C_{uv}	$\mathbb{Q}_{\geq 0}$	The undiscretised emergency response time from $u \in \mathcal{V}$ to $v \in \mathcal{V}_P$
G_v	$\mathbb{Z}^2$	The coordinates of node $v \in \mathcal{V}$
Variable	Domain	Description
x_{avt}	{0, 1}	Whether or not agent $a \in \mathcal{A}$ is at $v \in \mathcal{V}$ at time $t \in \mathcal{T}$
y_{uvt}	{0, 1}	Whether or not a potential emergency at $v \in \mathcal{V}_P$, time $t \in \mathcal{T}$ will be responded to from $u \in \mathcal{V}$
z_{ajt}	{0, 1}	Whether or not agent $a \in \mathcal{A}$ starts job $j \in \mathcal{J}$ at time $t \in \mathcal{T}$

3. Problem definition

In this section, we formally introduce MRP as a mathematical optimisation problem. For an example instance that explains the intuitive ideas, we refer the reader to Appendix A. We also describe the problem by means of a mixed-integer linear program in Section 4.1. In order to give a formal definition of the MRP, we employ the notation listed in Table 1.

In MRP, we observe a connected, undirected graph with node set $\mathcal{V}$. The neighbourhood of $v \in \mathcal{V}$ is denoted $\mathcal{V}_v \subseteq \mathcal{V}$. We demand that each node is in its own neighbourhood. Each node v has known coordinates $G_v \in \mathbb{Z}^2$. We discretise time into a finite time horizon $\mathcal{T} = \{0, 1, \ldots, T\}$.

There is a set of agents $\mathcal{A}$ that can move over the network. That is: if agent a is at node u at time $t \neq T$, it can only be at $v \in \mathcal{V}$

at time $t+1$ if $v \in \mathcal{V}_u$. At time 0, each agent a is at some start location $S_a \in \mathcal{V}$. At time T, each agent must be at a specific end location $E_a \in \mathcal{V}$.

There also exists a set of jobs $\mathcal{J}$, distributed over the graph. Each job $j \in \mathcal{J}$ has a location $L_j \in \mathcal{V}$ and a processing time $Q_j \in \mathbb{Z}_{\geq 0}$. Each job must be processed and the jobs must be processed non-preemptively to succeed: that is, whenever an agent starts processing a job, that agent must stay at that location to process the job for its full duration, unless an emergency occurs. If a job is aborted halfway due to an emergency, the job fails and must be processed entirely anew.

Note that we allow for distinct jobs to be at the same node: for example, the depot may host several equipment maintenance and administrative jobs. Though we could combine all jobs at a given location into one superjob, there exist instances where doing so destroys feasibility. The only exception we make is that if a job has length 0 and there is another job at the same location, we will merge them into one. An alternative could be to place each job on its own virtual node, but we believe having to distinguish between real and virtual nodes that describe the same location is less elegant than simply allowing one node to host multiple jobs.

Finally, emergencies may occur in any node $v \in \mathcal{V}_P \subseteq \mathcal{V}$. An emergency may occur at any time-step $t \in \mathcal{T}$ against a time-independent probability. The probability that the next emergency occurs in $v \in \mathcal{V}_P$ is $P_v > 0$, with $\sum_{v \in \mathcal{V}_P} P_v = 1$. If an emergency at $v \in \mathcal{V}_P$ is responded to from an agent at $u \in \mathcal{V}$, the emergency response time is $C_{uv} \in \mathbb{Q}_{\geq 0}$. This distance matrix C does not need to be symmetric, and the methods presented in Section 4 do not require C to be symmetric, though the benchmark instances discussed in Section 5 do all have a symmetric C. Note that, outside of emergency logistics, P_v can be more broadly interpreted as node weights, and C_{uv} can be more broadly interpreted as service costs.

A feasible solution of MRP must tell the agents where to be at each time-step and which jobs to start processing when, respecting the above constraints. Moreover, for each $v \in \mathcal{V}_P$ and each $t \in \mathcal{T}$, a node $u \in \mathcal{V}$ must be appointed to 'cover' v in case of an emergency; in any optimal solution, u is always the node with lowest response time C_{uv} that has at least one agent present at time t. We encode any solution to MRP with binary variables x_{avt} indicating whether agent $a \in \mathcal{A}$ is at node $v \in \mathcal{V}$ at time $t \in \mathcal{T}$, binary variables y_{uvt} indicating whether $v \in \mathcal{V}_P$ is covered from $u \in \mathcal{V}$ at time $t \in \mathcal{T}$, and binary variables z_{ajt} indicating whether agent $a \in \mathcal{A}$ starts processing job $j \in \mathcal{J}$ at time $t \in \mathcal{T}$.

Using this notation, we remark that if exactly one emergency occurs at some time t and node $v \in \mathcal{V}_P$, the response time equals C_{u^*v} for some $u^* \in \mathcal{V}$, which has $y_{u^*vt} = 1$, while the other $u \in V$ have $y_{uvt} = 0$. In other words, the response time equals $\sum_{u \in \mathcal{V}} C_{uv} y_{uvt}$. If indeed an emergency happens at time t, it happens at node $v \in \mathcal{V}_P$ with probability P_v, meaning the overall expected response time to an emergency at time t equals

$$\sum_{v \in \mathcal{V}_P} P_v \left(\sum_{u \in \mathcal{V}} C_{uv} y_{uvt} \right)$$

In MRP, the goal is to find a feasible $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ with minimal $\sum_{t \in \mathcal{T}} \sum_{v \in \mathcal{V}_P} P_v (\sum_{u \in \mathcal{V}} C_{uv} y_{uvt})$, where by the above discussion, the summed expression equals the expected response time to an emergency if exactly one emergency happens at time $t \in \mathcal{T}$ and none have happened earlier. Summing this expression over $\mathcal{T}$ and dividing by $|\mathcal{T}|$ yields the expected response time to the next emergency, given that at most one emergency can happen per time-step. Note that dividing by $|\mathcal{T}|$ makes no difference to the optimal solution, and we leave out the scalar $1/|\mathcal{T}|$ for legibility. Therefore, the objective in MRP is to minimise the expected response time to the next emergency, under the condition that all jobs are processed and that all agents are at their end location at the end of the time horizon.

Two non-trivial modelling choices have been made here.

Remark 1. Discretising space-time allows us to use linear optimisation on an approximation of continuous movement. Furthermore, it facilitates legible day plans as output, and we can always reduce the lost accuracy to acceptable levels by discretising more finely, at the cost of more computational effort. One may interpret this discretisation as having agents move around over the set of potential facility locations in an instance of P-MED. Though one could also take the DVRP perspective of directing agents over jobs, rather than over discretised nodes, we believe this would create too much inaccuracy in expected response times when agents traverse long roads from one job to another. As a consequence of a discrete space-time model, it is possible for agents to choose non-shortest roads between jobs if these give better response times, and to roam the network freely for the sake of coverage in their remaining time: these things would also not be possible when simply routing over jobs.

Remark 2. MRP seeks to minimise the expected response time to the next emergency, but in no way captures the actual processing of emergency events. Instead, when an emergency actually occurs and agents are deployed, it is advised to make a new planning for the rest of the shift and the remaining jobs by observing a new MRP instance, in which the remaining agents have their current location as their start location. In this model, we thus only prepare for the next emergency with optimal expected response time and re-optimise whenever it actually occurs. This 'single coverage'-approach is in a sense similar to a rolling horizon approach, where we keep the uncertainty tractable by only looking so far ahead, except we look towards the next emergency rather than towards some rolling horizon.

3.1. Complexity

We discuss two complexity results in this section that guide the design of our solution approaches. Firstly, we remark that MRP on a complete graph without jobs is equivalent to P-MED, implying that MRP is NP-hard. In practice, one could view MRP as much harder than P-MED, because it involves solving $(T-1)$ instances of P-MED, where the decision in any one instance influences the decision space of all other instances.

More importantly, we have the following stronger complexity result.

Theorem 1. *Deciding whether an instance of MRP admits a feasible solution is NP-complete in general, even for the case with one agent.*

Proof. First, note that the problem of deciding whether a feasible solution exists for an MRP instance is in NP, because any feasible solution can be stored and checked in polynomial time and size with respect to the input. Next, take any instance of the Hamiltonian Path problem: that is, observe some connected graph with N nodes, some start node S and some end node E; without loss of generality, assume $S \neq E$. It is NP-complete to decide whether this graph admits a Hamiltonian (S, E)-path (Garey & Johnson, 1979): that is, an (S, E)-path that visits all nodes exactly once. Transform this into an instance of MRP by observing the same graph, placing a job of length 0 on each node, setting $T = N$ and having $|\mathcal{A}| = 1$ agent start at S and end at E. Because $T = N = |\mathcal{V}|$, the only way the agent can reach node E at time T and process all jobs is if the agent visits all nodes at least once and never twice. Therefore, every feasible solution of this MRP instance corresponds to a Hamiltonian Path, meaning it is NP-complete to decide whether this MRP instance admits a feasible solution. □

Corollary 1. *Unless P=NP, there exists no polynomial-time algorithm that is guaranteed to return a feasible solution if one exists. In particular, no polynomial-time approximation algorithms exist for MRP.*

These results validate the following design choices: that heuristics are needed for MRP, and that these heuristics must contain NP-hard problems themselves if they are to always return a feasible solution.

4. Methods

In this section, we describe a solution method and several heuristics for MRP.

4.1. Mixed-integer linear programming

In the notation already presented, the MRP can be formulated as the following *mixed-integer linear program* (MIP):

$$\min \sum_{t\in\mathcal{T}} \sum_{v\in\mathcal{V}_p} P_v \sum_{u\in\mathcal{V}} C_{uv} y_{uvt}$$

$$s.t. \quad x_{aS_a0} = 1 \quad (\forall a \in \mathcal{A}) \tag{1}$$

$$x_{aE_aT} = 1 \quad (\forall a \in \mathcal{A}) \tag{2}$$

$$\sum_{v\in\mathcal{V}} x_{avt} = 1 \quad (\forall a \in \mathcal{A})(\forall t \in \mathcal{T}) \tag{3}$$

$$x_{av(t+1)} \le \sum_{u\in\mathcal{V}_v} x_{aut} \quad (\forall a \in \mathcal{A})(\forall v \in \mathcal{V})(t = 0, \dots, T-1) \tag{4}$$

$$\sum_{u\in\mathcal{V}} y_{uvt} = 1 \quad (\forall v \in \mathcal{V}_P)(\forall t \in \mathcal{T}) \tag{5}$$

$$y_{uvt} \le \sum_{a\in\mathcal{A}} x_{aut} \quad (\forall u \in \mathcal{V})(\forall v \in \mathcal{V}_P)(\forall t \in \mathcal{T}) \tag{6}$$

$$\sum_{a\in\mathcal{A}} \sum_{t\in\mathcal{T}} z_{ajt} = 1 \quad (\forall j \in \mathcal{J}) \tag{7}$$

$$\sum_{\tau=t}^{t+Q_j} x_{aL_j\tau} \ge (Q_j+1) z_{ajt} \quad (\forall a \in \mathcal{A})(\forall j \in \mathcal{J})(\forall t \in \mathcal{T}) \tag{8}$$

$$z_{ajt} + \sum_{\tau=t}^{t+Q_j-1} z_{ak\tau} \le 1 \quad (\forall j,k \in \mathcal{J} : j \ne k, L_j = L_k)(\forall a \in \mathcal{A})(\forall t \in \mathcal{T}) \tag{9}$$

$$x_{avt}, z_{ajt} \in \{0,1\}, \; y_{uvt} \in [0,1]$$

Here, the objective equals the expected response time to the next emergency (up to a scalar $|\mathcal{T}|$, as explained in Section 3).

Constraints (1) state that all agents must start at their start location, and (2) that they must end at their end location. Constraints (3) state that an agent can only be in one place at a time.

Constraints (4) state that an agent can only be at node v at time $t+1$ if he or she was at some adjacent node u at time t, where the adjacency is indicated by whether or not $u \in \mathcal{V}_v$.

Constraints (5) state that each emergency node, at each timestep, must receive coverage from somewhere. Constraints (6) add, however, that coverage at time t can only be given from some node $u \in \mathcal{V}$ if there is someone actually present at node u at time t.

Constraints (7) state that each job must be initiated by someone at some point in time. Constraints (8) add that when an agent starts a job j, that agent must stay at location L_j for the duration of the job. The formulation also ensures that the job is started early enough to be finished before the end of the time horizon. For jobs that are in different places, this implies that they cannot be processed at the same time by the same agent.

Recall, however, that we allow for multiple jobs to be hosted at the same node. Suppose some pair of jobs $j \ne k$ exists at the same location $L_j = L_k$. Constraints (8) do not forbid one agent to process them simultaneously. We thus need constraints (9) to address this fringe case. Suppose some agent a wishes to process j at some time t. The constraint

$$z_{ajt} + \sum_{\tau=t}^{t+Q_j-1} z_{ak\tau} \le 1$$

then states that a cannot also start processing k at any time between t and $t+Q_j-1$: k can only be processed after j is done or before j is started. If k is started very briefly before t, say at $t' := t-1$, then the constraint

$$z_{akt'} + \sum_{\tau=t'}^{t'+Q_k-1} z_{aj\tau} \le 1$$

ensures that j is not started at t anymore. Due to this symmetry, constraints (9) ensure that if two jobs are at the same location, an agent cannot process them simultaneously.

Though the variables x_{avt} and z_{ajt} are explicitly constrained to be binary, this is not necessary for the variables y_{uvt}. The reason is as follows. Any feasible solution has $\mathbf{x}$ binary. Therefore, any optimal solution obviously has $y_{uvt} = 1$ for the closest $u \in \mathcal{V}$ to a given $v \in \mathcal{V}$ that has someone present at $t \in \mathcal{T}$. If several nodes are tied for closest, then dividing the coverage fractionally over these nodes would still give a feasible solution with optimal solution value, but breaking the tie arbitrarily would result in a feasible solution with strictly less basic variables, meaning the original fractional solution cannot be a vertex of the solution polytope. We conclude that the variables y_{uvt} can be formulated as continuous between 0 and 1 without fear of non-integral optimal solutions. This is fortunate, as the variables y_{uvt} comprise the vast majority of the variables. On a random sample of benchmark instances (which are further described in Section 5), this indeed yields an average reduction of 29.1% in computation time.

Though simply plugging this mixed-integer linear program into a *Mixed-integer linear programming solver* (MIP solver) will eventually yield the optimal solution, this approach can come with long and unpredictable computation times as the instance size grows. This is indeed observed in Section 6 for the more difficult instance classes.

4.2. WAIT-AT-MEDIANS-heuristic

In practical applications, solving the MIP proposed in Section 4.1 may take excessively long. As an alternative, we discuss some heuristics here, starting with the following simple one.

If there is a large gap of time in which agents do not process jobs, for example when the instance has almost no jobs at all, then the optimal place for the agents to be is at those places given by the solution of P-MED. It may sometimes be costly or infeasible to reach those places within the time window. If the amount of remaining time goes to infinity, however, it will be feasible to reach that steady state, and any costs incurred while getting there are outweighed by the saved cost of being optimally distributed over a long period of time.

One heuristic strategy could be to identify these medians, and spend as much time as possible at these medians. This, indeed, is proposed by the WAIT-AT-MEDIANS-heuristic (WAM), described by Algorithm 1 and illustrated in Fig. 1. WAM requires solving P-MED and a special variant of DVRP, which are NP-hard problems in their own right; these subroutines themselves may be approached with heuristics, though the results in Section 6 suggest that the subroutines are easy enough to solve to optimality for the studied benchmark instances. Note that, in practice, step

Algorithm 1 WAIT-AT-MEDIANS, high-level overview

1: Solve induced P-MED, with $p = |\mathcal{A}|$
2: Obtain shortest paths between jobs, medians, starts and ends
3: Solve DVRP with one median per route and agent-specific start/end locations
4: Infer **x**, **y** and **z**, with waiting done only at medians

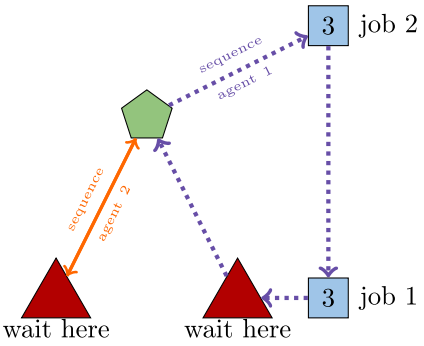


Fig. 1. The idea behind the WAIT-AT-MEDIANS-heuristic, applied to the example in Fig. A.3. First, solve P-MED, which results in identifying the large red triangles as the best places from which to offer emergency response. Then, solve a variant of DVRP to minimise the total time spent travelling, so as to maximise the total time spent waiting at the medians. The sequences thus obtained are shown here; they must still be translated back to a feasible solution in the discretised setting, but this can be done with ease. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

1 and part of step 2 can be done as preprocessing, assuming the network is known beforehand but the daily tasks are not. Though step 2 seems trivial, we remark that computing a distance matrix is not necessary for the MIP in Section 4.1, and we include this step in the description of WAM for the sake of fair computational comparison.

More in-depth, WAM consists of the following:

1. **Solve induced P-MED, with** $p = |\mathcal{A}|$. This can be done by solving the MIP formulated by Charikar et al. (2002), where the distances are given by C_{uv} and the nodes have weight P_v if they are in $\mathcal{V}_P$ and 0 otherwise. Denote the obtained medians $\mathcal{M} \subseteq \mathcal{V}$.
2. **Obtain shortest paths between jobs, medians, starts and ends.** This can be done in polynomial time using Dijkstra's algorithm (Dijkstra, 1959) or the Floyd-Warshall algorithm (Cormen, Leiserson, Rivest, & Stein, 2009). Denote $D_{ij} = D_{ji}$ the minimum number of steps needed to get from any job, median or start location i to any job, median or end location j.
3. **Solve DVRP with one median per route and agent-specific start/end locations.** This can be done by solving the MIP in Appendix B. The result is a sequence for each $a \in \mathcal{A}$, starting at S_a and ending at E_a, such that the sequences together visit all jobs, each median is visited by exactly one agent, each sequence admits a feasible execution with respect to the finite time horizon, and the total time spent travelling is minimised. This allows us to spend as much time as possible on waiting at medians. Note that, if there is enough time to visit all jobs but not enough time to also visit all medians, this subroutine fails.
4. **Infer x, y and z, with waiting done only at medians.** The sequences from the previous step can be translated to a feasible MRP solution easily. For each agent, observe the sequence from step 3, and demand that movement between any two goals follows the shortest path from step 2. If this requires strictly less time than T, allocate all remaining time to waiting at the median. This directly implies the values of x_{avt} and z_{ajt}. After thus fixing **x** and **z** completely, set y_{uvt} as follows: for any $v \in \mathcal{V}$, $t \in \mathcal{T}$, find the closest $u \in \mathcal{V}$ with respect to C_{uv} which has someone present, so which has $\sum_{a\in\mathcal{A}} x_{aut} > 0$, then set $y_{uvt} = 1$.

Though this heuristic seems intuitive, it comes with some immediately apparent downsides:

- It does not explicitly take coverage into account while routing over jobs, aside from creating as much median-waiting time as possible.
- It does not explicitly take coverage into account when translating routes back to the discrete network; it instead follows arbitrary shortest paths.
- Agents do not take into account where the other agents are.
- There exist feasible MRP instances where this heuristic does not produce a feasible solution, namely when there is not enough time to actually visit all medians.

4.3. *MEDIATE-DIVIDE-SEQUENCE-AGREE-heuristic*

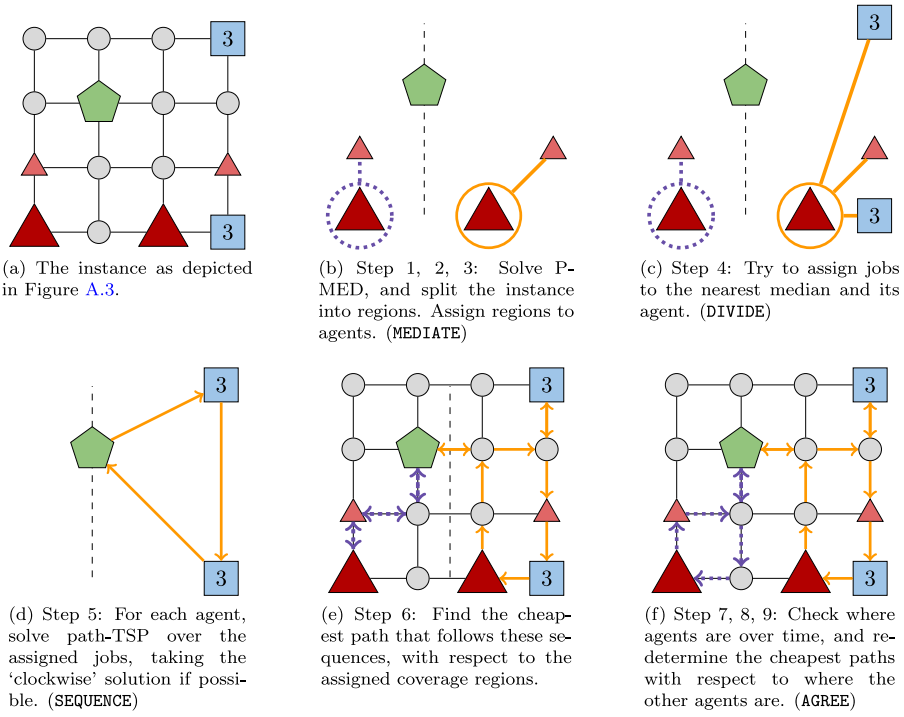
Observing the shortcomings of WAM, a heuristic is presented here which attempts to overcome the shortcomings. It will be referred to as MEDIATE-DIVIDE-SEQUENCE-AGREE (MDSA) and is presented as Algorithm 2 and illustrated in Fig. 2.

Algorithm 2 MEDIATE-DIVIDE-SEQUENCE-AGREE, high-level overview

1: Solve induced P-MED with $p = |\mathcal{A}|$, obtain coverage regions (MEDIATE)
2: Obtain shortest paths between jobs, medians, starts and ends
3: Assign medians optimally to nearest agents
4: Solve 'job division' (DIVIDE)
5: For each agent, solve path-TSP, take clockwise solution (SEQUENCE)
6: For each agent, given the sequence and region, find cheapest space-time path
7: Forget medians, refine to time-dependent regions (AGREE)
8: For each agent, given the sequence and time-dependent region, find cheapest space-time path
9: Infer **x**, **y** and **z**

In more detail, MDSA consists of the following steps:

1. **Solve induced P-MED with** $p = |\mathcal{A}|$, **obtain coverage regions (MEDIATE).** This step is identical to the one in WAM, resulting in a set $\mathcal{M}$ of medians. For any $m \in \mathcal{M}$, denote coverage region $\mathcal{V}_m \subseteq \mathcal{V}_P$ the nodes for which m is the nearest median, breaking ties arbitrarily.
2. **Obtain shortest paths between jobs, medians, starts and ends.** This step is again identical to the one in WAM. Denote again $D_{ij} = D_{ji}$ the minimum number of steps needed to get from any job, median or start location i to any job, median or end location j.
3. **Assign medians optimally to nearest agents.** If all agents start and end at one location, like a classical depot, then medians can be assigned arbitrarily to agents. Otherwise, each median m has an average distance $(D_{S_a m} + D_{m E_a})/2$ to the start and end point of a given agent $a \in \mathcal{A}$, and we can find the optimal assignment of medians to agents in polynomial time using the Hungarian algorithm (Kuhn, 1955). Denote by $m(a)$ the median assigned to a, and abbreviate $\mathcal{V}_a = \mathcal{V}_{m(a)}$.
4. **Solve 'job division' (DIVIDE).** In this step, each job is assigned to its nearest median and the corresponding agent, as well as possible. That is, for any $a \in \mathcal{A}$, $j \in \mathcal{J}$, denote the proxy cost of assigning j to a as $F_{aj} = (Q_j + 1) \cdot \sum_{v\in\mathcal{V}_a} P_v C_{L_j v}$;



(a) The instance as depicted in Figure A.3.

(b) Step 1, 2, 3: Solve P-MED, and split the instance into regions. Assign regions to agents. (MEDIATE)

(c) Step 4: Try to assign jobs to the nearest median and its agent. (DIVIDE)

(d) Step 5: For each agent, solve path-TSP over the assigned jobs, taking the 'clockwise' solution if possible. (SEQUENCE)

(e) Step 6: Find the cheapest path that follows these sequences, with respect to the assigned coverage regions.

(f) Step 7, 8, 9: Check where agents are over time, and redetermine the cheapest paths with respect to where the other agents are. (AGREE)

Fig. 2. An illustration of the MEDIATE–DIVIDE–SEQUENCE–AGREE-heuristic.

this quantity represents the cost incurred as a processes j, under the assumption that the nodes covered by a are always exactly $\mathcal{V}_a$. Blindly assigning jobs to their nearest median may result in an agent getting more jobs than feasibly executable. Instead, we find the feasible division of jobs over agents with minimal sum F_{aj} as follows: we again solve the MIP in Appendix B, except that we treat $\mathcal{M}$ as being empty, and we replace the objective with $\sum_{a\in\mathcal{A}}\sum_{j\in\mathcal{J}} F_{aj}z'_{aj}$. Note that, in contrast to WAM, this subroutine does *not* fail when there is not enough time to visit all medians. Denote $\mathcal{J}_a \subseteq \mathcal{J}$ the jobs assigned to agent $a \in \mathcal{A}$.

5. **For each agent, solve path-TSP, take clockwise solution (SEQUENCE).** Each agent $a \in \mathcal{A}$ now has a set of jobs $\mathcal{J}_a$ assigned to him or her. In this step, we decide in which sequence these jobs are visited. We do so by solving the (S_a, E_a)-path Travelling Salesman Problem over $\mathcal{J}_a$, as described in Appendix C, for each $a \in \mathcal{A}$.
   If $S_a = E_a$, then the found sequence in reverse is also optimal. Of these two optimal sequences, we choose the *clockwise* one, which we define as follows. Denote (X_s, Y_s) the 2D-coordinates of S_a. For any $j \in \mathcal{J}_a$, denote (X_j, Y_j) the 2D-coordinates of the job location L_j, and define the *angle* of j as $\text{atan2}(Y_j - Y_s, X_j - X_s)$, where atan2$(y, x)$ is a commonly used function to compute the geometric angle between the vector (x, y) and the vector (1,0) (De Dinechin & Istoan, 2015). When i is followed up by j, we define this move to be *clockwise* if i has a greater angle than j. Of the two optimal sequences, we choose the one that has the largest number of clockwise moves.
6. **For each agent, given the region and sequence, find cheapest space-time path.** If an agent $a \in \mathcal{A}$ is assumed to give coverage to a specific set of nodes $\mathcal{V}_a$, then each $u \in \mathcal{V}$ has a cost of a being there for one time-step, namely, $\sum_{v\in\mathcal{V}_a} P_v C_{uv}$. Based on these node costs, we compute for each $a \in \mathcal{A}$ the cheapest path starting at the space-time point $(S_a, 0)$ and ending at the space-time point (E_a, T), such that the jobs $\mathcal{J}_a$ are visited in the predetermined sequence. This can be done using a dynamic program, Algorithm 3 in Appendix D, which is essentially a modification of Dijkstra's algorithm (Dijkstra, 1959). This results in temporary values $\tilde{x}_{avt}$ describing the movement of the agents as they visit the jobs, while trying to keep optimal coverage over $\mathcal{V}_a$. In particular, if $\mathcal{J}_a = \emptyset$ for some $a \in \mathcal{A}$ and T is large enough, then this step results in a moving to median $m(a)$ and staying until it is time to go to end point E_a.
7. **Forget medians, refine to time-dependent regions (AGREE).** Up until this point, we assumed that $a \in \mathcal{A}$ would always cover a fixed area $\mathcal{V}_a$, so that a starting solution $\tilde{x}_{avt}$ could be constructed. In this step, we define more finely tuned, time-dependent coverage regions $\mathcal{V}_{at}$, by observing for each $v \in \mathcal{V}_P$ and $t \in \mathcal{T}$ which $a \in \mathcal{A}$ is 'nearest'; that is, which $a \in \mathcal{A}$ has minimal C_{uv}, where u is the location of a at time t according to the movement $\tilde{\mathbf{x}}$. This leads to the sets $\mathcal{V}_{at}$, which at each time-step partition $\mathcal{V}_P$ over the nearest agents.
8. **For each agent, given the sequence and time-dependent region, find cheapest space-time path.** We repeat step 6, but with the node cost of $a \in \mathcal{A}$ being at $u \in \mathcal{V}$ at time $t \in \mathcal{T}$ equal to $\sum_{v\in\mathcal{V}_{at}} P_v C_{uv}$, rather than $\sum_{v\in\mathcal{V}_a} P_v C_{uv}$. This results in final movement values x_{avt}.
9. **Infer x, y and z.** The previous step has produced final values for movement x_{avt}. The corresponding values for z_{ajt} can be easily inferred from Algorithm 3. The values for y_{uvt} can again be determined by checking for each $v \in \mathcal{V}_P$ and $t \in \mathcal{T}$ which $u \in \mathcal{V}$ with someone present has lowest C_{uv}.

The intuition of MDSA is less transparent from the algorithm. It is illustrated in Fig. 2 and described here.

As in WAM, we observe that as $T \to \infty$, it is optimal to have the agents spend their remaining time at the medians. Therefore, in step 1, we determine where the medians are. In step 2, we obtain the distances between the points of interest. In step 3, we use this to assign the $|\mathcal{A}|$ medians to their most suitable agent, with respect to start and end locations S_a and E_a. For the majority of

the algorithm, this not only assigns each agent a median, but also the region covered by that median: all agents now have a region $\mathcal{V}_a \subseteq \mathcal{V}_P$ that they are 'responsible' for.

In step 4, each job is assigned to its nearest median and handled by that median's agent if possible, so that all agents can try and stay in 'their' region; if this leads to too many jobs for one agent, the MIP in step 4 instead tries to divide jobs such that agents can stay as close to 'their' region as possible.

As soon as jobs are divided over the agents, each agent then determines in step 5 in which sequence the jobs will be visited. This is done by solving path-TSP, following again the logic that any time-step saved can be used to improve coverage. We synchronise all agents to move 'clockwise' if possible, so that whenever one agent moves away from a node, another will hopefully already be approaching to take over coverage.

In this discretised setting, there may exist many shortest paths between any two nodes. By taking the cheapest space-time path in step 6 with respect to coverage over $\mathcal{V}_a$, the ties between these shortest paths are broken sensibly rather than arbitrarily. In fact, this also allows non-shortest paths with better coverage to be chosen. Moreover, following the cheapest coverage paths is a more robust way of 'visiting the medians', rather than explicitly demanding they be visited: if there is not enough time, the cheapest coverage path will likely only approach the median the best it can.

However, assuming that some $a \in \mathcal{A}$ will always be the most appropriate agent to cover every node in $\mathcal{V}_a$, is a somewhat crude assumption. As agents visit jobs in a clockwise fashion, it may well be that nodes have one agent closest at one point and another at another point. Now that an initial movement profile $\tilde{\mathbf{x}}$ has been created, this can be used to determine time-dependent coverage regions in step 7 that are more fine and realistic than the coverage regions based on the medians. Finding the cheapest space-time paths based on these finer regions in step 8 produces a final movement profile in which agents actually observe where the other agents are at a given time, albeit that they look at $\tilde{\mathbf{x}}$ and hope that the other agents do not deviate too much from $\tilde{\mathbf{x}}$.

We remark that, if the MRP instance admits a feasible solution, then MDSA finds a feasible solution as well. The argument is as follows. If a feasible solution exists, then the MIP in step 4 will succeed in finding a feasible way to divide jobs over agents. By construction of step 4, every instance of path-TSP in step 5 admits a feasible solution, which will be found by the MIP. By the feasibility of step 5, the dynamic programs in the remaining steps will terminate successfully.

This guaranteed feasibility is one of the ways in which MDSA improves over WAM. Furthermore, MDSA prioritises coverage over travel time when dividing jobs, it breaks ties between fastest (and even non-fastest) paths by taking the cheapest ones with respect to a coverage profile, and it allows agents to respond to what the other agents have done in the final steps.

We acknowledge that this heuristic involves solving some NP-hard problems, as well as pseudo-polynomial dynamic programs with a running time that depends on T. However, we observe in Section 6 that MDSA needs mere seconds to run for the studied instances. An explanation for this speed is that, aside from step 1, the NP-hard subproblems involve routing over a relatively tiny set of jobs, rather than over a network with many nodes. In fact, one could say that the point of MDSA is that it first takes the most restricting decisions on a very small decision space with proxy costs, and then makes the best out of those decisions in pseudo-polynomial time. If one still chooses to replace the complex subroutines by heuristics, the guaranteed feasibility may be lost. As long as the bottleneck in studying heuristics for larger instances of MRP lies in finding the optimal solutions to compare against, developing a fully polynomial version of MDSA is left as a point for future research.

4.4. Partial versions of MDSA

The MDSA-heuristic involves a number of complex steps, the added value of which may not be immediately clear. To investigate this, the heuristic was modified slightly to allow switching steps on and off. This resulted in the heuristics MDS, MDA, DSA, MD, DS, DA and D, where for example MDA performs all steps except the SEQUENCE-step. That is to say, these seven heuristics are identical to MDSA, except:

- If 'M' is not in the name, the MEDIATE-step is skipped, or in particular step 1 and step 3 are skipped. It also implies that in step 4, a normal DVRP is solved, so simply the travelled distances are minimised; that step 5 can be skipped because of this; and that in step 6, translation back to the discrete graph is done by means of shortest paths, rather than paths that give the best coverage to a predetermined region.
- If 'S' is not in the name, the SEQUENCE-step is skipped. The sequences in which agents visit their jobs are then taken directly from step 4.
- If 'A' is not in the name, the AGREE-step is skipped. No re-optimisation to observed coverage regions is done, or put simply, steps 7 and 8 are skipped.

Note that the DIVIDE-step is never skipped, as the division of jobs over agents cannot be arbitrary. This procedure of skipping steps produces heuristics that are faster and conceptually easier, but on average perform worse, as can be observed in Section 6. In particular, D simply performs the jobs and returns to the depot as quickly as possible. We view this as a benchmark heuristic, that models how agents would move if they only cared about getting their jobs done as quickly as possible.

5. Experimental setup

In Section 4, we discussed a solution method and several heuristics for MRP. To compare how well these work in practice, it would be insightful to apply them to a set of benchmark problems. Furthermore, it would be of interest to see what typical solutions look like. In Section 2, however, we concluded that the MRP or similar problems have received little academic attention. To the best of our knowledge, benchmark instances for this problem did not yet exist prior to this research. Therefore, we created benchmark instances from both a case study and from an instance generator and compared the methods on these.

The code used to generate the benchmark instances is publicly available, as is the code to perform the experiments.

5.1. Used instances

We compared the methods on benchmark instances from two sources.

Firstly, we obtained six benchmark instances from a case study with a European railway infrastructure manager. These six instances, denoted $\mathcal{I}_R$, are defined on the same piece of the railway network, with emergency probabilities based on historical incident data. The jobs have been sampled from a database of typical jobs for this area. The six instances differ in which jobs have been sampled and how many. The instances have 143 nodes, 4 agents and $T = 16$. They have 3, 4, 5, 7, 8 and 9 jobs respectively.

Secondly, Algorithm 4 was developed to generate benchmark instances, in order to also study algorithmic behaviour under variation of problem features. The idea is simple: randomly draw node coordinates on the Euclidean plane, connect them if their Euclidean distance is under a given threshold, let all agents start and end at a central depot, scatter jobs randomly over the network and check if the result admits a feasible solution. Because these ideas

Table 2
Brief description of the benchmark instance classes used.

Class	Amount	Type	Size	Productivity	Sparseness
$\mathcal{I}_1$	50	random	small	productive	sparse
$\mathcal{I}_2$	50	$\mathcal{I}_1$-derived	small	unproductive	sparse
$\mathcal{I}_3$	50	$\mathcal{I}_1$-derived	small	productive	dense
$\mathcal{I}_4$	50	$\mathcal{I}_1$-derived	small	unproductive	dense
$\mathcal{I}_5$	50	random	medium	productive	sparse
$\mathcal{I}_6$	50	$\mathcal{I}_5$-derived	medium	unproductive	sparse
$\mathcal{I}_7$	50	$\mathcal{I}_5$-derived	medium	productive	dense
$\mathcal{I}_8$	50	$\mathcal{I}_5$-derived	medium	unproductive	dense
$\mathcal{I}_R$	6	real-life	± medium	both	± dense

are simple and the created instances are publicly available, further details on how the benchmark instances were generated have been moved to Appendix E.

The classes we created are described in Table 2. The details of their parameters are described in Appendix E, but their features roughly represent the following:

- 'Small' problems have 3 agents and 20 nodes, where 'medium' problems have 4 agents and 100 nodes;
- 'Productive' problems have more than two jobs per agent, where 'unproductive' problems have less than one;
- 'Sparse' problems have smaller node neighbourhoods than 'dense' problems, meaning it will typically take more time-steps to get from any one node to another.

Note that the medium size problems are of a comparable size to the real-life instances in class $\mathcal{I}_R$.

In order to purely observe the differences in performance under variation of features, only classes $\mathcal{I}_1$ and $\mathcal{I}_5$ were randomly generated: the unproductive problems were created from the productive problems by deleting jobs, and the dense problems were created from the sparse problems by updating adjacency for a higher adjacency threshold. By construction, these operations preserve feasibility of the instances.

5.2. Metrics and methods for solution structure

For each instance and method, we computed the expected response time and the computation time. On a selection of methods and a random sample of the instances, we also computed three metrics that give more insight into the structure of solutions.

Firstly, we compare the response time to how much it would have been if there had been no jobs at all. This indicates how much 'response power' we have sacrificed to do jobs. Secondly, the solutions from the different methods are compared by total travel distance, measured in the total number of 'hops'. Thirdly, making many tiny steps back and forth to continually compensate the movement of others can yield a marginal cost improvement, but may be irritating for the agent. To measure how 'jittery' a solution is, we also counted the total number of *shortcuts*, where a shortcut is defined as any occurrence of an agent visiting the distinct locations u, v and w at times t, $t+1$ and $t+2$ respectively, while the agent could also have gone from u to w directly. If the agent did so to process a job of length 0 at location v, this is not counted as a shortcut. This definition of a shortcut also accounts for an agent moving towards a destination at an irritatingly slow pace.

To make a more meaningful comparison of these metrics, we applied two additional solution methods to these instances. `Optimal jobless` means we delete the jobs and run the MIP: this does not give a feasible solution, but does provide a lower bound, and illustrates what solutions would have looked like if we did not care about jobs. Recall that `D` illustrates what solutions would have looked like if we did not care about coverage and simply solved VRP. `Split fleet` means we cut the fleet in half, where the larger half is only concerned with jobs, and the smaller half is only concerned with coverage. We apply `D` to the larger half and `Optimal jobless` to the smaller half, with the two halves ignoring each other's existence. In the case of an emergency, however, the nearest agent is still called upon, regardless of what half they are in. This illustrates what solutions would look like if, with the same resources, we decided for simplicity not to coordinate emergency response and non-urgent job processing jointly. Note that this method may also fail to find a feasible solution. It is also possible, of course, to split the fleet into portions of unequal size, but we believe examining this fifty-fifty split should suffice for our goal of better understanding solution structures.

5.3. Hardware specifications

All experiments were conducted on the Lisa Cluster, a computing cluster hosted by Surfsara. Each benchmark instance was solved on its own 16-core 2.60 GHz node with 64 GB QPI 8.00 GT/s memory. In particular, experiments were queued until they could get a node of their own, meaning all processing power of the node was dedicated to the experiment, and that clock times correspond to CPU times. Here, an 'experiment' means either solving a benchmark instance with a MIP solver, or applying all heuristics on it. Gurobi 8.0.1 was used as a MIP solver, using all 16 cores.

6. Results

The ten methods were applied to the nine classes of benchmark instances. The running time of these methods was recorded, as well as how large the gap was between the produced solution value and the optimal solution value, as a percentage of the optimal solution value. The full results are presented in Tables F.5, F.6 and F.7 in Appendix F. A summarised version is presented in Table 3. Table 4 presents comparisons of the other metrics. From the results in these tables, we make several observations.

In almost every class, `MDSA` is the best scoring heuristic in both average optimality gap and worst optimality gap. Class $\mathcal{I}_5$ is the most difficult class to solve for Gurobi, at an average of 140 minutes. Here, `MDSA` supplies a solution that is on average 3.5% away from optimal in an average 4.1 seconds. Taken over all 406 instances, Gurobi needs an average of 39 minutes to solve MRP, while `MDSA` needs 2.4 seconds to find a solution that is 3.2% away from optimal. It appears that simultaneously timetabling jobs and managing coverage is difficult, but that decomposing these decisions into different decision stages makes for an effective heuristic.

Averaged over all instances, the benchmark algorithm `D` is 127.4% away from optimal. The average optimality gap increases as the problem becomes denser and less productive. This is easily explained: in `D`, all agents go to their end point as fast as possible and stay there. In the benchmark instances, this end point is the same for all agents, meaning that near the end of the time horizon, everyone is in the same place, which is bad for coverage. The more unproductive the problem is and the denser the network, the sooner in the time horizon this occurs.

There is a large difference between the average optimality gap of heuristics `D` and `MDSA`. Of this difference, examining marginal contributions shows that 70.0% is due to step `M`, 29.95% is due to step `A` and 0.05% is due to step `S`. For the difficult class $\mathcal{I}_5$, the absence of `M`, `S` and `A` explain 71.8%, 0.4% and 27.8% of the difference respectively. In the case study instances $\mathcal{I}_R$, these marginal contributions are on average 65.6%, −0.7%, and 35.1%, respectively.

The added value of step `S` is apparently small in the observed instances. Unfortunately, the average added value of `S` is even negative in the case study instances, though this is caused largely by one outlier. In particular, on a random sample of the instances, the clockwising feature gives a strict improvement in only 13.3% of the

Table 3
Summarised results of applying the ten solution methods on the 200 small instances, 200 medium instances and 6 real-life instances.

Method	Average gap (%)	Average time (s)	Worst gap (%)	Worst time (s)
opt	0	2334.5	0	132750.0
WAM	11.7	56.8	140.5	4749.7
D	127.4	0.3	520.5	6.0
MD	4.2	1.8	45.9	7.1
DS	127.4	0.4	520.5	6.1
DA	53.7	0.6	281.9	7.0
MDS	4.0	1.9	43.2	7.3
MDA	3.4	2.3	56.6	8.3
DSA	54.0	0.7	284.5	7.2
MDSA	3.2	2.4	43.0	8.1

Table 4
Differences in solution structure when applying different methods on a sample of instances.

Method	Average gap to jobless (%)	Average number of hops	Average number of shortcuts	Average cost decrease versus split fleet (%)	Average hop increase versus split fleet (%)
opt	11.5	20.8	1.7	28.3	69.9
Optimal jobless	0	11.3	0	–	–
D	158.0	9.8	0	-53.2	-33.0
WAM	23.0	16.5	0.7	21.7	34.3
MDSA	13.0	19.1	1.2	27.3	52.5
Split fleet	69.1	13.5	0.5	0	0

cases, and the average objective improvement is less than 0.01%. On the studied instances, clockwising may not be worth its implementational effort. The ineffectiveness of S is understandable for unproductive problems, where by construction agents are unlikely to get more than one job in the first place, so there is not much to re-sequence. In general, we expect the added value of S to increase with the average number of jobs per agent. We note, moreover, that the added value of step S is at its largest in productive, sparse problems, which are the most difficult to solve in terms of Gurobi time.

The added value of step A is more immediately apparent. Comparing heuristic D to DA or DS to DSA indicates that in step A, taking routes that are good for coverage rather than generic shortest routes has a significant amount of added value. However, because step M is skipped, the coverage profiles are based on earlier movement, and the earlier movement still sends everyone back to the depot as quickly as possible, meaning that near the end of the time horizon, only one of the agents bunched up at the depot is given a coverage instruction in step A. Indeed, we see that the difference between DSA and MDSA becomes larger as the problem becomes denser and less productive, meaning agents bunch up at the depot earlier. Therefore, the difference between MDS and MDSA is much smaller: because of step M, agents have prioritised staying as well positioned in a sub-region as possible over being back at the depot as quickly as possible.

This may well explain the large added value of step M: it gives agents a way to use their remaining time more fruitfully than bunching up at the depot, and this way of spending remaining time is clearly optimal as the amount of remaining time goes to infinity. However, this alone is not a sufficient explanation for the success of MDSA, as WAM operates by the same logic but does not perform as well. Firstly, it must be reiterated that WAM does not always succeed at finding a feasible solution because it may not always be possible to actually visit a median in the length of a shift; in such cases, MDSA is capable of merely approaching the median as best as it can, among other options. More importantly, it appears that agents in WAM have less 'restraint' in selecting jobs; job selection in MDSA seeks to keep agents as close as possible to their respective medians, where agents in WAM are encouraged to take on jobs anywhere if they can reduce the total amount of distance travelled this way, even if this means taking on jobs that are closer to the median of someone else. This can create unbalanced situations where some agents move around a lot while others get no jobs; the agents without jobs can cover their neighbourhood well, but other parts of the network can become deserted because 'their' agent is out doing jobs everywhere. This argument is supported by the fact that WAM performs worst for productive, dense problems, where there exists most opportunity for agents to take on many jobs and spend time away from 'their' median, thus creating poorly covered parts of the network.

We conclude that MDSA is likely a successful heuristic for two reasons: step M allows agents to spend their remaining time in a way that is asymptotically optimal, as opposed to spending it bunched up at the depot; and dividing jobs by distance to medians rather than by total amount of distance travelled better ensures that no part of the network is completely deserted for the sake of jobs, unless absolutely necessary.

As a point for improvement, it deserves noting that while MDSA finds solutions with near-optimal values, it only succeeds in finding a completely optimal solution in 49 out of 406 instances, 22 of which are in the 'easiest' class $\mathcal{I}_4$. In fact, the partitioning of jobs over agents found by MDSA only corresponds with the partitioning in the optimal solution in 125 out of 406 instances. Further improvements to the heuristic may be achieved by better understanding the logic by which jobs should be divided over agents.

The success of MDSA prompted us to see if the produced solutions are useful for warm-starting: that is, to see if the MIP running time could be significantly reduced by offering a good feasible solution to start from. For the easier problem instances, this is not the case: apparently, the overhead of running MDSA and feeding this solution into Gurobi is hardly worth the saved Gurobi run time, which is often already in the order of a few seconds. For the difficult classes $\mathcal{I}_5$, $\mathcal{I}_7$ and $\mathcal{I}_R$, however, the average computation time reduction is quite significant. The average computation time in $\mathcal{I}_5$ goes down from 8739.1 seconds to 4949.7 seconds, which is a reduction of 43.4%. $\mathcal{I}_7$ drops 81.8% from

8390.3 seconds to 1523.1 seconds, and $\mathcal{I}_R$ drops 18.7% from 1202.7 seconds to 977.5 seconds. These average savings are largely incurred by the hardest instances suddenly being much easier to solve. There exist instances, however, where warm-starting increases the total run time by more than twenty minutes, and it is hard to predict whether warm-starting will be helpful for any given instance. This could be dealt with by solving instances in parallel, where one thread uses a warm start and the other does not, and terminating when either finish. On average, though, warm-starting seems very useful for difficult instance classes.

Looking at Table 4, we can see that the solutions produced by the different methods have structural differences. Comparing the hops between D and opt, we could observe that 'caring about coverage' more than doubles the number of hops. Similarly, comparing Optimal jobless and opt, we could observe that 'caring about jobs' increases emergency response time by 11.5%. However, in making such comparisons, we implicitly assume that there are $|\mathcal{A}|$ agents available solely for jobs or for coverage, while in the opt-solutions, we only have $|\mathcal{A}|$ agents available in total for both jobs and coverage. It is more fair, therefore, to compare these metrics against the Split fleet-solutions. Then, we can observe that joining the coverage fleet and the job fleet into one increases the number of hops by 69.9%, but we also cut our primary objective of expected response time by 28.3%. The increase in hops is explained as follows: in a split fleet, we would run D on half the agents and Optimal jobless on the other, which are both very efficient methods with respect to hops, as Optimal jobless-solutions mainly go to medians and back. The improvement in response time is due to having twice as many agents available to be spread for effective emergency response. We remark that this trade-off between increased travel costs and decreased emergency response costs is less extreme when running MDSA instead of opt, and least extreme when running WAM instead of opt. Following the same trend, opt-solutions are most 'jittery' with respect to the average number of shortcuts, MDSA-solutions are less jittery and WAM-solutions are least jittery among these three. D is most efficient with respect to hops and shortcuts, but offers even poorer emergency response than splitting the fleet does.

As a final observation, we acknowledge that the average running time of WAM sees a remarkable spike in class $\mathcal{I}_7$. Class $\mathcal{I}_7$ has the largest decision space for WAM, as medium, productive problems have most variables, while dense problems have more feasible solutions than sparse ones. This, apparently, is reflected in Gurobi needing much more time to verify optimality of found routings. We presume that this difficulty is not observed in step D because D does not demand that each route has exactly one median, and because step M makes routes more distinct in solution value. As WAM is typically outperformed by MDSA, resolving this issue was not further considered in this research.

7. Conclusions

The problem of scheduling non-urgent jobs in a way that provides optimal emergency response time has received little academic attention, although it is an interesting problem from both academic and applied viewpoints.

We proposed the Median Routing Problem as a mathematical model for this problem. An optimal solution method using mixed-integer linear programming was presented, as well as several heuristics. Of these, MDSA typically worked best: on the benchmark instances, it found solutions that were on average 3.2% away from optimal in 2.4 seconds, where solving them with a MIP solver takes 39 minutes. Using the MDSA solution to warm-start the MIP is especially useful for the most difficult problem instances.

By studying partial versions of MDSA under parametric variation in the benchmark instances, and comparing it with the simpler WAM-algorithm, our experiments suggest that the success MDSA is due to two factors, both of which rely on the MEDIATE-step. Firstly, the MEDIATE-step allows agents to spend their remaining time in a way that is asymptotically optimal as the amount of free time goes to infinity. Secondly, creating coverage subregions and trying to divide jobs as well as possible into these subregions helps guarantee that no subregion is ever completely abandoned for the sake of jobs, unless absolutely necessary. The speed of MDSA was due to first taking the most restrictive decisions in a very compressed decision space, then making the best out of those decisions in pseudo-polynomial time. Though MDSA contains NP-hard subroutines, this was not yet found to be a bottleneck, even for the practical case study instances. Studying marginal contributions showed that the MEDIATE-step contributed most; it is essential to both success factors of MDSA.

Finally, we observed that combining an emergency response fleet and a job processing fleet into one joint fleet produced a reduction in expected emergency response time of 28.3%. This comes at the cost, however, of an increase in total travelled distance, as the optimal solutions for the separated problems are typically very efficient in travelled distance. This trade-off is strongest in optimal solutions, then in MDSA-solutions, and mildest in WAM-solutions.

As venues for future research, we propose the following. Our model discretises space and time so that joint coverage can be measured tractably, but other strategies than discretisation may exist. MDSA contains NP-hard subroutines, and for (much) larger instances, a fully polynomial version may be needed. As mentioned earlier, better understanding the logic by which optimal solutions cluster jobs may lead to even stronger heuristics. The single-agent case of MRP is much easier to analyse, as its coverage profile is always known, and interesting solution properties may be discovered. Many flavours of heuristics have not yet been explored, including meta-heuristics. Many natural model extensions can be made to the MRP, including time windows, multi-agent jobs and time-varying emergency probabilities. We have studied a 'single coverage' problem, and it merits research how to well prepare for more emergencies than just the next one. Perhaps most importantly, the set of jobs to be processed was given as a hard constraint: it is an interesting and non-trivial question to decide which jobs will be selected as input for a series of shifts, and what role multi-objective optimisation can play in weighing responsiveness against productivity.

Acknowledgements

We thank our partner in the railway industry for co-funding this research and supplying case study data. We thank the three anonymous reviewers for their valuable suggestions.

Appendix A. Example instance

An example of MRP is illustrated in Fig. A.3. In this example, we may move two agents discretely over a network. They both start and end their work shift at a central node, which acts as a classical 'depot'. At some of the nodes, an emergency may occur; each of these nodes has a probability P_v of being the site of the next emergency. There are also two jobs to be processed at specific locations in the 'right half' of the network, each with a processing time of 3. A feasible solution tells all agents where to be at each time-step and when to start processing which jobs. The goal is to move the agents such that the weighted distance of all nodes to their nearest agent is minimised, where weights may represent emergency probabilities and distances may represent response times, while ensuring that all jobs get done. In the optimal solution, presented in Fig. A.4, agent 1 processes both jobs, thus giving 'coverage' to

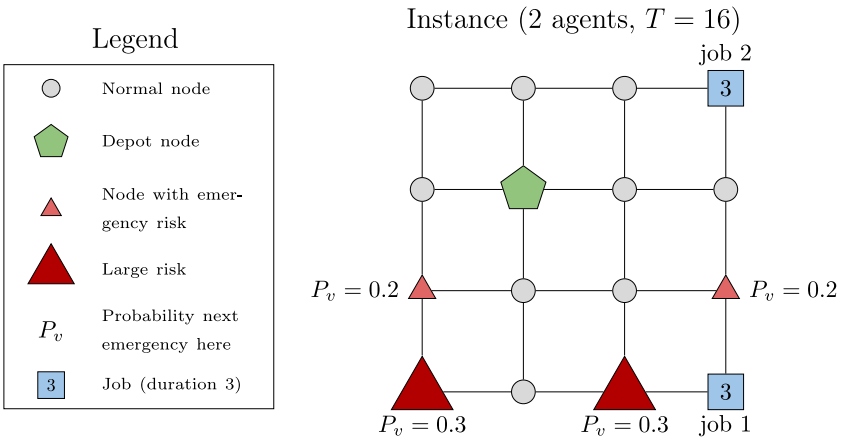
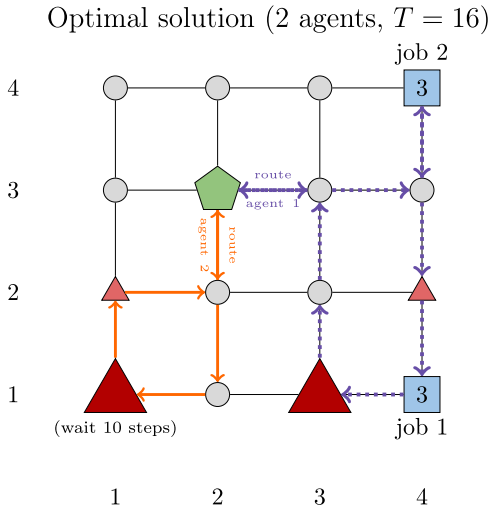


Fig. A.3. An example instance of MRP. Two agents, starting at a depot at $t = 0$, must process all jobs and be back at $t = 16$, whilst minimising average expected emergency response time. The optimal solution is presented in Fig. A.4.



Time-step	Location agent 1	Location agent 2	Expected response time
$t = 0$	(2,3), depot	(2,3), depot	2.8
$t = 1$	(3,3)	(2,2)	1.8
$t = 2$	(4,3)	(2,1)	1.2
$t = 3$	(4,4), start job 2	(1,1), wait	1.2
$t = 4$	(4,4), process job 2	(1,1), wait	1.2
$t = 5$	(4,4), process job 2	(1,1), wait	1.2
$t = 6$	(4,4), job 2 just finished	(1,1), wait	1.2
$t = 7$	(4,3)	(1,1), wait	1
$t = 8$	(4,2)	(1,1), wait	0.8
$t = 9$	(4,1), start job 1	(1,1), wait	0.7
$t = 10$	(4,1), process job 1	(1,1), wait	0.7
$t = 11$	(4,1), process job 1	(1,1), wait	0.7
$t = 12$	(4,1), job 1 just finished	(1,1), wait	0.7
$t = 13$	(3,1)	(1,1)	0.6
$t = 14$	(3,2)	(1,2)	0.8
$t = 15$	(3,3)	(2,2)	1.8
$t = 16$	(2,3), depot	(2,3), depot	2.8
Average			1.247 (optimal)

Fig. A.4. The optimal solution to the instance in Fig. A.3. It is optimal to let agent 1 (purple, dotted) process both jobs and be responsible for the 'right half' of the network, and to put agent 2 (orange, solid) in a good position to respond to emergencies in the 'left half'. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the right half of the network; meanwhile, agent 2 processes no jobs but moves to a good position to give coverage to the 'left half' of the network. Both agents make sure to be back at the depot just at the end of the shift.

In the optimal solution, at $t = 3$, there is one agent at (1,1) and one agent at (4,4). This means that the expected response time (assuming Manhattan distances) for an emergency at $t = 3$ equals

$$0.3 \cdot 0 + 0.3 \cdot 2 + 0.2 \cdot 1 + 0.2 \cdot 2 = 1.2$$

as this is, summed over the four possible emergency locations, the probability of the emergency occurring there times the distance of the nearest agent.

The solution in Fig. A.4 is feasible because it visits all jobs and everyone is back in time; it is optimal because it has minimal average expected response time among feasible solutions.

Note that it is by no means necessary in MRP that the start and end location for all agents are the same: this is simply the case for this example. In fact, allowing any node to be an agent's start location is necessary to allow for dynamic re-solving after an emergency occurs. Also, in this example, the response time between nodes is equal to the number of steps needed to get there via the graph; in general, response times may follow a different metric.

Appendix B. MIP for routing subroutine of WAM

In step 1 of WAM, as described in Algorithm 1, several medians are identified as points of interest. In step 2, a distance matrix D is obtained. In step 3, the agents are routed over the jobs and these medians in such a way that the amount of total travelling time is minimised, so that the total amount of time that can be spent at these medians is maximised. Aside from the standard condition that all jobs and medians must be visited exactly once, each agent must visit exactly one median. If there were one location where all agents start and end, and the number of medians visited per agent were allowed to be arbitrary, then this would simply be an instance of DVRP. However, these two side constraints merit the explicit description of the MIP used to solve this subroutine.

Define a set $\mathcal{S} = \{s_1, \dots, s_{|\mathcal{A}|}\}$ of virtual nodes, representing the start 'locations' of the agents. Similarly, define $\mathcal{E} = \{e_1, \dots, e_{|\mathcal{A}|}\}$ the virtual end locations of the agents. For any distinct pair $i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J}$, $j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}$, let variable $w_{ij} \in \{0, 1\}$ indicate whether or not someone goes directly to j after visiting i. For each $j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}$, define a variable $f_j \in [0, T]$ describing the arrival time at j; for each $j \in \mathcal{S}$, denote $f_j = 0$. For $j \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{E}$, denote 'processing time' $Q_j = 0$. Finally, for each $a \in \mathcal{A}$ and each $j \in \mathcal{M} \cup \mathcal{J}$, define a variable $z'_{aj} \in \{0, 1\}$ indicating whether or not j is visited by a. For $a_i \in \mathcal{A}$, $s \in \mathcal{S}$ and $e \in \mathcal{E}$, denote $z'_{a_i s} = 1$ if $s = s_i$ and 0 otherwise, and $z'_{a_i e} = 1$ if $e = e_i$ and 0 otherwise: in other words, encode that start point s_i and end point e_i are always assigned to agent a_i. Then the following MIP produces the optimal sequences:

$$\begin{aligned} \min \quad & \textstyle\sum_{i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J}} \sum_{j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}} D_{ij} w_{ij} \\ s.t. \ & \textstyle\sum_{j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}} w_{ij} = 1 \quad (\forall i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J}) \end{aligned} \tag{B.1}$$

$$\textstyle\sum_{i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J}} w_{ij} = 1 \quad (\forall j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}) \tag{B.2}$$

$$f_j \geq f_i + Q_i + D_{ij} - T \cdot (1 - w_{ij}) \quad (\forall i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J})(\forall j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}) \tag{B.3}$$

$$\textstyle\sum_{m \in \mathcal{M}} z'_{am} = 1 \quad (\forall a \in \mathcal{A}) \tag{B.4}$$

$$\textstyle\sum_{a \in \mathcal{A}} z'_{aj} = 1 \quad (\forall j \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}) \tag{B.5}$$

$$\begin{aligned} z'_{aj} \geq z'_{ai} + w_{ij} - 1 \quad & (\forall a \in \mathcal{A})(\forall i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J})(\forall j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}) \\ w_{ij}, z'_{aj} \in \{0, 1\}, \ \ & f_j \in [0, T] \end{aligned} \tag{B.6}$$

Here, the objective value equals the total number of time-steps spent travelling. Constraints (B.1) state that, aside from end points, each node must be departed from exactly once. Constraints (B.2) state that, aside from start points, each node must be visited exactly once. Constraints (B.3) have two functions: they recursively ensure that agents are at their end points no later than T, and act as subtour elimination constraints. Constraints (B.4) state that each agent must be assigned exactly one median. Constraints (B.5) state that each node must be assigned to exactly one agent. Constraints (B.6) state that if $i \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{J}$ is assigned to some agent $a \in \mathcal{A}$, and i is followed up by some $j \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{J}$, then j is assigned to a as well. These last three rules, combined with the fact that each starting point is assigned to exactly one agent, ensure that each median is assigned to exactly one agent. Furthermore, constraints (B.4) and constraints (B.5) ensure that the route that starts at s_a also ends in the right e_a.

When this MIP is solved with a MIP solver, the values of w_{ij} describe the sequences that minimise the time spent travelling, thus creating a maximal amount of time that can be spent waiting at medians.

Appendix C. Solving path-TSP

Observe a node set $\mathcal{J}_a$, a start node S_a, an end node E_a and a symmetric distance matrix D_{ij}. For brevity, denote $\mathcal{J}^+ = \{S_a\} \cup \mathcal{J}_a \cup \{E_a\}$. For each distinct $i \in \mathcal{J}^+$, $i \neq E_a$ and $j \in \mathcal{J}^+$, $j \neq S_a$, define a variable $w_{ij} \in \{0, 1\}$ describing whether or not j is the next node visited after i. Also define a variable $u_j \in [1, |\mathcal{J}^+| - 1]$, indicating how many nodes have been visited before j. Denote $u_{S_a} = 0$. The path-TSP, to find the shortest path that starts at S_a, ends at E_a and visits all nodes exactly once, can be solved by the following MIP.

$$\begin{aligned} \min \quad & \textstyle\sum_{i \in \mathcal{J}^+, i \neq E_a} \sum_{j \in \mathcal{J}^+, j \neq S_a} D_{ij} w_{ij} \\ s.t. \ & \textstyle\sum_{j \in \mathcal{J}^+, j \neq S_a} w_{ij} = 1 \quad (\forall i \in \mathcal{J}^+, i \neq E_a) \end{aligned} \tag{C.1}$$

$$\textstyle\sum_{i \in \mathcal{J}^+, i \neq E_a} w_{ij} = 1 \quad (\forall j \in \mathcal{J}^+, j \neq S_a) \tag{C.2}$$

$$\begin{aligned} u_j \geq u_i + w_{ij} - 1 \quad & (\forall i \in \mathcal{J}^+, i \neq E_a)(\forall j \in \mathcal{J}^+, j \neq S_a) \\ w_{ij} \in \{0, 1\}, \ \ & u_j \in [1, |\mathcal{J}^+| - 1] \end{aligned} \tag{C.3}$$

Here, the objective value describes the total amount of distance travelled. Constraints (C.1) state that each node, except the end node, must be departed from exactly once. Constraints (C.2) state that each node, except the start node, must be visited exactly once. Constraints (C.3) act as subtour elimination constraints.

After solving this MIP with a MIP solver, reading out w_{ij} produces a sequence with minimal time spent travelling.

Appendix D. Cheapest space-time path for sequence and node costs

The dynamic program Algorithm 3 takes as input some start point S_a, some end point E_a, some sequence over job set $\mathcal{J}_a$ and some cost function on the nodes that may depend on the time-step. It returns the cheapest feasible space-time path with respect to these node costs.

Algorithm 3 Optimal execution of job sequence against given node costs

1: Initialise the set of active nodes $\mathcal{N} := \{(depot, 0, j_0)\}$, with j_0 the first job that has to be visited, or $j_0 = E_a$ if the sequence has no jobs;
2: Initialise the reach cost $reach : \mathcal{N} \to \mathbb{R}_{\geq 0}$ as $reach(S_a, 0, j_0) = 0$;
3: Initialise the explored nodes $\mathcal{E} := \emptyset$;
4: Take $(u, t, state) := \arg\min_{\mathcal{N}} reach(u, t, state)$, remove this from $\mathcal{N}$ and add it to $\mathcal{E}$;
5: Observe each 'neighbour' $(v, t+1, state')$ of $(u, t, state)$. $(v, t+1, state')$ is a neighbour of $(u, t, state)$ if $v \in \mathcal{V}_u$, and either:
- $state = j \in \{E_a\} \cup \mathcal{J}_a$ and $state' = j$ (the goal does not change);
- $state = j \in \mathcal{J}$, $L_j = v$ and $state' = (j, Q_j)$ (the next job is started);
- $state = (j, q)$, $q > 1$, $u = v$ and $state' = (j, q-1)$ (the job is processed further);
- $state = (j, 1)$, $u = v$, j' is the next job if there is one and E_a otherwise, and $state' = j'$ (the job is finished, the next goal retrieved);
- $state = (j, 1)$, $u = v$, j' is the next job to be visited, $L_{j'} = v$ and $state' = (j', Q_{j'})$ (the job is finished, and the next job happens to be in the same location).

6: For each neighbour $(v, t+1, state')$, if it is not already in $\mathcal{N}$ or $\mathcal{E}$, add it to $\mathcal{N}$ and set $reach(v, t+1, state') = reach(u, t, state) + nodecost_a(v, t+1)$;
7: If the end node $(E_a, T, E_a) \notin \mathcal{N}$, go to step 4;
8: Using *reach*, backtrack the cheapest path to (E_a, T, E_a) from $(S_a, 0, j_0)$.

Appendix E. Instance generator

To create an instance of the MRP, the following parameters should be provided:

- The desired number of agents, $|\mathcal{A}|$;
- The desired number of jobs, $|\mathcal{J}|$;
- The desired number of nodes, $|\mathcal{V}|$;
- The desired number of time-steps, T;
- The width of a large square, *SCALE*;
- The threshold for adjacency, *G_CHOP*;
- The minimal distance between nodes (for visualisation and realism purposes), *MIN_DIST*;
- The range of potential processing times for jobs, *Q_RANGE*;
- The probabilities that a given node has large, medium or small emergency demand, *PROB_LARGE* and *PROB_MEDIUM* and *PROB_SMALL*;
- The range of how many 'emergency points' nodes with large demand can have, *P_POINTS_RANGE_LARGE*;
- The range of how many 'emergency points' nodes with medium demand can have, *P_POINTS_RANGE_MEDIUM*;
- The range of how many 'emergency points' nodes with small demand can have, *P_POINTS_RANGE_SMALL*;
- The allowed number of attempts to generate a feasible instance, *ATTEMPTS*.

Given these parameters, an instance of MRP is generated using Algorithm 4.

Algorithm 4 Generate an instance of MRP

1: Set $attempt = 0$;
2: Set $attempt := attempt + 1$. If $attempt > ATTEMPTS$, QUIT;
3: Create $|\mathcal{V}| - |\mathcal{J}|$ nodes with uniformly random integer X and Y coordinates between 0 and *SCALE*;
4: Create the other $|\mathcal{J}|$ nodes with uniformly random integer X and Y coordinates between 0 and *SCALE*. At each of these nodes, place one job, thus setting $L : \mathcal{J} \to \mathcal{V}$. For each job, pick its processing time uniformly randomly from *Q_RANGE*;
5: Check if none of the nodes are closer than *MIN_DIST* to each other in Euclidean norm (truncated to two decimals). If there exist nodes that are too close together, go to Step 2;
6: Determine neighbour sets $\mathcal{V}_v$ by checking, for each pair of nodes (u, v), whether or not their Euclidean distance (truncated to two decimals) is under *G_CHOP*;
7: Check if the graph implied by $\mathcal{V}_v$ is connected, by picking an arbitrary node and checking neighbours in a width-first search. This width-first search terminates before finding all nodes if and only if the graph is not connected. If the graph is not connected, go to Step 2;
8: Check if the triangle inequality still holds after truncating Euclidean distances to two decimals, by checking for each $u, w \in \mathcal{V}$ whether or not $C_{uw} \leq \min_{v \in \mathcal{V}}\{C_{uv} + C_{vw}\}$. If there exists a pair where this inequality does not hold, go to Step 2;
9: Denote *depot* the centermost non-job node, that is, the non-job node with smallest Euclidean distance to $(0.5 \cdot SCALE, 0.5 \cdot SCALE)$;
10: Determine the distance matrix between the job nodes and *depot* using the Floyd-Warshall algorithm (Cormen, Leiserson, Rivest, & Stein, 2009);
11: Check whether this instance of MRP admits a feasible solution, by checking whether the DVRP over the jobs and *depot* has a feasible solution, for example by solving the MIP describing the DVRP but with objective function 0. If the DVRP is infeasible, go to Step 2;
12: Set C_{uv} equal to the Euclidean distance between each pair of nodes (u, v);
13: For each non-job, non-depot node v, first determine whether v has a large, medium or small emergency demand against probabilities *PROB_LARGE*, *PROB_MEDIUM* and *PROB_SMALL*. Then, uniformly randomly draw a number of 'emergency points' from *P_POINTS_RANGE_LARGE* or *P_POINTS_RANGE_MEDIUM* or *P_POINTS_RANGE_SMALL*.
14: Set P_v for all $v \in \mathcal{V}_P$ by normalising until $\sum_{v \in \mathcal{V}_P} P_v = 1$;
15: Create $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$, set $S_a = depot$ and $E_a = depot$ and RETURN this instance.

The instances of class $\mathcal{I}_1$ were generated with the following parameters:

$|\mathcal{A}| = 3$, $|\mathcal{J}| = 2|\mathcal{A}| + 1$, $|\mathcal{V}| = 20$,
$T = 16$, $ATTEMPTS = 100$,
$MIN\_DIST = 1$,
$P\_POINTS\_RANGE\_LARGE = \{100, \ldots, 300\}$,
$P\_POINTS\_RANGE\_MEDIUM = \{10, \ldots, 50\}$,
$P\_POINTS\_RANGE\_SMALL = \{0, \ldots, 5\}$,
$G\_CHOP = 300$,
$SCALE = 1000$,
$Q\_RANGE = \{0, 1, 2\}$,
$PROB\_LARGE = 0.15$,
$PROB\_MEDIUM = 0.30$,
$PROB\_SMALL = 0.55$

The class $\mathcal{I}_5$ instances were generated with identical parameters, except $|\mathcal{A}| = 4$ and $|\mathcal{V}| = 100$.

As mentioned in Table 2, the classes $\mathcal{I}_1$ and $\mathcal{I}_5$ contain instances that are 'productive' and 'sparse'. The 'dense' instances were created from the 'sparse' instances by recomputing the node adjacencies for $G\_CHOP = 600$. The 'unproductive' classes were created from the 'productive' classes by deleting jobs until $|\mathcal{J}| = |\mathcal{A}| - 1$. We chose to set $|\mathcal{J}|$ equal to either $2|\mathcal{A}| + 1$ or $|\mathcal{A}| - 1$, rather than

$2|\mathcal{A}|$ or $|\mathcal{A}|$, to break symmetry and introduce the additional challenge of assigning unequal amounts of jobs to agents.

Appendix F. Numerical results

This section presents the full result of testing the various MRP methods on the various MRP benchmark instance classes. For ease of reading, we have bold-faced some of the table entries that we discuss in Section 6: most notably, the best performing heuristics with respect to average solution quality and worst solution quality, and the computation times of the MIP and MDSA.

Table F.5
Results of applying the ten solution methods on the 6 real-life instances.

Method	Number of feasible solutions	Average gap (%)	Average time (s)	Worst gap (%)	Worst time (s)
$\mathcal{I}_R$	(real-life instances)				
opt	6	0	**1202.7**	0	2871
WAM	6	5.9	14.0	9.3	62.9
D	6	27.0	0.4	40.2	0.9
MD	6	4.4	3.7	8.0	4.3
DS	6	27.4	0.7	40.2	2.0
DA	6	13.6	0.8	24.6	1.3
MDS	6	4.4	3.9	7.4	4.4
MDA	6	**3.1**	4.4	**4.9**	5.3
DSA	6	12.3	0.9	24.6	1.4
MDSA	6	3.4	**4.5**	7.4	5.5

Table F.6
Results of applying the ten solution methods on the 200 small problems.

Method	Number of feasible solutions	Average gap (%)	Average time (s)	Worst gap (%)	Worst time (s)
$\mathcal{I}_1$	(small, productive, sparse)				
opt	50	0	**29.1**	0	353
WAM	50	20.7	1.2	106.0	12.8
D	50	102.0	0.3	241.6	0.4
MD	50	9.3	0.4	35.3	0.8
DS	50	101.7	0.4	238.0	0.6
DA	50	30.0	0.3	85.3	0.5
MDS	50	8.4	0.6	**34.2**	0.9
MDA	50	8.0	0.5	35.3	0.8
DSA	50	32.9	0.4	85.3	0.6
MDSA	50	**7.0**	**0.6**	**34.2**	0.9
$\mathcal{I}_2$	(small, unproductive, sparse)				
opt	50	0	**2.7**	0	9
WAM	50	10.5	0.3	80.8	0.5
D	50	167.2	0.1	314.6	0.1
MD	50	4.4	0.2	**25.7**	0.3
DS	50	167.2	0.1	314.6	0.2
DA	50	53.8	0.1	162.5	0.2
MDS	50	4.4	0.3	**25.7**	0.4
MDA	50	**3.5**	0.2	**25.7**	0.3
DSA	50	55.0	0.1	162.5	0.2
MDSA	50	**3.5**	**0.3**	**25.7**	0.5
$\mathcal{I}_3$	(small, productive, dense)				
opt	50	0	**5.7**	0	48
WAM	50	29.4	3.2	140.5	16.6
D	50	180.7	0.2	314.5	0.4
MD	50	5.4	0.4	45.9	0.6
DS	50	180.6	0.3	314.5	0.5
DA	50	63.4	0.3	160.7	0.5
MDS	50	5.3	0.5	43.2	0.7
MDA	50	5.0	0.5	45.9	0.7
DSA	50	64.4	0.4	158.8	0.6
MDSA	50	**4.3**	**0.6**	**43.0**	0.8
$\mathcal{I}_4$	(small, unproductive, dense)				
opt	50	0	**0.7**	0	5
WAM	50	8.0	0.3	56.6	0.6
D	50	258.9	0.1	520.5	0.1
MD	50	2.4	0.3	24.5	0.4
DS	50	258.9	0.1	520.5	0.2
DA	50	107.9	0.2	281.9	0.2
MDS	50	2.4	0.3	24.5	0.5
MDA	50	**2.0**	0.3	**24.2**	0.4
DSA	50	106.5	0.2	284.5	0.3
MDSA	50	**2.0**	**0.4**	**24.2**	0.5

Table F.7
Results of applying the ten solution methods on the 200 medium-sized problems.

Method	Number of feasible solutions	Average gap (%)	Average time (s)	Worst gap (%)	Worst time (s)
$\mathcal{I}_5$	(medium, productive, sparse)				
opt	50	0	**8739.1**	0	132750
WAM	50	7.7	11.4	24.0	101.8
D	50	60.5	0.5	82.3	0.8
MD	50	4.7	3.2	14.5	7.1
DS	50	60.4	0.7	83.0	1.1
DA	50	30.3	0.9	49.8	1.6
MDS	50	4.5	3.4	14.6	7.3
MDA	50	3.7	3.9	14.4	8.3
DSA	50	29.4	1.0	54.0	1.6
MDSA	50	**3.5**	**4.1**	**14.3**	8.1
$\mathcal{I}_6$	(medium, unproductive, sparse)				
opt	50	0	**1476.1**	0	12839
WAM	50	4.6	2.4	21.7	3.6
D	50	88.0	0.1	115.6	0.3
MD	50	2.3	2.7	18.7	3.3
DS	50	88.1	0.2	115.6	0.3
DA	50	51.1	0.5	80.8	0.8
MDS	50	2.0	2.7	12.1	3.3
MDA	50	**1.4**	3.2	**11.9**	4.1
DSA	50	51.1	0.5	80.8	0.8
MDSA	50	1.5	**3.3**	**11.9**	3.9
$\mathcal{I}_7$	(medium, productive, dense)				
opt	50	0	**8390.3**	0	72703
WAM	50	11.3	**438.5**	35.1	4749.7
D	50	73.9	1.1	101.9	6.0
MD	50	3.9	3.6	14.3	5.0
DS	50	74.0	1.2	101.9	6.1
DA	50	39.1	1.8	63.4	7.0
MDS	50	4.0	3.9	14.1	5.2
MDA	50	**2.8**	5.1	13.8	7.8
DSA	50	38.4	1.9	63.4	7.2
MDSA	50	**2.8**	**5.4**	**13.3**	7.8
$\mathcal{I}_8$	(medium, unproductive, dense)				
opt	50	0	**168.3**	0	2963
WAM	50	4.1	2.6	12.6	3.5
D	50	100.1	0.2	128.1	0.6
MD	50	1.3	3.5	6.1	3.8
DS	50	100.0	0.2	128.1	0.4
DA	50	58.8	1.0	103.4	1.2
MDS	50	1.3	3.5	6.1	4.0
MDA	50	**0.9**	4.5	**5.1**	6.6
DSA	50	59.0	1.0	112.6	1.3
MDSA	50	**0.9**	**4.6**	**5.1**	5.2

References

Agarwal, A., Hiot, L. M., Joo, E. M., & Nghia, N. T. (2007). Rectilinear workspace partitioning for parallel coverage using multiple unmanned aerial vehicles. *Advanced Robotics, 21*(1-2), 105–120.

Almoustafa, S., Hanafi, S., & Mladenović, N. (2013). New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research, 226*(3), 386–394.

Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., & Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing, 33*(3), 544–562.

Başdere, M., & Bilge, Ü. (2014). Operational aircraft maintenance routing problem with remaining time consideration. *European Journal of Operational Research, 235*(1), 315–328.

van den Berg, P., & Van Essen, T. (2019). Scheduling non-urgent patient transportation while maximizing emergency coverage. *Transportation Science, January 2019*.

Bertsimas, D. J., & Van Ryzin, G. (1993). Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles. *Operations Research, 41*(1), 60–76.

Brotcorne, L., Laporte, G., & Semet, F. (2003). Ambulance location and relocation models. *European Journal of Operational Research, 147*(3), 451–463.

Caunhye, A. M., Nie, X., & Pokharel, S. (2012). Optimization models in emergency logistics: A literature review. *Socio-Economic Planning Sciences, 46*(1), 4–13.
Charikar, M., Guha, S., Tardos, É., & Shmoys, D. B. (2002). A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences, 65*(1), 129–149.
Cohn, A. M., & Barnhart, C. (2003). Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research, 51*(3), 387–396.
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.
Daskin, M. S., & Maass, K. L. (2015). The p-median problem. In *Location Science* (pp. 21–45). Springer.
De Dinechin, F., & Istoan, M. (2015). Hardware implementations of fixed-point ATAN2. In *Proceedings of the IEEE 22nd Symposium on Computer Arithmetic* (pp. 34–41). IEEE.
Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271.
Doitsidis, L., Weiss, S., Renzaglia, A., Achtelik, M. W., Kosmatopoulos, E., Siegwart, R., & Scaramuzza, D. (2012). Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. *Autonomous Robots, 33*(1-2), 173–188.
Drexl, M., & Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research, 241*(2), 283–308.
van Ee, M., & Sitters, R. (2014). Routing under uncertainty: the a priori traveling repairman problem. In *Proceedings of the international workshop on approximation and online algorithms* (pp. 248–259). Springer.
Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering, 57*(4), 1472–1483.
Farahani, R. Z., & Hekmatfar, M. (2009). *Facility location: Concepts, models, algorithms and case studies*. Springer.
Fontecha, J. E., Guaje, O. O., Duque, D., Akhavan-Tabatabaei, R., Rodríguez, J. P., & Medaglia, A. L. (2020). Combined maintenance and routing optimization for large-scale sewage cleaning. *Annals of Operations Research, 286*, 441–474.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Freeman.
Goodson, J. C., Ohlmann, J. W., & Thomas, B. W. (2013). Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research, 61*(1), 138–154.
Gopalan, R., & Talluri, K. T. (1998). The aircraft maintenance routing problem. *Operations Research, 46*(2), 260–271.
Haouari, M., Shao, S., & Sherali, H. D. (2012). A lifted compact formulation for the daily aircraft maintenance routing problem. *Transportation Science, 47*(4), 508–525.
Hoogeveen, J. (1991). Analysis of Christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters, 10*(5), 291–295.
Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science, 34*(4), 426–438.
Irawan, C. A., Ouelhadj, D., Jones, D., Stålhane, M., & Sperstad, I. B. (2017). Optimisation of maintenance routing and scheduling for offshore wind farms. *European Journal of Operational Research, 256*(1), 76–89.
Jaillet, P., Bard, J. F., Huang, L., & Dror, M. (2002). Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science, 36*(3), 292–300.
Joubert, J. W. (2007). *An integrated and intelligent metaheuristic for constrained vehicle routing*. University of Pretoria.
Kergosien, Y., Gendreau, M., Ruiz, A., & Soriano, P. (2014). Managing a fleet of ambulances to respond to emergency and transfer patient transportation demands. In *Proceedings of the international conference on health care systems engineering* (pp. 303–315). Springer.
Kiechle, G., Doerner, K. F., Gendreau, M., & Hartl, R. F. (2009). Waiting strategies for regular and emergency patient transportation. In *Operations research proceedings* (pp. 271–276). Springer.
Koutsoupias, E. (2009). The k-server problem. *Computer Science Review, 3*(2), 105–118.
Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(1-2), 83–97.
Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(3), 345–358.
Laporte, G., Desrochers, M., & Nobert, Y. (1984). Two exact algorithms for the distance-constrained vehicle routing problem. *Networks, 14*(1), 161–172.
López-Santana, E., Akhavan-Tabatabaei, R., Dieulle, L., Labadie, N., & Medaglia, A. L. (2016). On the combined maintenance and routing optimization problem. *Reliability Engineering & System Safety, 145*, 199–214.
Maróti, G., & Kroon, L. (2005). Maintenance routing for train units: the transition model. *Transportation Science, 39*(4), 518–525.
Nagarajan, V., & Ravi, R. (2012). Approximation algorithms for distance constrained vehicle routing problems. *Networks, 59*(2), 209–214.
Owen, S. H., & Daskin, M. S. (1998). Strategic facility location: A review. *European Journal of Operational Research, 111*(3), 423–447.
Palma-Behnke, R., Benavides, C., Lanas, F., Severino, B., Reyes, L., Llanos, J., & Sáez, D. (2013). A microgrid energy management system based on the rolling horizon strategy. *IEEE Transactions on Smart Grid, 4*(2), 996–1006.
Penicka, M., Strupchanska, A. K., & Bjørner, D. (2003). Train maintenance routing. *Proceedings of the Symposium on formal methods for railway operation and control systems, FORMS*.
Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.
Pillac, V., Guéret, C., & Medaglia, A. L. (2012). An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems, 54*(1), 414–423.
Plane, D. R., & Hendrick, T. E. (1977). Mathematical programming and the location of fire companies for the denver fire department. *Operations Research, 25*(4), 563–578.
Raghavan, S., Sahin, M., & Salman, F. S. (2019). The capacitated mobile facility location problem. *European Journal of Operational Research, 277*(2), 507–520.
ReVelle, C. S., & Swain, R. W. (1970). Central facilities location. *Geographical Analysis, 2*(1), 30–42.
Sarac, A., Batta, R., & Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research, 175*(3), 1850–1869.
Shu, L., Wang, W., Lin, F., Liu, Z., & Zhou, J. (2013). A sweep coverage scheme based on vehicle routing problem. *Indonesian Journal of Electrical Engineering and Computer Science, 11*(4), 2029–2036.
Talluri, K. T. (1998). The four-day aircraft maintenance routing problem. *Transportation Science, 32*(1), 43–53.
Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*. SIAM.
Wang, Y., & Hussein, I. I. (2007). Cooperative vision-based multi-vehicle dynamic coverage control for underwater applications. In *Proceedings of the IEEE international conference on control applications* (pp. 82–87). IEEE.
Zenklusen, R. (2019). A 1.5-approximation for path TSP. In *Proceedings of the thirtieth annual ACM-SIAM symposium on discrete algorithms* (pp. 1539–1549). SIAM.
Zhang, S., Ohlmann, J. W., & Thomas, B. W. (2014). A priori orienteering with time windows and stochastic wait times at customers. *European Journal of Operational Research, 239*(1), 70–79.