Transportation Research Part C 157 (2023) 104401

Contents lists available at ScienceDirect

Transportation Research Part C

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

Reinforcement learning for humanitarian relief distribution with trucks and UAVs under travel time uncertainty ☆

Robert van Steenbergen *, Martijn Mes, Wouter van Heeswijk

University of Twente, Netherlands

ARTICLE INFO

Dataset link: https://data.mendeley.com/datasets/j36zgxnc5p/2

Keywords:
Humanitarian logistics
Last-mile relief distribution
Travel time uncertainty
UAVs
Reinforcement learning
Comparative analysis

ABSTRACT

Effective humanitarian relief operations are challenging in the aftermath of disasters, as trucks are often faced with considerable travel time uncertainties due to damaged transportation networks. Efficient deployment of Unmanned Aerial Vehicles (UAVs) potentially mitigates this problem, supplementing truck fleets in an impactful manner. To plan last-mile relief distribution in this setting, we introduce a multi-trip, split-delivery vehicle routing problem with trucks and UAVs, soft time windows, and stochastic travel times for last-mile relief distribution, formulated as a stochastic dynamic program. Within a finite time horizon, we aim to maximize a weighted objective function comprising the number of goods delivered, the number of different locations visited, and late arrival penalties. Our study offers insights into dealing with travel time uncertainty in humanitarian logistics by (i) deploying Unmanned Aerial Vehicles (UAVs) as partial substitutes for trucks, (ii) evaluating dynamic solutions generated by two deep reinforcement learning (RL) approaches – specifically value function approximation (VFA) and policy function approximation (PFA) – and (iii) comparing the RL solutions with solutions stemming from mathematical programming and dynamic heuristics. Experiments are performed on both Solomon-based instances and two real-world cases. The real-world cases – the 2015 Nepal earthquake and the 2018 Indonesia tsunami – are based on locally collected field data and real-world UAV specifications, and aim to provide practical insights. The experimental results show that dynamic decision-making improves both performance and robustness of humanitarian operations, achieving reductions in lateness penalties of around 85% compared to static solutions based on expected travel times. Furthermore, the results show that replacing half of the trucks with UAVs improves the weighted objective value by 11% to 56%, benefitting both reliability and location coverage. The results indicate that both the deployment of UAVs and the use of dynamic methods successfully mitigate travel time uncertainties in humanitarian operations.

1. Introduction

In the aftermath of major disasters such as floods or earthquakes, large amounts of relief goods – for example, tents, food, water, and medical supplies – are urgently required. Satisfying the needs of affected people in a disaster area is crucial during a relief operation, since a lack of relief goods may cause human suffering and even loss of life (Ahmadi et al., 2015). A recent study of Stumpf et al. (2022) reports that on average 74% of the total costs of these life-saving operations stem from the supply chain, with

☆ This article belongs to the Virtual Special Issue on "Transport Logistics".

* Corresponding author.
E-mail addresses: r.m.vansteenbergen@utwente.nl (R.M. van Steenbergen), m.r.k.mes@utwente.nl (M.R.K. Mes), w.j.a.vanheeswijk@utwente.nl (W.J.A. van Heeswijk).

https://doi.org/10.1016/j.trc.2023.104401
Received 13 February 2023; Received in revised form 8 September 2023; Accepted 26 October 2023
Available online 7 November 2023

transportation costs being the second-largest expenditure after the procurement of supplies. According to Van Wassenhove (2006, p. 476), logistics "is the most expensive part of any relief operation and the part that can mean the difference between a successful or failed operation". Hence, efficient and effective logistics is imperative for humanitarian operations. Unfortunately, road vehicles face large travel time uncertainties due to the state of transport networks after a disaster (Holguín-Veras et al., 2012), as roads may be damaged, destroyed, unaffected, or already repaired. In particular, large vehicles such as trucks often have difficulty accessing disaster areas (Balcik and Beamon, 2008). This paper studies dynamic routing methods and the deployment of Unmanned Aerial Vehicles (UAVs) as potential solutions to this problem.

To deal with uncertain parameters in disaster operations management, two-stage stochastic programs are the most widely applied models (Liberatore et al., 2013; Hoyos et al., 2015). These programs typically assume that, after realizing uncertain parameters, the disaster area remains static in terms of travel times and demand. However, the existing uncertainties and information updates in real-world disaster areas (e.g., road blockages/repairs, population movements, or aftershocks) suggest that dynamic and stochastic elements (e.g., real-time information) should be incorporated in dynamic routing (Holguín-Veras et al., 2012; Besiou and Van Wassenhove, 2020; Anuar et al., 2021).

Besiou and Van Wassenhove (2020) state that research should not only put more emphasis on the dynamically changing conditions of a disaster area, but also on the use of new technologies, such as drones, big data, and artificial intelligence. Humanitarian cargo UAVs offer operational flexibility due to their speed and ability to drop goods at hard-to-reach places, while providing opportunities to increase operational cost-effectiveness (Rabta et al., 2018; Rejeb et al., 2021). The deployment of cargo UAVs enables equity and inclusivity for those who otherwise are less likely to receive aid, thus improving the humanitarian principles of neutrality, impartiality, independence, and humanity (Van Wynsberghe and Comes, 2020). With this work, we collaborate with the Wings For Aid foundation. The specifications of the UAVs in our experiments are based on their MiniFreighter aircraft (Bamsey, 2021), which has a range of 500 km, a cruise speed of 125 km/h, and can carry 160 kg of relief goods. These characteristics allow the aircraft to supplement truck fleets operating in damaged transport networks.

This study provides insight into the influence of stochastic travel times in humanitarian logistics and analyzes UAV deployment and different planning strategies to mitigate the impact of travel time variability. We study a dynamic multi-vehicle, multi-trip, split-delivery vehicle routing problem with soft time windows and stochastic travel times, formulated as a stochastic dynamic program for sequential decision-making. Adopting a weighted objective function, we aim to (i) maximize the total number of relief goods delivered from a local distribution center (LDC) to points of distribution (PODs), (ii) visit as many different PODs as possible, and (iii) limit late deliveries. Similar to the team orienteering problem (e.g., Karunakaran et al., 2019), we maximize the objective function within a limited time frame, adopting the practically relevant assumption that demand is too large to fulfill completely. To solve the problem, we propose two distinct deep reinforcement learning (RL) approaches. Whereas previous humanitarian logistics studies (e.g., Yu et al., 2021; Fan et al., 2022) applied tabular RL on small-scale deterministic problems and more general routing works focus on single vehicle problems (e.g., Mao and Shen, 2018), we apply deep reinforcement learning (RL) to handle large-scale stochastic problems and continuous state spaces. We compare two deep RL methods with three dynamic heuristics, as well as with solutions obtained by solving a static problem variant using a commercial solver. Theoretical instances provide insights into algorithmic performance. Additionally, real-world cases – constructed by local data collections in the field and using the aforementioned specifications of a real-world humanitarian cargo UAV – yield valuable insights for practice.

The remainder of this paper is organized as follows. Section 2 presents an overview of related works on stochastic dynamic vehicle routing problems, uncertainty in travel times, deployment of UAVs, and the application of reinforcement learning methods in humanitarian logistics. Section 3 describes the problem and in Section 4 we formulate the corresponding stochastic dynamic program. Section 5 introduces two reinforcement learning-based solution strategies. Section 6 presents the test instances, the real-world cases, the settings for the computational study, and the benchmark approaches. In Section 7, the results and analysis of the computational study are discussed. Finally, conclusions and directions for future work are given in Section 8.

2. Literature review

As our work categorizes under the broader umbrella of dynamic vehicle routing problems with stochastic travel times, we first provide an overview of works in this area (Section 2.1). Moreover, we analyze literature in the field of humanitarian logistics and distinguish three types of solution methods: exact methods, heuristics, and learning-based methods. The overview comprises literature discussing solution methods that deal with travel time uncertainty (Section 2.2), the application of UAVs (Section 2.3), and reinforcement learning methods in humanitarian operations (Section 2.4). Section 2.5 summarizes our contributions to literature.

2.1. Dynamic decision-making in routing problems with stochastic travel times

Compared to dynamic customers or stochastic demand, the consideration of stochastic travel times has generally received less attention in dynamic stochastic vehicle routing literature (Hildebrandt et al., 2022). In their review, Pillac et al. (2013) note that vehicle routing problems with uncertain travel times can be solved a priori or dynamically. Examples of a priori solutions methods are, e.g., Laporte et al. (1992), Van Woensel et al. (2008), Taş et al. (2014) and Bruni et al. (2021), whereas dynamic approached are discussed by, e.g., Potvin et al. (2006), Schilde et al. (2014), and Liu (2019). In this section, we review vehicle routing literature with stochastic travel times, covering solution approaches that focus on dynamic decision-making. In the dynamic variants, routing decisions are made based on the real-time state of the system, which typically updates every time a vehicle arrives at a location (Yu and Yang, 2019). Ritzinger et al. (2016) conclude in their review that savings in transportation costs are generally not extensive,

whereas the reliability of service can improve significantly when considering travel time uncertainty in the solution approach. The approaches are often benchmarked to myopic or static counterparts.

Early works on vehicle routing with stochastic travel times and dynamic decision-making typically use local search procedures (e.g., Potvin et al., 2006; Xiang et al., 2008; Pureza and Laporte, 2008; Lorini et al., 2011; Schilde et al., 2014) or genetic algorithms (e.g., Taniguchi and Shimamoto, 2004; Ghannadpour et al., 2014). Experimental evaluations in these papers show that dynamic decision-making generally results in decreased total costs or travel times and more robust outcomes with significantly fewer delays, especially in situations with large variance in travel times.

Later works consider heuristics that incorporate reinforcement learning methods and are often formulated as stochastic dynamic programs or Markov decision processes (MDPs). For example, Köster et al. (2015) propose a Markov decision process (MDP) for a dynamic traveling salesman problem with stochastic time-dependent travel times. Decisions are made at customer locations, from where the next destination for the vehicle is chosen. A 2-step nearest neighbor heuristic is used and situations with different communication levels on traffic information are compared. The numerical analysis shows that utilizing real-time information can decrease travel times up to 12%. Kim et al. (2016) present a dynamic VRP under traffic congestion. The problem is decomposed into customer clusters and each cluster is served by one vehicle. An MDP is formulated for each subproblem and solved by a rollout approach based on approximate dynamic programming and nearest neighbor as rollout heuristic. After visiting each customer, the next location is chosen. Compared to static routes, the dynamic routes achieve a 7% decrease in travel time. Köster et al. (2018) analyze the communication between traffic management systems and delivery companies. The authors define a dynamic VRP with stochastic changes in travel time matrices and formulate a route-based MDP. As the state space, the action space, and the number of possible transitions are very large, an approximative method is proposed to minimize the expected travel time while avoiding areas impacted by emission level changes. Based on an updated expected travel time matrix, an open TSP is solved for each vehicle and state. Results show that the dynamic policy reduces travel times up to 16% and time spent in polluted areas by more than 50%. Karunakaran et al. (2019) propose a dynamic approach for the stochastic team orienteering problem with time windows and stochastic service times to generate personalized tourist trip designs. The objective is to maximize total reward by visiting a subset of locations within a specified time frame. The authors design a multitasking genetic programming-based hyper-heuristic (GPHH) to create a set of policies for a wide range of scenarios. They show that the multitasking heuristic improves upon the general GPHH. Mao and Shen (2018) present a reinforcement learning framework to support the dynamic routing of a vehicle on a stochastic time-dependent network. They deploy Q-learning for a discrete action space and a fitted Q-iteration method with a tree-based approximation for a continuous action space. Compared to a dynamic programming approach, both learning-based methods show significant advantages in small instances and the fitted Q-iteration method is especially beneficial during peak hours due to flexible strategies. Liu (2019) present a dynamic VRP for UAV delivery with a mixed integer programming model and an optimization-driven progressive algorithm. A temporal progression framework that is solved in a rolling horizon fashion connects decisions over time periods and processes new information. A hierarchical objective ensures safety, minimizes lateness, and maximizes efficiency. Yu and Yang (2019) analyze a VRP with stochastic travel and service times. An approximate linear programming approach and a greedy heuristic policy are proposed and results show that the total makespan can be reduced significantly compared to static VRP solutions based on expected travel times. Liu et al. (2022) consider the flying sidekick traveling salesman problem with stochastic travel times on the road network. They use a deep Q-network and advantage actor-critic learning to overcome the curses of dimensionality. In a stochastic dynamic environment, the authors compare the results of reinforcement learning with static solutions of a mixed integer programming model and a dynamic nearest-neighbor heuristic. Results show that RL yields flexible policies that adapt to traffic conditions and save up to 29% on delivery time compared to the benchmark algorithms.

To summarize, in vehicle routing literature with stochastic travel times, both a priori and dynamic solution approaches have been considered and most solution methods presented are based on (meta)heuristics, especially approaches involving dynamic decision-making. Reinforcement learning presents a promising direction for these dynamic solution methods, but literature in this area mostly focuses on the routing of a single vehicle (e.g., Mao and Shen, 2018) or one truck and one drone (e.g., Liu et al., 2022). We contribute to the field by (i) analyzing a novel heterogeneous fleet problem variant with multiple trucks and drones and continuous travel time distributions, (ii) solving the problem dynamically with two different RL methods to overcome the curses of dimensionality in this complex problem setting, and (iii) comparing two RL methods and four benchmarks in a humanitarian setting in which high travel time uncertainties form an inherent challenge.

2.2. Travel time uncertainty in humanitarian logistics

The majority of humanitarian logistics studies that include uncertainty focus on demand uncertainty, with other forms of uncertainty relating to supply, affected areas and demand locations, product prices, and the transportation network (Liberatore et al., 2013). We focus on the latter, being a relatively understudied form of uncertainty. Transportation network uncertainty can be modeled by an uncertain number of vehicles that can simultaneously traverse arcs (uncertain road capacity), by probabilities of being able to traverse arcs (uncertain availability), or by probability distributions for the time traversing arcs (uncertain travel time) (Liberatore et al., 2013). Our study considers the latter. Regardless of the modeling choice, the associated time, costs, or distance from one location to another will be uncertain. We discuss approaches to deal with both small- and large-scale logistics problems including these uncertain travel times.

Most small-scale problems that embed transportation uncertainties in humanitarian logistics are solved with exact methods, including branch-and-bound (Ahmadi et al., 2015) or branch-and-cut algorithms (Elçi and Noyan, 2018). To mitigate uncertainties with exact solutions, one may deploy stochastic programs (e.g., Rennemo et al., 2014; Elçi and Noyan, 2018; Hu et al., 2019) or

robust optimization approaches (e.g., Fereiduni and Shahanaghi, 2017; Mohammadi et al., 2020; Mondal and Roy, 2021). Exact solution methods are generally used for instances of up to 50 nodes and a limited number of travel time scenarios. For example, Hu et al. (2019) apply a progressive hedging algorithm to solve a scenario tree with 16 scenarios and a multi-echelon network with 25 locations, and Mondal and Roy (2021) solve a robust pick-up and delivery routing problem with a network up to 44 nodes and 3 scenarios. Zhang et al. (2021) develop a distributionally robust optimization model for last-mile distribution with multiple centers. Their method outperforms a scenario-based model, both reducing costs and improving reliability in a case study with 13 areas. Nezhadroshan et al. (2021) address a humanitarian logistics network design problem with stochastic road availability by a scenario-based possibilistic–stochastic programming approach. The fuzzy chance constraint programming solves the problem with 14 candidate zones. Anuar et al. (2022) present a two-stage stochastic problem as well as several heuristics for a dynamic VRP with stochastic road capacities. The heuristics include a partial random construction approach, a sequential insertion heuristic, a cheapest insertion heuristic, and two dynamic look-ahead variants of the sequential insertion and cheapest insertion heuristic, respectively. The heuristics show performance comparable to the two-stage stochastic integer programming solution, while computation times are considerably lower.

For large-scale humanitarian logistics problems with travel time uncertainty, metaheuristics are commonly applied. Golabi et al. (2017) apply several heuristics to solve a facility location problem with accessibility uncertainties, with the aim to minimize aggregate travel times. Their genetic algorithm-based metaheuristic outperforms simulated annealing and a memetic algorithm in terms of computational efficiency. Mills et al. (2018) formulate an MDP model for dynamic ambulance routing after a disaster under uncertainty in travel times and congestion levels. Since solving the MDP exact is computationally intractable, several heuristics and policies are applied, e.g., a nearest facility heuristic, a myopic approach that only considers what is best for the current patient, and a one-step policy improvement approach. The authors show that a myopic approach increases the expected number of survivors compared to the nearest facility heuristic and that the policy improvement approach provides a slight additional improvement. Bruni et al. (2020a) address the selective minimum latency problem under stochastic travel times. The authors use a mean-risk approach to address the uncertainty of travel times, which can be adapted to different risk attitudes of the decision-maker. The aim is to minimize the sum of arrival times and risk, and a solution method based on VNS is developed to provide solutions quickly in the aftermath of a disaster. A bi-objective variant is studied by Bruni et al. (2020b), in which a minimum latency problem with profits is optimized with respect to profit and latency in the presence of stochastic travel times. The conditional Value-at-Risk is used to deal with the risk expressed by the standard deviation of the arrival times, and an iterative local search heuristic is able to find good quality solutions for small and medium-sized instances up to 76 nodes. Chang et al. (2022) apply simulation optimization techniques in combination with sample average approximation to obtain split-delivery inventory routing decisions for a network flow model with damages and congestion levels.

In short, literature on travel time uncertainty in humanitarian logistics considers various small-scale problems that are solved with exact approaches, whereas metaheuristics are applied for large-scale instances. For dynamic variants, most problems are solved with heuristics and look-ahead approaches, yielding reasonable results. Learning-based methods such as deep Q-learning or policy function approximation are not yet studied. We contribute to the existing body of literature by considering continuous travel time distributions, solving large-scale instances and real-world cases with two distinct reinforcement learning methods.

2.3. Cargo UAVs in humanitarian operations

Besides the use of UAVs for post-disaster assessment efforts (e.g., Oruc and Kara, 2018), there is a growing body of literature on the application of cargo UAVs in humanitarian response operations. These works consider facility location problems with UAV stations that address the limited range of UAVs (e.g., Golabi et al., 2017; Chauhan et al., 2019), allocation and routing problems (e.g., Nedjati et al., 2016; Rabta et al., 2018), and combinations of them (e.g., Chowdhury et al., 2017; Macias et al., 2020; Ghelichi et al., 2022). The majority of UAV-related problems are deterministic and solved either exact (Nedjati et al., 2016; Chowdhury et al., 2017; Rabta et al., 2018; Ghelichi et al., 2021; Gentili et al., 2022) or with metaheuristics, e.g., tabu search (Fikar et al., 2016), genetic algorithms (Shavarani, 2019), or large neighborhood search (Macias et al., 2020). Stochastic variants of UAV planning problems are typically solved with heuristic approaches. Golabi et al. (2017), as also mentioned in Section 2.2, consider a stochastic facility location problem in which UAVs perform deliveries to inaccessible locations. Ghelichi et al. (2022) define a discrete set of demand scenarios and formulate a chance-constrained program for locating UAV platforms and UAV scheduling in a disaster area. To overcome computational challenges, the authors develop a three-stage approximation heuristic, in which (i) a set of preferable platform combinations is identified, (ii) sub-problems for UAV scheduling are solved, and (iii) sample average approximation is employed to select the final set of platform locations.

To summarize, the application of UAVs in humanitarian missions is promising for relief distribution in hard-to-reach places and overcoming uncertainties in the transportation network after a disaster. For further reviews about the use of UAVs in humanitarian operations, we refer to Bravo and Leiras (2015) and Rejeb et al. (2021). These works discuss not only logistics problems, but also problems regarding communication networks, mapping, and monitoring. We contribute to the field of humanitarian cargo UAVs by analyzing fleet configurations with and without UAVs under stochastic travel times for trucks, providing insight into the added value of UAVs.

2.4. Reinforcement learning in humanitarian logistics

The literature review by Anuar et al. (2021) on modeling and optimization approaches in humanitarian vehicle routing problems states that learning-based approaches are scarce. Another recent review on reinforcement learning for logistics and supply chain management (Yan et al., 2022) also identified only a limited number of RL applications on disaster relief. As future directions in this area, Yan et al. (2022) propose to analyze the potential of multiple types of vehicles to provide more effective and agile services. Additionally, they propose to study the uncertainties that arise within a disaster area, such as those stemming from the typically damaged transportation infrastructure. RL may leverage new information being revealed during the response phase to learn to make better decisions. Current learning-based methods in the area of humanitarian operations include assessment activities, search-and-rescue operations, resource allocation problems, and emergency communication networks.

With respect to reinforcement learning in assessment and rescue operations, Su et al. (2011) propose an MDP-based path selection model for disaster response management. They apply Q-learning to find a safe and short path between locations for a single rescue team in a dynamic disaster environment. Nadi and Edrisi (2016) formulate an MDP model to coordinate the assessment of network conditions and relief demand in a disaster area in real-time. Q-learning is applied to decide how teams assess the area, such that the total assessment time is minimized. Nadi and Edrisi (2017) extend this work by integrating relief assessment and emergency response. The authors formulate an MDP for a multi-agent assessment and response system and solve it with tabular Q-learning, considering uncertain demand and road damage. An assessment team is routed to provide demand and route information, which is used to coordinate the response team to distribute goods and perform search-and-rescue operations. Yang et al. (2020) present a scheduling problem to deploy volunteers to rescue victims. A Q-learning algorithm carries out the matching based on the locations of victims. State–action spaces in these works are sufficiently small to be computationally tractable.

Within the area of humanitarian resource allocation problems, Yu et al. (2021) develop a Q-learning framework for a deterministic problem to allocate available goods to demand points in the disaster area, while minimizing transportation costs, deprivation costs, and terminal penalty costs. An exact dynamic programming solution is compared with a greedy heuristic and a Q-learning algorithm. By analyzing instances of up to 8 demand points and 12 periods, Q-learning outperforms the greedy heuristic by yielding optimal solutions, with computation times up to six times smaller than for the exact dynamic programming solution. Fan et al. (2022) formulate an MDP model for a deterministic resource allocation problem. The objective is to minimize transportation costs, deprivation costs throughout the periods, and the fairness penalty at the terminal state. They compare deep Q-learning with tabular Q-learning, a greedy heuristic, and an exact algorithm. As the size of the problem instances increases up to 10 points, deep Q-learning performs significantly better than tabular Q-learning in terms of calculation time and the objective value, whereas the exact algorithm fails to provide the optimal solution within a reasonable time limit.

An adaptable emergency communication network can be set up with UAVs in the event of a disaster in which communication infrastructure is disrupted. Managing these networks during humanitarian operations can be accomplished with RL methods. Klaine et al. (2018) apply Q-learning to find the best placement of multiple UAVs with wireless capabilities. The objective is to maximize the network radio coverage while attaining reliability against dynamic network conditions. Similarly, Zhou et al. (2021) apply Q-learning to optimize a disaster communication network. The authors suggest a deep reinforcement learning approach to approximate the Q-function for optimizing the delay and packet drop rate in the telecommunication network.

To summarize, reinforcement learning methods in humanitarian operations are currently mostly based on tabular Q-learning. The problem complexities addressed in existing literature are relatively low, as the Q-learning applications in assessment problems exhibit small-size state–action spaces and the resource allocation problems addressed are deterministic with a maximum of 10 locations. We fill this gap by considering an intractable state space with continuous probability distributions in travel times. To handle such state spaces, we deploy two deep RL methods, based on approximate value iteration and approximate policy iteration, respectively. Furthermore, we fill the literature gaps that were pointed out by Yan et al. (2022) by analyzing the deployment of multiple vehicle types and studying uncertainties in disaster areas resulting from damages to transportation networks.

2.5. Contributions to literature

Summarizing the literature review, our work features the following contributions to literature: (i) We address a last-mile distribution problem with a heterogeneous fleet with multiple trucks and UAVs and continuous stochastic travel times, and formulate this problem as a stochastic dynamic program. (ii) We perform a computational study to provide insight into the performance and robustness of deep reinforcement learning methods in comparison with dynamic heuristics and the static solutions of a commercial solver. (iii) Through this computational study, we analyze the impact of travel time uncertainty in humanitarian logistics operations, and (iv) analyze the added value of deploying UAVs to mitigate these uncertainties in the transportation network. Lastly, (v) we provide experimental insights considering stylized benchmark instances as well as two real-world cases, using realistic UAV specifications.

3. Problem description

In this section, we describe the multi-vehicle, multi-trip, split-delivery vehicle routing problem with soft time windows and stochastic travel times. The problem setting is a disaster area where last-mile distribution takes place via a local distribution center towards points of distribution. The local distribution center, hereafter referred to as the depot, is a location that consolidates relief goods from suppliers and serves as the departure and return point for the fleet of vehicles, in our case constituted of trucks and

UAVs. The trucks and UAVs perform distributions in parallel, similar to the parallel drone scheduling problem of Murray and Chu (2015). Aid workers from humanitarian organizations receive relief goods at the points of distribution (denoted as demand nodes) and distribute the goods among the beneficiaries nearby. We consider a VRP variant in which facility location and fleet size are fixed and information on the location and demand of nodes is known.

To formally introduce the problem, let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a complete graph with a set of nodes $\mathcal{N} = \{0, 1, \dots, |\mathcal{N}|\}$ and a set of arcs $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}\}$. Node 0 represents the depot in the network and nodes $\{1, 2, \dots, |\mathcal{N}|\}$ represent demand locations with an associated demand $d_n \in \mathbb{Z}^+, n \in \mathcal{N}$. The vehicle fleet $\mathcal{V}$ includes both trucks and UAVs $v \in \mathcal{V}$ with capacities Q_v. Let $T_{v,i,j}$ be the random variable for the travel times associated with vehicle v traveling from node i to node j over arc $(i, j) \in \mathcal{A}$.

We consider a single, interchangeable type of good that can be delivered to all nodes (e.g., food, shelter, or first aid kits). Decisions include the routing of vehicles and the number of relief goods to deliver at each node. Since the relief goods are interchangeable (i.e., one type of good for all locations), decisions on destination and delivery amount can be made dynamically after a loaded vehicle has departed from the depot. Vehicles are allowed to make multiple trips, perform split-deliveries, and are not required to visit all nodes or fulfill the complete demand at a node. In this way, it is allowed to skip nodes, fulfill demand with multiple vehicle visits, or partially fulfill the demand at (multiple) nodes. All vehicles return to the depot before the end of the day (denoted by $t^{\max}$), as reflected by a soft time window of $[0,\ t^{\max}]$ imposed on the depot. Soft time windows are also defined for each demand node, as aid workers from humanitarian organizations need to be present at the demand nodes to receive the goods and distribute them to the local beneficiaries in an orderly manner. Each node $n \in \mathcal{N}$ has an associated time window $[a_n, b_n]$ s.t. $0 \leq a_n < b_n \leq t^{\max}$ and vehicles are expected to visit the nodes within these time windows. When vehicles arrive early, they must wait until a_n, i.e., no penalty costs are incurred, but waiting will reduce the ability to visit locations within the limited time frame $[0,\ t^{\max}]$. When vehicles arrive after b_n, they incur a superlinear penalty that weighs lateness by the number of goods delivered and whether the vehicle is the first to arrive at this location. Lateness is weighed by the number of goods delivered because if more goods arrive late, the deprivation effects (e.g., suffering due to lack of resources) are worse than with fewer goods. The same holds for the delivery of the first vehicle, where the effects of a late arrival of the first delivery are worse compared to succeeding deliveries: if a location already received goods in the first delivery (e.g., to serve the people there that are most in need), subsequent deliveries that arrive late have a comparatively reduced negative impact.

The weighted objective is to maximize (i) the total number of relief goods delivered and (ii) the total number of nodes visited at least once before $t^{\max}$, while (iii) minimizing penalties for late arrivals at nodes. To weigh these partial objectives, we assign weight $\xi_1 \in \mathbb{R}^+$ to the number of goods delivered, weight $\xi_2 \in \mathbb{R}^+$ to the number of nodes visited, and weight $\xi_3 \in \mathbb{R}^-$ to the lateness penalty incurred. Recall that typical demand is too large to fulfill completely, such that balancing the partial objectives is inevitable. Weights can be adjusted according to the decision-maker's preference. In particular, increasing the weight on the number of nodes visited promotes a higher coverage of nodes (egalitarian perspective), potentially at the cost of lowering system-wide utility regarding the total number of relief goods delivered (utilitarian perspective).

We distinguish between two types of vehicles, namely trucks and UAVs. Trucks have a larger payload Q_v, travel slower, and are faced with uncertain travel times, since the transportation network is disrupted by the disaster and may be affected during the operation by, e.g., aftershocks or road repairs. As parts of the disaster area might be damaged more than others, e.g., a part close to the epicenter of an earthquake or the path of a hurricane, higher levels of travel time uncertainty are experienced in severely affected parts. We assume that the continuous distribution functions $T_{v,i,j}$ are known by the planner. In contrast to trucks, UAVs have relatively small payloads Q_v, travel quickly, and are assumed to fly in Euclidean distance circumventing the infrastructural uncertainties. Hence, we assume deterministic travel times for UAVs. Service times at the depot and at demand nodes are ignored for both vehicles.

4. Stochastic dynamic program

We formulate the problem as a finite horizon stochastic dynamic program and define the state variables, decision variables, exogenous information, transition function, and objective function. This enables us to develop policies that anticipate uncertainties in travel time and return dynamic decisions with evolving information. We follow the sequential decision-making framework proposed by Powell (2019). We choose this framework for the following four reasons. (i) The transition function in the framework of Powell (2019) clearly describes system dynamics and explicitly enables sampling state transitions, which deviates from transition probabilities defined in the classical MDP framework as proposed by Puterman (1990). (ii) We study a system with continuous travel times, resulting in an intractable outcome- and state space, which does not allow an explicit calculation of transition probabilities. (iii) In our state formulation, we include variables that track the last obtained delivery- and location reward for each vehicle. Although the inclusion of historical data in the state is uncommon in MDPs, the sequential decision-making framework allows for this adaptation, which ensures that the transition and reward solely depend on the current state and action. (iv) Being geared towards RL methods, the framework provides a suitable starting point for the deep reinforcement learning methods we apply to solve the sequential decision problem at hand.

We first define the decision epochs of our dynamic program as the moments at which a vehicle becomes available. Vehicles become available at time 0 at the depot, upon arrival at their destination while the other vehicles are still in transit, or, in case of early arrival, at the start of the node's time window. When a vehicle becomes available, its next location to visit as well as the delivery amount will be determined. The exact time between adjacent decision epochs (i.e., vehicle availabilities) is not known beforehand and decisions are thus not made in equidistant time intervals. Instead, the time between decision epochs is subject to the stochastic process of travel times, locations' time windows, and the policy being used, as traveling to nearby locations (e.g., by

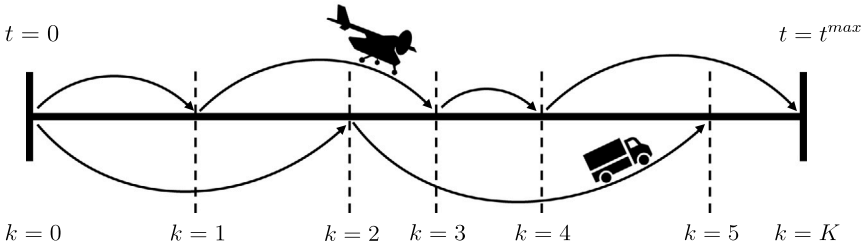


Fig. 1. Single truck, single UAV illustration of decision epochs over time. The horizon of the stochastic dynamic program is limited by a time frame $[0,\ t^{\max}]$. The arrows reflect the travel times and potential waiting times between nodes for each vehicle and their lengths follow from the routing decisions, time windows, and the realization of the travel times, jointly determining the number of decision epochs within the time frame.

Table 1
Description of state variables.

State variables		Description
$\hat{v}_k$	$\in \mathcal{V}$	Current available vehicle at decision epoch k, $k \in \mathcal{K}$
$q_{k,v}$	$\in \mathbb{Z}_{\geq 0}$	Remaining payload of vehicle v at decision epoch k, $k \in \mathcal{K}$, $v \in \mathcal{V}$
$i_{k,v}$	$\in \mathcal{N}$	Origin of vehicle v at decision epoch k, $k \in \mathcal{K}$, $v \in \mathcal{V}$
$j_{k,v}$	$\in \mathcal{N}$	Destination of vehicle v at decision epoch k, $k \in \mathcal{K}$, $v \in \mathcal{V}$
$t^{\text{dep}}_{k,v}$	$\in \mathbb{R}_{\geq 0}$	Time since departure from origin of vehicle v at decision epoch k, $k \in \mathcal{K}$, $v \in \mathcal{V}$
$c_{k,v}$	$\in \mathbb{R}_{\geq 0}$	Previously gained reward by the amount delivered and first visit of vehicle v at decision point k, $k \in \mathcal{K}$, $v \in \mathcal{V}$
$d_{k,n}$	$\in \mathbb{Z}_{\geq 0}$	Unfulfilled demand at node n at decision epoch k, $k \in \mathcal{K}$, $n \in N$
$a_{k,n}$	$\in \mathbb{R}$	Time until start of time window of node n at decision epoch k, $k \in \mathcal{K}$, $n \in \mathcal{N}$
$b_{k,n}$	$\in \mathbb{R}$	Time until end of time window of node n at decision epoch k, $k \in \mathcal{K}$, $n \in \mathcal{N}$
$f_{k,n}$	$\in \{0,1\}$	Equals 1 if node n at decision epoch k is not yet visited and 0 otherwise, $k \in \mathcal{K}$, $n \in \mathcal{N}$
t_k	$\in \mathbb{R}_{\geq 0}$	Current time at decision epoch k, $k \in \mathcal{K}$

using a nearest neighbor heuristic) will result in relatively shorter time periods between decision epochs. As we have a finite number of goods to deliver $\sum_{n\in\mathcal{N}}(d_n)$ and a limited time frame $[0,\ t^{\max}]$ is considered, the system must terminate and the set of decision epochs is finite. Under the assumption that goods are delivered one by one, the theoretical upper bound of the number of decision epochs is the sum of the number of goods to be delivered. We define the decision epochs $k \in \{0, 1, \ldots, K\}$, where K is a random variable dependent on the policy being used and the travel time realizations within $[0,\ t^{\max}]$, similar to, e.g., Ulmer et al. (2020), and the stochastic shortest path problem described by Bertsekas (2019). See Fig. 1 for an illustration of the decision epoch structure. Following Powell (2019), we next define the state variables, decision variables, exogenous information, transition function, and objective function.

State variables. The state S_k of a system contains all necessary information to model the system from epoch k onwards (Powell, 2019), capturing all information needed to make feasible decisions, calculate rewards, and compute transitions. The state S_k includes physical resource information R_k and information I_k, as subsequently detailed.

The resource vector R_k captures the vehicle availabilities $\hat{v}_k$ and the remaining vehicle payloads $q_{k,v}$. To compute sequential arrivals correctly, the state tracks the origins $i_{k,v}$ and destinations $j_{k,v}$ of each vehicle, as well as the times elapsed since departure $t^{\text{dep}}_{k,v}$. To facilitate computation of the lateness penalty, variable $c_{k,v}$ stores the potential costs in case of lateness, based on the number of relief goods to deliver and whether it is the first one arriving at the destination node. By including $c_{k,v}$ in the state, the transition function and reward depend only on the current state and action (Powell, 2019). In summary, vehicle information is captured by: $R_k = \left(\hat{v}_k, q_{k,v}, i_{k,v}, j_{k,v}, t^{\text{dep}}_{k,v}, c_{k,v}\right) v \in \mathcal{V}$.

The information vector I_k captures the remaining demand $d_{k,n}$ of node n at epoch k. For each node, variables $a_{k,n}$ and $b_{k,n}$ keep track of the time remaining until the start and end of the time window, respectively. Furthermore, $f_{k,n}$ is a binary variable that stores whether location n has been visited, being initialized at 1 and set to 0 upon the first visit. Since time intervals between decision epochs are not equidistant, vector I_k captures the current time t_k corresponding to decision epoch k. Combining these variables, the information vector reads as $I_k = \left(d_{k,n}, a_{k,n}, b_{k,n}, f_{k,n}, t_k\right) n \in \mathcal{N}$. The state variables are summarized in Table 1 and the state is defined as:

$$S_k = (R_k, I_k) \tag{1}$$

Decision variables and reward function. At every decision epoch k with current time t_k, one vehicle $\hat{v}_k$ is marked as available (via randomization if needed) at its destination, determined by the exogenous information and transition function described later. The decision $x_k = \left(x^{\text{dest}}_k, x^{\text{amount}}_k\right)$ determines (i) the next destination x^{dest}_k for the available vehicle and (ii) the number of goods x^{amount}_k to deliver at the next destination. The decision space for the destination is $\mathcal{N}$ and the decision space for the number of goods has a lower bound of zero and an upper bound limited by the remaining vehicle payload $q_{k,\hat{v}_k}$ for vehicle $\hat{v}_k$ and the remaining the demand at the next destination d_{k,x^{dest}_k}. The decisions are made under the conditions that:

- Decisions are only made for the vehicle that is marked available, not for other vehicles.
- Remaining demand at the next destination must be larger than zero, i.e., $d_{k,x_k^{\text{dest}}} > 0$, except when the next destination is the depot.
- If the next destination is the depot, $x_k^{\text{dest}} = 0$, the amount to deliver $x_k^{\text{amount}} = 0$. Furthermore, the vehicle will be reloaded up to full capacity, so the remaining payload will be updated to $q_{k,\hat{v}_k} = Q_{\hat{v}_k}$.
- When the remaining payload of the vehicle is zero, i.e., $q_{k,\hat{v}_k} = 0$, the next destination is always the depot.
- If the next destination is not the depot:
  - the number of relief goods to deliver can be anywhere between 1 and $\min\left(q_{k,\hat{v}_k}, d_{k,x_k^{\text{dest}}}\right)$;
  - the expected travel time to the next destination and from the next destination to the depot must be smaller than the time remaining until $t^{\max}$, i.e., $t_{\hat{v}_k,i_{k,\hat{v}_k},x_k^{\text{dest}}} + t_{\hat{v}_k,x_k^{\text{dest}},0} < t^{\max} - t_k$, where $t_{\hat{v}_k,i_{k,\hat{v}_k},x_k^{\text{dest}}}$ is the expected travel time of vehicle $\hat{v}_k$ from current origin $i_{k,\hat{v}_k}$ to the next destination x_k^{dest} and $t_{\hat{v}_k,x_k^{\text{dest}},0}$ is the expected travel time of vehicle $\hat{v}_k$ from the next destination x_k^{dest} to the depot 0. This constraint enforces that all vehicles are expected to be returned to the depot by the end of the day.

The reward (or contribution) C_k of decision x_k in state S_k follows the weighted objective of the problem. The reward function is based on (i) the number of goods that is delivered, (ii) whether a location is visited for the first time, and (iii) the lateness penalty, where $b_{k,j_{k,\hat{v}_k}}$ adopts a negative value when vehicle $\hat{v}_k$ arrives late at its destination. Note that the lateness penalty is calculated upon arrival of the vehicle at its current location, whereas the rewards for the delivery amount and first arrival are assigned upon making the decision (i.e., the reward for the next destination). Hence, the reward at decision epoch k is defined as:

$$C_k(S_k, x_k) = \xi_1 x_k^{\text{amount}} + \xi_2 f_{k,x_k^{\text{dest}}} + \xi_3 \min(0, b_{k,j_{k,\hat{v}_k}}) c_{k,\hat{v}_k} \tag{2}$$

Exogenous information. Once a decision is made, the vehicle heads toward its new destination. The travel times of the vehicles are subject to stochasticity. The realizations of these travel times are conditional to the fact that the arrivals need to be in the future at epoch k, i.e., $\mathbb{F}\left(T_{v,i,j} \mid T_{v,i,j} \geqslant t_{k,v}^{\text{dep}}\right)$, with $\mathbb{F}$ being the inverse probability function for the travel time conditional to the time since departure. It is uncertain which vehicle will become available first and in which time interval. In case a vehicle arrives early, it only becomes available at the start of the time window of the respective node. The exogenous information W_{k+1} that becomes known at decision epoch $k+1$ is the realization of the first vehicle that becomes available among all vehicles. The exogenous information is characterized by the specific vehicle that becomes available W_{k+1}^{v} and the time period between this availability time and the previous one, denoted by W_{k+1}^{τ}.

The individual time period for each vehicle $v \in \mathcal{V}$ until it becomes available is defined by $\max\left\{T_{v,i_{k,v},j_{k,v}} - t_{k,v}^{\text{dep}},\ a_{k,j_{k,v}}\right\}$, with the first term describing the realization of the travel time minus the time since departure and the second term describing the time until the start of the vehicle's destination's time window $a_{k,j_{k,v}}$. The travel time realization of vehicle v depends on its origin $i_{k,v}$ and destination $j_{k,v}$. Similarly, the time until the start of the time window depends on the specific destination node $j_{k,v}$ of vehicle v. If a realized travel time results in an early arrival, the time until availability is equal to the time until the start of the time window. If the vehicle does not arrive early, the time until availability is equal to the realized travel time. Taking the minimum time until availability over all vehicles provides the first time until availability W_{k+1}^{τ} and the argument of this minimum gives the associated vehicle W_{k+1}^{v}. Hence, the exogenous information W_{k+1} is defined as:

$$\begin{aligned} &W_{k+1} = (W_{k+1}^{v},\ W_{k+1}^{\tau}) = \\ &\left(\underset{v\in\mathcal{V}}{\operatorname{argmin}}\left(\max\{T_{v,i_{k,v},j_{k,v}} - t_{k,v}^{\text{dep}},\ a_{k,j_{k,v}}\}\right),\ \min_{v\in\mathcal{V}}\left(\max\{T_{v,i_{k,v},j_{k,v}} - t_{k,v}^{\text{dep}},\ a_{k,j_{k,v}}\}\right)\right) \end{aligned} \tag{3}$$

Transition function. Once a decision is made, the reward is calculated, and the exogenous information is known, the system can be updated from S_k to S_{k+1} according to transition function S^M:

$$S_{k+1} = S^M(S_k, x_k, W_{k+1}) \tag{4}$$

As discussed in Powell (2011), we can split the state transition into a deterministic transition from the state S_k to the so-called post-decision state S_k^x and a stochastic transition from the post-decision state S_k^x to the next state S_{k+1}. The post-decision state S_k^x incorporates the new decision x_k via the function $S_k^x = S^{M,x}(S_k, x_k)$ and the next state S_{k+1} incorporates the random information W_{k+1} via the function $S_{k+1} = S^{M,W}(S_k^x, W_{k+1})$.

We first describe the transition to the post-decision state. For the sake of readability, we only show the variables that change as a result of decisions x_k^{amount} and x_k^{dest} of the available vehicle $\hat{v}_k$. The transition $S^{M,x}$ to the post-decision state is performed such that:

$$q_{k,\hat{v}_k}^{x} = \begin{cases} Q_{\hat{v}_k}, & \text{if } x_k^{\text{dest}} = 0, \\ q_{k,\hat{v}_k} - x_k^{\text{amount}}, & \text{otherwise.} \end{cases} \tag{5}$$

$$j_{k,\hat{v}_k}^{x} = x_k^{\text{dest}} \tag{6}$$

$$t_{k,\hat{v}_k}^{\text{dep},\ x} = 0 \tag{7}$$

$$c^x_{k,\hat{v}_k} = \xi_1 x_k^{\text{amount}} + \xi_2 f_{k,x_k^{\text{dest}}} \tag{8}$$

$$d^x_{k,x_k^{\text{dest}}} = d_{k,x_k^{\text{dest}}} - x_k^{\text{amount}} \tag{9}$$

$$f^x_{k,x_k^{\text{dest}}} = 0 \tag{10}$$

Eq. (5) updates the remaining payload of vehicle $\hat{v}_k$ by subtracting the amount to deliver to the next destination from the current payload. If the next destination is the depot, the payload will be reloaded to full capacity. Eq. (6) updates the new destination of vehicle $\hat{v}_k$ to x_k^{dest}. Eq. (7) sets the time since the departure of vehicle $\hat{v}_k$ to 0. Eq. (8) stores the direct reward stemming from the delivery amount and the first vehicle arrival. The remaining demand of the destination is updated in Eq. (9) and the visit is updated in Eq. (10).

The transition to the new state S_k involves updates to all state variables with regard to the newly available vehicle W^v_{k+1} and its interarrival time W^τ_{k+1}. Again, we only show variables that change during the transition. After the new information W_{k+1} is revealed, the transition function $S^{M,W}$ from the post-decision state S^x_k to the new state S_{k+1} reads as follows:

$$\hat{v}_{k+1} = W^v_{k+1} \tag{11}$$

$$i_{k+1,W^v_{k+1}} = j_{k+1,W^v_{k+1}} \tag{12}$$

$$t^{\text{dep}}_{k+1,v} = t^{\text{dep},\ x}_{k,v} + W^\tau_{k+1} \qquad \forall v \in \mathcal{V} \tag{13}$$

$$a_{k+1,n} = a^x_{k,n} - W^\tau_{k+1} \qquad \forall n \in \mathcal{N} \tag{14}$$

$$b_{k+1,n} = b^x_{k,n} - W^\tau_{k+1} \qquad \forall n \in \mathcal{N} \tag{15}$$

$$t_{k+1} = t^x_k + W^\tau_{k+1} \tag{16}$$

Eq. (11) updates the available vehicle and since this vehicle arrived at its destination, Eq. (12) sets the origin of the available vehicle to its destination. Eqs. (13) update the times since departure for all vehicles by adding W^τ_{k+1}. With time period W^τ_{k+1}, the times until the start and end of the time windows are updated in Eqs. (14)–(15), and the same is done for the time of the next decision epoch t_{k+1} in Eq. (16).

Objective function. We aim to find a policy $\pi \in \Pi$ that maximizes the expected rewards $C_k(S_k, x_k)$ with decisions $X^\pi_k(S_k)$ summed over all decision epochs given the initial state S_0, as reflected in the objective function:

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{K} C_k\left(S_k, X^\pi_k(S_k)\right) \mid S_0\right] \tag{17}$$

Here, the state trajectory follows transition functions $S_{k+1} = S^M(S_k, x_k, W_{k+1}) = S^{M,W}\left(S^{M,x}(S_k, x_k), W_{k+1}\right)$ and initial state S_0 captures all the parameters and settings known at the beginning of the day. The stochastic dynamic program terminates when the current time reaches the time limit, i.e., $t_k \geqslant t^{\max}$.

As we do not know the travel times for the full horizon, rewards cannot be computed deterministically. Instead, we maximize the direct reward and expected future rewards, considering all possible future outcomes. Let Ω_k be the set of all possible vehicle arrivals at k and let $\omega_k \in \Omega_k$ be a particular realization of vehicle arrivals. The exogenous information W_{k+1} may be any realization ω_{k+1}. Let $\mathbb{P}(W_{k+1} = \omega_{k+1})$ describe the conditional probabilities of vehicle arrivals. The probability $\mathbb{P}(W_{k+1} = \omega_{k+1})$ depends on the travel time distributions, the state, and the decision. Furthermore, the probability is conditional to that each arrival lies in the future and that the time windows for all other vehicles open later, as discussed in the paragraph about the exogenous information and as shown in Eq. (3). The probability of vehicle $\hat{v}_k \in \mathcal{V}$ being the first available vehicle after time W^τ_{k+1} – considering travel time realizations of all vehicles – can be calculated as the product of the probabilities that each other vehicle $v \in \mathcal{V} \setminus \{\hat{v}_k\}$ arrives later when traveling from origin $i_{k,v}$ to destination $j_{k,v}$:

$$\mathbb{P}(W_{k+1} = \omega_{k+1}) = \mathbb{P}^\omega_{\hat{v}_k \in \mathcal{V}} = \prod_{v \in \mathcal{V} \setminus \{\hat{v}_k\}} \mathbb{P}\left(T_{\hat{v}_k, i_{k,\hat{v}_k}, j_{k,\hat{v}_k}} - t^{\text{dep}}_{k,\hat{v}_k} < T_{v,i_{k,v},j_{k,v}} - t^{\text{dep}}_{k,v} \mid T_{\hat{v}_k, i_{k,\hat{v}_k}, j_{k,\hat{v}_k}} > t^{\text{dep}}_{k,\hat{v}_k}, T_{v,i_{k,v},j_{k,v}} > t^{\text{dep}}_{k,v}, a_{k,j_{k,\hat{v}_k}} < a_{k,j_{k,v}}\right) \tag{18}$$

Theoretically, we obtain the optimal policy by solving the system of Bellman equations (Bellman, 1957):

$$V_k(S_k) = \max_{\pi \in \Pi}\left[C_k(S_k, X^\pi_k(S_k)) + \sum_{\omega_{k+1} \in \Omega_{k+1}} \mathbb{P}(W_{k+1} = \omega_{k+1}) V_{k+1}(S_{k+1} \mid S_k, X^\pi_k(S_k), \omega_{k+1})\right] \forall S_k \tag{19}$$

However, due to the state and outcome space, determining the second term of the equations with Ω_k and transition functions V is intractable. To address this issue, we use deep reinforcement learning methods to approximately solve the sequential decision problem.

5. Reinforcement learning methods

We apply two reinforcement learning methods rooted in two fundamental learning-based classes, namely value function approximation (VFA) and policy function approximation (PFA) (Powell, 2022). The VFA class contains strategies that seek to approximate the value functions found in Bellman's optimality equations. The PFA class contains approximation strategies that map states directly to decisions by a lookup table or (non-)parametric function. For both classes, we apply parameterized policies in which an algorithm learns parameters enabling to identify both direct and indirect effects of decisions (Powell, 2022). To handle continuous states, we employ neural networks as parameterized function approximators in both solution approaches. Neural networks offer an adaptable class of nonlinear functions that can be updated recursively. The details of the applied methods are discussed in Sections 5.1 and 5.2, and contrasted in Section 5.3. Section 5.4 introduces input features and an action space reduction to ease training.

5.1. Value function approximation

As stated in Section 4, it is impossible to enumerate all possible states and their corresponding values. Instead, we estimate a value function based on the post-decision state with $\overline{V}(S_k^x) = \overline{V}(S^{M,x}(S_k, x_k))$. As the objective function is nonlinear, we approximate the value function with a neural network (denoted by $\mathbb{G}$), which performs nonlinear transformations on the input (the post-decision state and derived features, see Section 5.4) to approximate the post-decision state value. From Eq. (19), we replace the expected future value with the approximate value function $\overline{V}(S_k^x) = \mathbb{G}(S_k^x)$. The VFA policy π^{VFA} chooses the action that maximizes the sum of direct reward and corresponding approximated future value:

$$\begin{aligned} x_k^{\text{dest}} = \pi^{VFA}(S_k) &= \underset{x_k \in \mathcal{X}}{\operatorname{argmax}} \left[C_k(S^{M,x}(S_k, x_k)) + \overline{V}(S^{M,x}(S_k, x_k)) \right] \\ &= \underset{x_k \in \mathcal{X}}{\operatorname{argmax}} \left[C_k(S^{M,x}(S_k, x_k)) + \mathbb{G}(S^{M,x}(S_k, x_k)) \right] \end{aligned} \tag{20}$$

where $C_k(S^{M,x}(S_k, x_k))$ is the direct reward observed with the transition to the post-decision state and $\mathbb{G}(S^{M,x}(S_k, x_k))$ is the neural network approximating the future value of the post-decision state.

To learn function $\overline{V}$, we adapt the approximate value iteration (AVI) algorithm described in Powell (2011). For stabler and more efficient learning, we update based on sample batches (originally proposed by Lin (1992) as experience replay) rather than single samples. We rely on Monte Carlo simulation to collect sample batches (containing post-decision states and their sampled future values). Additionally, we replicate state transitions for each sample, improving the reliability of downstream reward estimates. A new policy $\hat{\pi}^{VFA}$ is obtained by updating the neural network $\mathbb{G}$ with the new sample batch. To train the neural network $\mathbb{G}$ to map the post-decision state to post-decision state values, we minimize the loss via the mean squared error. We perform multiple sample- and training iterations to gradually improve the policy. Exploration and exploitation during training is balanced via an ϵ-greedy approach, selecting random decisions with probability ϵ and reward-maximizing decisions with probability $1 - \epsilon$. During updating in the replications, we only make reward-maximizing decisions, generally leading to faster convergence. Algorithm 1 outlines the pseudo-code of the approximate value iteration algorithm.

Algorithm 1: Approximate value iteration (AVI)

Initialize π_0 with $\mathbb{G}_0$, and set ϵ_0
for $\alpha = 0$ to maximum iterations **do**
  Generate sample paths $\omega \in \Omega$, and initialize state S_0
  for $\beta = 0$ to maximum samples **do**
    Make decision $\tilde{x}$ using policy π_α and ϵ-greedy approach
    Transition to post-decision state $S_\beta^{\tilde{x}} = S^{M,x}(S_\beta, \tilde{x})$
    for $\rho = 0$ to maximum replications P **do**
      Transition to new state $S_\rho = S^{M,W}(S_\rho^{\tilde{x}}, \omega_\rho)$
      Approximate post-decision state value $v_\rho = \max_{x \in \mathcal{X}} \left[C_\rho(S_\rho, x_\rho) + \mathbb{G}(S_\rho^x) \right]$
    end
    Save sample with S_β and $v_\beta = \frac{1}{\mathrm{P}} \sum_\rho v_\rho$
    Transition to new state $S_{\beta+1} = S^{M,W}(S_\beta^{\tilde{x}}, \omega_0)$
    if $t_\beta \geqslant t^{max}$ **then**
      Generate new sample paths $\omega \in \Omega$ and initialize state $S_{\beta+1}$
    end
  end
  Update approximation $\mathbb{G}_{\alpha+1} \leftarrow \mathbb{G}_\alpha$ with stored samples
  Update $\epsilon_{\alpha+1} \leftarrow \max(\epsilon_\alpha - \epsilon\text{-decay}, 0)$
end

5.2. Policy function approximation

Instead of approximating the direct and future values of post-decision states (i.e., state and action combinations) as with VFA, the PFA policy π^{PFA} aims to determine the best action directly from the state S_k. Hence, π^{PFA} employs a policy network $\mathbb{G}'$ that maps state S_k to an action, without performing a transition to the post-decision state S_k^x:

$$x_k^{\text{dest}} = \pi^{PFA}(S_k) = \underset{x \in \mathcal{X}}{\text{argmax}} \left[\left(\mathbb{G}'(S_k) \right)_x \right] \tag{21}$$

Whereas the VFA relies on regression, the PFA employs a classification approach. The size of the output layer of the neural network $\mathbb{G}'$ is equal to the number of decisions (i.e., the number of nodes $|\mathcal{N}|$). The categorical decision chosen with PFA corresponds to the output node with the highest probability of being the best decision $x_k \in \mathcal{X}$. A cross-entropy function describes the loss to be minimized. As the neural network has a fixed number of output nodes, infeasible decisions are 'masked' such that they cannot be selected (e.g., demand nodes without remaining demand). For more details on the procedures, we refer to Silver et al. (2018).

For the learning process of the PFA, we adopt an approximate policy iteration (API) algorithm that shows conceptual similarities to Silver et al. (2018), see also Van Jaarsveld (2020). With policy iteration, we would enumerate over all states and as this is not possible, we instead collect a large batch of samples. These samples are collected under the current policy for all feasible decisions of sampled states to find the decisions that corresponded to the highest reward over the sample trajectory. Similar to the AVI algorithm, we only update the PFA after collecting a large batch of samples to enhance the stability and efficiency of training and we perform multiple replications to improve the reliability of the samples. We perform multiple iterations to improve the policy. Algorithm 2 outlines the pseudo-code of the approximate policy iteration algorithm.

Algorithm 2: Approximate policy iteration (API)

Initialize π_0 with $\mathbb{G}'_0$ and ϵ_0
for $\alpha = 0$ to maximum iterations **do**
  Choose sample paths $\omega \in \Omega$, and initialize state S_0
  for $\beta = 0$ to maximum samples **do**
    for $x \in \mathcal{X}$ **do**
      for $\rho = 0$ to maximum replications P **do**
        Transition to next state S_ρ with decision x
        Find downstream rewards $\hat{\mathcal{Z}}_{x,\rho}$ with policy π_α and sample path ω_ρ
      end
    end
    Store state S_β with best decision $\hat{x} = \underset{x \in \mathcal{X}}{\text{argmax}} \left(\frac{1}{\text{P}} \sum_\rho \hat{\mathcal{Z}}_{x,\rho} \right)$
    Make decision $\tilde{x}$ using policy π_α and ϵ-greedy approach
    Transition to post-decision state and transition to new state $S_{\beta+1}$
    if $t_\beta \geqslant t^{max}$ **then**
      Choose new sample paths $\omega \in \Omega$ and re-initialize state $S_{\beta+1}$
    end
  end
  Update approximation $\mathbb{G}'_{\alpha+1}$ using collected samples
  Update $\epsilon_{\alpha+1} \leftarrow \max(\epsilon_\alpha - \epsilon\text{-decay}, \text{minimum epsilon})$
end

5.3. Comparison of VFA and PFA

The VFA class of RL bases its decisions on the approximated value of states whereas the PFA class directly maps states to decisions. To make a decision in the current state, VFA iterates over all actions $\mathcal{X}$ to calculate the direct reward and expected future value and selects the one that maximizes the sum of direct reward and expected future value. PFA does not require calculating this for every action, but directly outputs which action to take in the current state. Hence, PFA is computationally much more efficient in execution.

Besides the difference in decision-making between VFA and PFA, there are also clear differences in the sampling methods employed by AVI in Algorithm 1 and API in Algorithm 2. To obtain a post-decision state sample, AVI performs one deterministic transition to the post-decision state $S_\beta^{\tilde{x}} = S^{M,x}(S_\beta, \tilde{x})$ and P replications of stochastic state transitions $S_\rho = S^{M,W}(S_\rho^{\tilde{x}}, \omega_\rho)$ with $|\mathcal{X}|$ state approximations to determine $v_\rho = \max_{x \in \mathcal{X}} \left[C_\rho(S_\rho, x_\rho) + \mathbb{G}(S_\rho^x) \right]$. This is a computationally efficient but potentially inaccurate way of sampling (Ulmer and Thomas, 2020), such that AVI may require many iterations. In contrast, API obtains a state sample by performing for all feasible decisions $x \in \mathcal{X}$ and for P replications a downstream reward calculation by simulating the system of transition functions to the final epoch K. This will require $\overline{K}$ downstream calculations at the start of the horizon, in which $\overline{K}$ is the expected number of epochs in one horizon, and 1 downstream calculation near the end of the horizon. Hence, whereas AVI performs one transition to the post-decision state, P transitions to a next state, and $|\mathcal{X}|$P state approximations, API performs between $|\mathcal{X}|$P and

$|\mathcal{X}|\overline{PK}$ full state transitions to obtain a sample. Although at higher computational costs, API likely obtains better sample estimates due to the full horizon rollouts. Consequently, PFA algorithms often converge in much fewer iterations than VFA algorithms (Ulmer and Thomas, 2020). Furthermore, according to Ulmer and Thomas (2020), PFA algorithms may experience difficulties with large action spaces, as each action has a unique output node in the neural network and rollouts are required for each possible action. Powell and Meisel (2016) argue that the VFA class works well when the future costs of a state can be approximated easily, and the PFA class works well when the relationship between the state and action is clear. However, the authors note that it is difficult to assess these characteristics upfront. Due to the approximative nature of both approaches, with finite samples and iterations, and limited neural network size, there is no guarantee of convergence. Nonetheless, due to the structural properties of AVI, we expect to find proper results after many iterations, but within reasonable time. As the action space in our problem is fairly limited ($\mathcal{X} = \mathcal{N}$), we expect to find appropriate results with the API algorithm as well, requiring fewer but computationally heavier iterations. However, as API extensively evaluates each decision, it does not scale well to larger instances.

Our problem formulation originates from a practical perspective, which led to some undesirable properties from a theoretical perspective, such as the non-linear objective function, the need for a complex function approximation, the intractable state space, and the random number of decision epochs. The proposed VFA and PFA methods are able to handle these aspects, but they do not come with performance guarantees or the ability to provide error bounds of the solutions. While such bounds for reinforcement learning methods are discussed in literature (e.g., Bertsekas and Tsitsiklis, 1996; Munos, 2005), they require strong assumptions whose validity is hard to guarantee in practice (Bertsekas, 2019). Error bounds often only exist under restrictive assumptions, such as linear function approximations, linearly independent features, the Bellman completeness condition, known transition probabilities, fixed-step sizes, finite state spaces, and discounted infinite horizons (Munos, 2005; Beck and Srikant, 2012; Wang and Giannakis, 2020). Given the complexities of our problem setting and the applied methods, we will rely on an empirical comparison with benchmark methods in our numerical analysis, these benchmark methods will be discussed in Section 6.2.

5.4. Feature design and action space reduction

To guide the methods, we derive three feature variables from the state, which together with the state vector are provided as input to the neural network. These features are computed at each decision epoch $k \in \mathcal{K}$ for each vehicle $v \in \mathcal{V}$. The designed features read as follows: (i) the expected time until arrival of vehicle ($\tilde{t}^{\text{arr}}_{k,v}$), (ii) the expected earliness ($\tilde{e}_{k,v}$), and (iii) the expected lateness ($\tilde{l}_{k,v}$). The expected time until arrival can be computed by the probability function of the travel time, truncated by the time since departure. The other two features are derived from the expected time until arrival and the time windows of nodes. Analyses including additional features, such as the probabilities of arriving early, within the time window, or late, and the uncertainty levels of arcs did not improve results and are therefore not presented in this study.

For the sake of computational efficiency, we reduce the action space. In the stochastic dynamic program, both the next destination x^{dest} and the amount to deliver x^{amount} need to be decided. To simplify this action variable, we set the latter equal to the maximum amount possible considering the demand at the node and the payload of the vehicle, i.e., $x^{\text{amount}}_{\hat{v},n} = \min(d_n, q_{\hat{v}})$. This simplification reduces the action variable to only x^{dest}.

6. Cases and experimental settings

This section covers the experimental design. Section 6.1 introduces the problem instances on which we evaluate our solution methods. Section 6.2 introduces the benchmark methods based on mathematical programming and dynamic heuristics. In Section 6.3, we describe the experimental settings used to evaluate the solution methods.

6.1. Problem instances

To analyze both algorithmic performance and provide practical insights, our numerical analysis evaluates stylized benchmark instances adapted from Solomon (1987) as well as two historical disaster cases, namely the 2015 earthquake in Nepal and the 2018 tsunami in Indonesia. The time horizon for the operation for all instances is an eight-hour working day, i.e., $t^{\max} = 480$ minutes. We now discuss the modeling of the travel time uncertainty, the adapted Solomon instances, the real-world cases, and the vehicle fleets for each instance.

6.1.1. Travel time uncertainty

Travel time uncertainties are differentiated to reflect that, e.g., parts close to the path of a hurricane or near the epicenters of earthquake aftershocks exhibit a larger variance in travel time. We assume planners are able to come up with reasonable estimates for the probability distribution functions of travel times in the area. We assume triangular distributions, which can be constructed easily with minimal information, requiring only estimates for minimum, maximum, and mode. This is common practice in situations in which data about the actual distribution is lacking, which is typically the case in a disaster area. To quantify uncertainty, each node has an associated uncertainty factor that is low, medium, or high. For the arc connecting two nodes, the travel time depends on the nodes' uncertainty factors. To obtain stochastic travel times between nodes, we multiply the expected travel time with a stochastic variable from a triangular distribution with a mean of 1. The distributions used to reflect travel time uncertainties between nodes are presented in Table 2.

Table 2
Triangular distributions for travel time multiplier between nodes, with T[min, max, mode].

	Low	Medium	High
Low	T[0.6, 1.6, 0.8]	T[0.45, 1.825, 0.725]	T[0.3, 2.05, 0.65]
Medium	T[0.45, 1.825, 0.725]	T[0.3, 2.05, 0.65]	T[0.15, 2.275, 0.575]
High	T[0.3, 2.05, 0.65]	T[0.15, 2.275, 0.575]	T[0, 2.5, 0.5]

6.1.2. Adapted solomon instances

To test the algorithmic performance under a variety of circumstances, we adapt the R101, C101, and RC101 instances of Solomon (1987). These instances are generally used to benchmark vehicle routing problems with time windows (VRPTW), with locations randomly scattered over the area (R), clustered in groups of locations (C), and a mix of randomized and clustered locations (RC). As our problem differs from the general VRPTW in terms of the objective function and problem characteristics (e.g., maximizing deliveries, multiple vehicles types, and stochastic travel times), we adapt the instances and generate uncertainty levels, random demands, and time windows, and determine expected travel times for both trucks and UAVs. We create six instances based on the Solomon instances: three instances with 25 demand nodes, and three instances with 100 demand nodes, adopting the location coordinates used in R101, C101, and RC101.

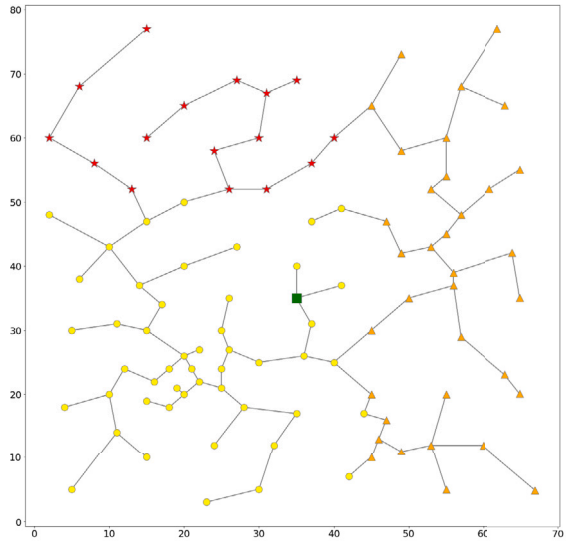
We now detail the modifications. For each node, we sample a demand between 1 and 5 from a discrete uniform distribution and generate a random time window between 60–180 min. To obtain realistic estimates of travel times for trucks, we create a network between nodes by generating a minimum spanning tree and extending this tree by iteratively inserting direct arcs between nodes for which the ratio between the Euclidean distance and the shortest path in the network is below a predetermined threshold, similar to Chen et al. (2019). Subsequently, we compute an origin–destination matrix that contains the shortest paths between all node pairs. Preliminary experiments showed that reasonably connected networks were provided with thresholds of 0.1, 0.2, and 0.3 for R101, RC101, and C101, respectively. The distances between some locations are Euclidean, but most distances are longer due to road network limitations, with distances exceeding the Euclidean distance by up to a factor 10 (see Figs. 2(a), 2(b), and 2(c)). The colors in the figures represent the uncertainty levels, differentiating the impact per district. Based on the real-world case detailed in Section 6.1.3, we calculate expected truck travel times assuming a truck driving speed of 30 km/h. For UAVs, we calculate travel times based on Euclidean distances between all nodes and a flight speed of 120 km/h. For the smaller instances, we have selected a fixed set of 25 random demand nodes out of the original 100 demand nodes and use the same distance matrix as obtained for the 100 demand node instances.

6.1.3. Real-world cases

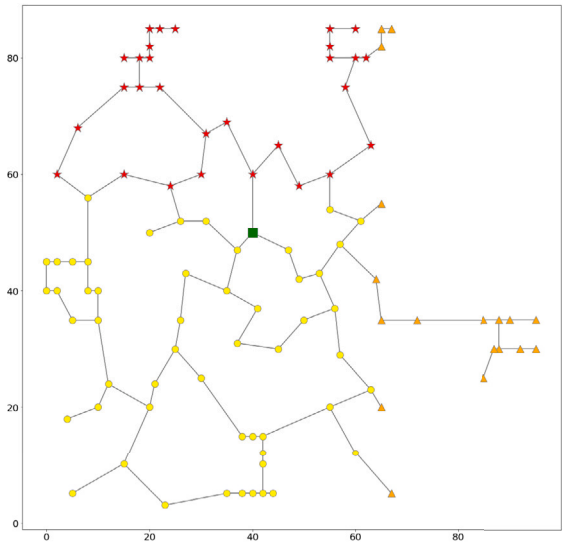
The real-world cases we analyze are the disaster areas after the 2015 earthquake in Nepal and the 2018 tsunami in Indonesia. First, in 2015, a 7.8M earthquake caused major disruption in Nepal. Eleven districts near the capital city Kathmandu were affected the most, where the mountainous surroundings posed challenges to road transport. We focus on two districts that were significantly disrupted: Nuwakot and Dhading. Based on data collected from a field trip in August 2021 (carried out as part of this research) along multiple municipalities in the affected districts, we composed a set of 84 demand nodes and a depot located in Dhading Besi, the district headquarter of the Dhading district (see Fig. 3(a)). We marked uncertainty levels for all nodes, with high uncertainty nodes close to the epicenter, and medium uncertainty nodes up in the mountains. Since detailed data about demand is not available, we generate random demands and time windows similar to the Solomon instances. Second, in 2018, a 7.5M undersea earthquake and a tsunami that followed resulted in a massive disruption of the city of Palu and the surrounding areas of the Palu Bay on the Indonesian island of Sulawesi. Data from a displacement tracking matrix (International Organization for Migration, 2019) with locally collected data from 2018 allows us to extract a set of 343 demand nodes and a depot at the Palu Airport (see Fig. 3(b)). We marked the uncertainty levels of each node with high uncertainty nodes along the coastline and medium uncertainty levels in areas that experienced soil liquefaction. Demands and time windows are again determined randomly. To obtain realistic estimates of actual travel times of trucks in the Nepal and Indonesia cases, we used ESRI ArcMap 10.7 to determine expected travel times between each pair of nodes, yielding approximate average travel speeds of 30 km/h in both areas.

6.1.4. Vehicle fleets

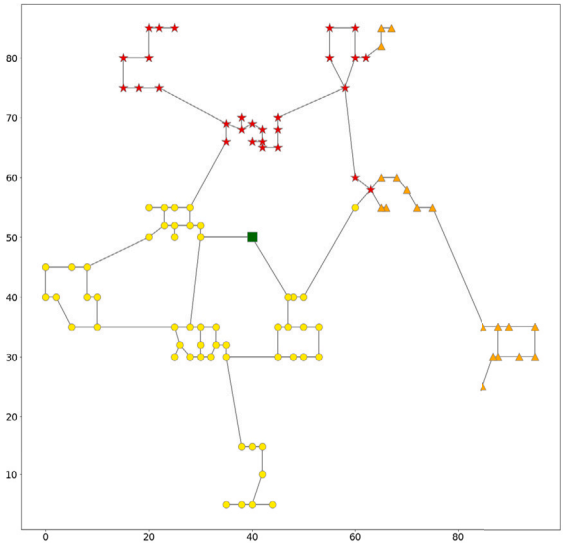
For our experiments, we analyze two fleet configurations: a fleet with only trucks, and a mixed fleet in which 50% of the trucks are replaced by UAVs. We scale fleet sizes to the instance sizes. For the Solomon-based instances with 25 nodes we deploy two vehicles, for the 100 node instances eight vehicles, for the Nepal case six vehicles, and for the Indonesia case 16 vehicles. We assume that a truck has a capacity of 10 units. With demands between one and five, a truck can visit at least two and at most 10 nodes. The specifications of the UAV are inspired by the MiniFreighter from Wings For Aid (Bamsey, 2021). As this UAV has a flight range of 500 km, we ignore range limitations, since 500 km is much larger than any distance in each of the disaster areas. Due to its ability to drop cargo boxes, it does not require to land at the demand nodes, and in practice, it only needs an unpaved 300-meter air strip nearby a depot. For the experiments, we assume that UAVs carry one unit of demand, and travel with a constant speed of 120 km/h following Euclidean distances, as their flights are not influenced by road network structure and infrastructural damages.



(a) Solomon R - 101 nodes - threshold = 0.1



(b) Solomon RC - 101 nodes - threshold = 0.2



(c) Solomon C - 101 nodes - threshold = 0.3

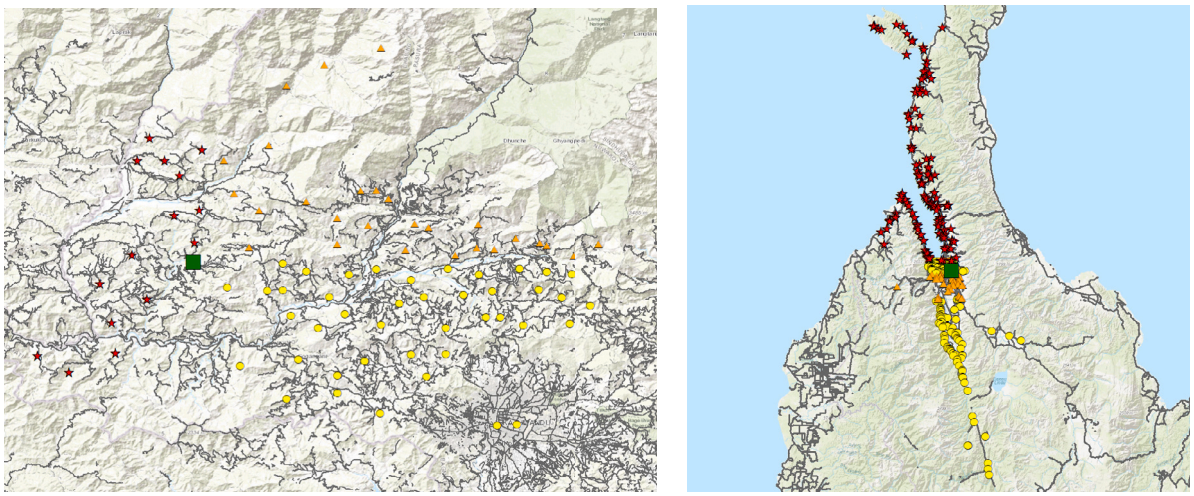
Fig. 2. Maps of adapted Solomon instances with all nodes and arcs. The depots are marked with green squares, low uncertainty areas with yellow circles, medium uncertainty areas with orange triangles, and high uncertainty areas with red stars.

6.2. Benchmark methods

We benchmark the two reinforcement learning methods with several other approaches. First, we develop a deterministic mixed integer quadratic program (MIQP) for the static variant of the problem, for which commercial solvers can provide optimal solutions. Second, we develop three dynamic heuristics – inspired by decision rules used in practice – that take sequential decisions based on the current state. The benchmark methods are conceptually described in this section. The formulation of the mathematical program is detailed in Appendix A.1 and heuristics procedures are detailed in Appendix A.2.

6.2.1. Static solution approach

We approximate the stochastic dynamic problem with a static deterministic variant with the same objective function and constraints but where the stochastic travel times are replaced by expected values. We develop a MIQP, with which we can obtain optimal solutions of the static deterministic problem variant using general commercial solvers, in our case Gurobi 10.0.0. To make a proper comparison with the stochastic dynamic variant, we use the routing sequences of the MIQP solutions and evaluate these by performing Monte Carlo simulations with sampled travel times. This way, the MIQP solution is evaluated under equal conditions



(a) Nepal - 85 nodes (b) Indonesia - 344 nodes

Fig. 3. Maps of the disaster areas in Nepal and Indonesia with all nodes and main roads. The depots are marked with green squares, low uncertainty areas with yellow circles, medium uncertainty areas with orange triangles, and high uncertainty areas with red stars.

as the reinforcement learning methods. Despite random realizations of travel times, the routing sequences can be feasibly executed. This potentially results in higher waiting times for early arrivals and additional lateness penalties for arrivals after the end of the time windows. The operation will be terminated at $t^{\max}$, skipping potential remaining nodes at the end of the day. Hence, the static solution serves as a pre-made plan that is executed in a stochastic environment (given by the Monte Carlo simulation), which we refer to as the stochastic evaluation of the static solution.

6.2.2. Dynamic heuristics

We configure three heuristics that dynamically decide on the next destination to visit whenever a vehicle becomes available. The heuristics are based on expected travel times (i.e., distance to the locations), the reward of deliveries (i.e., number of relief goods and locations), and the corresponding time windows (i.e., promoting punctuality). Decision rules are commonly used in humanitarian practice, as they are comprehensive, do not need to be trained, and require minimal computational runtime, such that they are easily implemented and decisions are made quickly in a chaotic disaster environment. We refer to Gralla and Goentzel (2018) for an evaluation of practice-based heuristics in humanitarian operations. The authors find that the best heuristics send cargo to the closest destinations or to high-priority destinations and these heuristics have a relatively good performance compared to a mathematical programming solution. Our heuristics also relate to dynamic dispatching rules for scheduling (see, e.g., Raghu and Rajendran, 1993) or queueing policies (see, e.g., Adan et al., 2001), as travel times can be considered as processing times or queue lengths, locations as jobs or customers, vehicles as machines or servers, and time windows as release and due dates. As we can observe the state when making the decision, we may utilize decision rules like the earliest due date, least slack, shortest queue policy, or combined priority rules. As for the static MIQP solutions, the heuristic performances are evaluated dynamically via Monte Carlo simulations. We describe the three heuristics based on (i) the nearest locations, (ii) the highest reward, and (iii) a specific priority.

NearestTW. First, we devise a nearest-neighbor heuristic (similar to Mills et al., 2018; Anuar et al., 2022), denoted as NearestTW. For the available vehicle, the node with the shortest expected travel time is selected as the next destination. To promote punctuality (i.e., prevent earliness or lateness), the next destination can only be selected when the arrival is expected to fall within the destination's time window. Note that arriving before or after a time window may still occur due to the stochastic realizations of travel times. Still, NearestTW is expected to generate solutions that perform well in terms of timeliness.

GreedyTW. Second, we construct a greedy heuristic (similar to Mills et al., 2018; Yu et al., 2021; Fan et al., 2022), denoted as GreedyTW. This heuristic chooses as next destination the node with the highest expected reward gained from visiting that location. If multiple nodes will provide an equally high expected reward, it will select the node with the shortest expected travel time. As for NearestTW, nodes are only considered when the expected arrival falls within the destination's time window. If no location is feasible or the vehicle is empty, the vehicle will return to the depot and reload, which is always allowed. GreedyTW is expected to perform well in terms of cumulative reward.

Priority. Third, we deploy a heuristic denoted by Priority that balances rewards and travel times. To this end, we design combined priority rules for trucks and UAVs. For trucks, we use a priority rule based on the expected reward divided by the travel time. For UAVs, we pick the location with the highest expected reward, as time division would overly hinder visiting remote locations. To determine the priority, we use expected rewards based on probability functions and conditional expectations of the travel times,

instead of rewards based only on location and delivery amount. The expected reward also includes penalties for both earliness and lateness, allowing some flexibility in location selection. The expected lateness penalty follows the reward function and is based on the probability of arriving late and the expected minutes arriving after the time window when being late. The earliness penalty is calculated based on the conditional probabilities of arriving early, multiplied by a user-selected earliness penalty. We motivate our setting for the earliness penalty as follows. In the Solomon instances, a vehicle can collect on average a maximum reward of 75 in 8 h. If a vehicle has to wait, we assume it loses half of the maximum possible reward per time unit, corresponding to an earliness penalty of 4.6875 per hour.

6.3. Experimental settings

In this section, we detail the RL training settings, the neural network architectures, the objective function weights, and the evaluation procedure. Table 3 provides an overview of the instance-specific experimental settings and Table 4 summarizes the general experimental setup.

RL training. For the reinforcement learning methods, we limit training time to 4 h. Longer computation times are not realistic in practice, as the algorithm must be trained before the start of every day. For each iteration of sampling and training in API and AVI, we generate 50,000 samples and perform 25 replications within each sample. For the ϵ-greedy approach to balance exploration and exploitation, we consider for AVI a linearly decreasing scheme that starts at $\epsilon = 0.5$ and decays with 0.01 every iteration until it is 0. For API, we consider a fixed value of $\epsilon = 0.05$. We update the neural networks during training using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 and a mini-batch size of 256.

Neural network architecture. To define the neural networks used in the RL methods, we use multilayer perceptrons (MLPs) with fully connected hidden layers and a ReLU activation function. Based on a preliminary study, we found that networks with three hidden layers strike a good balance between performance, convergence, and computational effort. Specifically, we set the first hidden layer roughly 1.5–2 times as large as the input layer and halve subsequent layer sizes. For example, the Solomon-based instances with 25 locations have an input of 122 variables, and we deploy an MLP with three hidden layers of size 256, 128, and 64 respectively, shortly denoted as a 256-128-64 MLP. Network architectures per instance are detailed in Table 4. Note that for all architectures, the last hidden layer is larger than the number of locations.

Objective function. To weigh the components of the objective function, we set $\xi_1 = 1$ for the number of goods delivered, $\xi_2 = 3$ for the number of visited locations, and $\xi_3 = \frac{1}{60}$ as lateness penalty weight. Since the average demand per node is three for all instances (sampled from $\mathcal{U}\{1, 5\}$), the objective of delivering goods is balanced with visiting locations. The lateness penalty of $\frac{1}{60}$ multiplied by lateness values $c_{k,\hat{v}_k}$ generates negative rewards upon late arrival and nullifies the rewards for a delivery when a vehicle arrives an hour late.

Evaluation. We evaluate all solutions found for each instance by performing 10,000 Monte Carlo simulations to obtain reliable results. All model formulations are implemented in C++. The RL methods are solved and evaluated using Dynaplex (Akkerman et al., 2022) and the MIQP is solved in Gurobi 10.0.0. The experiments are performed on an 32-core Intel® Xeon® Platinum 8272CL 2.6 GHz computer with 64 gigabyte RAM. In line with the time limit on RL training, we terminate MIQP solving after four hours, after which the acquired solution is evaluated. We denote the static solution values (based on expected travel times) of the mathematical model by MIQP-stat, whereas the stochastic Monte Carlo evaluation (based on realized stochastic travel times) is denoted by MIQP-stoch. Preliminary tuning experiments led us to adopt the following non-default options: reduce the number of non-zero values in the pre-solved model (PreSparsify = 1), and let the no relaxation heuristic search for high-quality feasible solutions for one hour before solving the root relaxation (NoRelHeurTime = 3600). The three heuristic methods have negligible execution time and are applied directly during the stochastic evaluation phase.

7. Experimental results

This section analyzes the experimental results, starting with the Solomon instances in Section 7.1 and followed by the real-world cases of Nepal and Indonesia in Section 7.2.

7.1. Analysis of solomon instances

In this section, we present the computational results and analyze the performance and robustness of the solution methods and fleet configurations for the adjusted Solomon instances with 25 and 100 demand nodes. We aim to identify under which conditions the methods perform well and how robust they are in terms of performance. All reported results are based on 10,000 Monte Carlo evaluations. As described in Section 6, 4 h of training or solving are allotted to the RL- and MIQP methods respectively. For the reinforcement learning approaches, we present convergence during training. For the MIQP solutions, optimality gaps are reported in case optimal solutions are not found within the time limit. For each method, we reflect on performance and robustness.

Table 3
Instance specific experimental settings.

Instance	Case	Number of trucks	Number of UAVs	Number of demand nodes	Number of state variables	Hidden layers of the MLP in VFA and PFA
1	C25	2	0	25	122	256-128-64
2	C25	1	1	25	122	256-128-64
3	R25	2	0	25	122	256-128-64
4	R25	1	1	25	122	256-128-64
5	RC25	2	0	25	122	256-128-64
6	RC25	1	1	25	122	256-128-64
7	C100	8	0	100	470	1024-512-256
8	C100	4	4	100	470	1024-512-256
9	R100	8	0	100	470	1024-512-256
10	R100	4	4	100	470	1024-512-256
11	RC100	8	0	100	470	1024-512-256
12	RC100	4	4	100	470	1024-512-256
13	Nepal	6	0	84	390	512-256-128
14	Nepal	3	3	84	390	512-256-128
15	Indonesia	16	0	343	1506	2048-1024-512
16	Indonesia	8	8	343	1506	2048-1024-512

Table 4
General experimental settings.

Settings	
Demand distribution per node	$\mathcal{U}(1,5)$
Time window size (minutes)	$\mathcal{U}(60,180)$
t^{max}	480
ξ_1	1
ξ_2	3
ξ_3	1/60
Truck speed	30 km/h
Truck capacity	10
UAV speed	120 km/h
UAV capacity	1
Computational time limit	14,400 s
Monte Carlo evaluations	10,000
Samples per PFA/VFA iteration	50,000
Replications per PFA/VFA sample	25

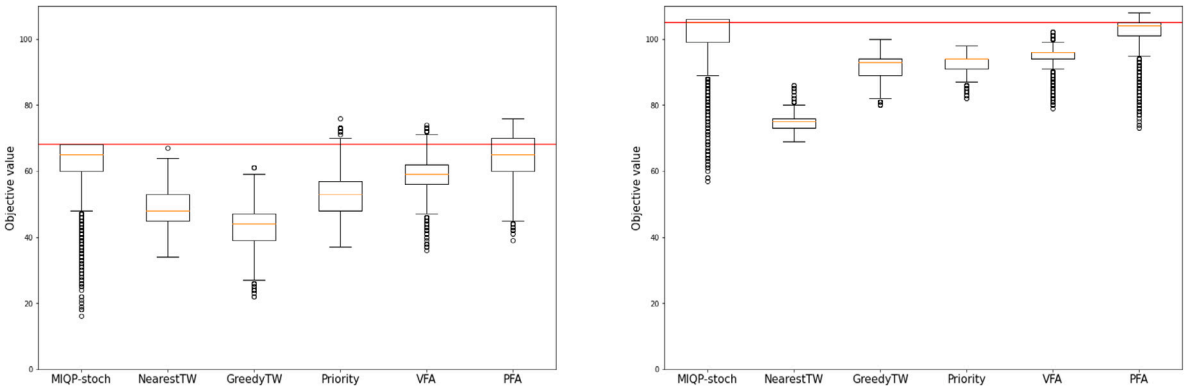
7.1.1. Small instances

Table 5 summarizes the results for the 25-node Solomon instances. We observe that PFA outperforms the other methods in 5 out of 6 instances. Furthermore, we note that MIQP-stoch performance (evaluated using realized stochastic travel times) is consistently worse than MIQP-stat performance (using expected travel times). The difference is caused by additional lateness penalties and premature returns to the depot, and is significantly larger for the truck-only instances (about 7.7% decrease) compared to the truck-UAV instances (about 2.7%). UAV deployment offers considerably more robust results, as the UAVs are not subject to travel time uncertainty. As the decrease is more than 50%, UAVs also take over more challenging locations from trucks, decreasing the lateness penalties. Nonetheless, the difference between MIQP-stat and MIQP-stoch is relatively small, which means that the pre-determined static plans are not significantly affected by travel time uncertainty. This can be explained by the effect that fluctuations in travel time partly nullify each other, e.g., a trip shorter than expected followed by a trip that is longer than expected can result in the same objective value as found by the static plan. Hence, the MIQP may be preferred over the other methods for situations with limited travel time uncertainty. The heuristics clearly perform worse than both the MIQP outcomes and reinforcement learning methods. Interestingly, NearestTW consistently performs better than GreedyTW for the truck cases, whereas GreedyTW is consistently better than NearestTW for the truck-UAV instances. NearestTW generates rather efficient truck routes, as trucks combine multiple locations in close proximity in a single trip. In contrast, GreedyTW selects locations with higher rewards, which are more profitable for UAVs that are flying back and forth from the depot. Compared to the other heuristics, the Priority rule obtains good results for the truck-UAV cases, balancing rewards and travel times. In all cases, the deployment of UAVs results in higher objective values. Deploying UAVs improved the objective value by around 36% for the C and RC instances and even by 56% for the R instance, in which locations are scattered over the area.

Due to the stochastic travel times, routes and punctuality vary between individual evaluations. This results in various outputs of the objective function for the obtained solutions. To illustrate this, Fig. 4 shows the variation of objective values for each method for the R25 instances. The robustness in the truck-only instance is comparable across methods. For the truck-UAV instance, the results of all methods are more robust than in the truck-only instance. Nonetheless, the objective values between different methods differ, with VFA being the most robust method with the smallest interquartile range. The MIQP-stoch obtains good results but also shows the largest downward outliers. Similar observations can be made for the C25 and RC25 instances, whose boxplots are shown in Appendix B.

Table 5
Results of the different solution methods for the Solomon instances with 25 demand nodes. The potential maximum reward when covering all demand without lateness is 150. The gap is the percentage difference between the objective value and objective bound found by Gurobi when reaching the time limit. The best solutions are marked in bold.

Instance	Fleet	MIQP-stat	Gap	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
C25	2 trucks	74.00	96%	70.71	52.93	49.52	50.70	74.75	**78.89**
C25	1 truck, 1 UAV	108.03	39%	105.31	86.60	91.54	96.52	102.26	**107.00**
R25	2 trucks	68.00	104%	62.59	48.82	43.14	53.86	60.17	**66.16**
R25	1 truck, 1 UAV	104.87	34%	100.82	74.80	91.52	93.43	95.44	**103.09**
RC25	2 trucks	66.05	84%	58.96	53.13	46.18	42.84	60.15	**64.12**
RC25	1 truck, 1 UAV	89.93	23%	**87.51**	66.96	78.32	76.06	81.34	87.20



(a) R25 - 2 trucks (b) R25 - 1 truck, 1 UAV

Fig. 4. Objective values for all solution approaches for the R25 instances. For reference, a red line has been added corresponding to the objective value of the MIQP-stat solution objectives. PFA performs overall best, also considering the interquartile ranges.

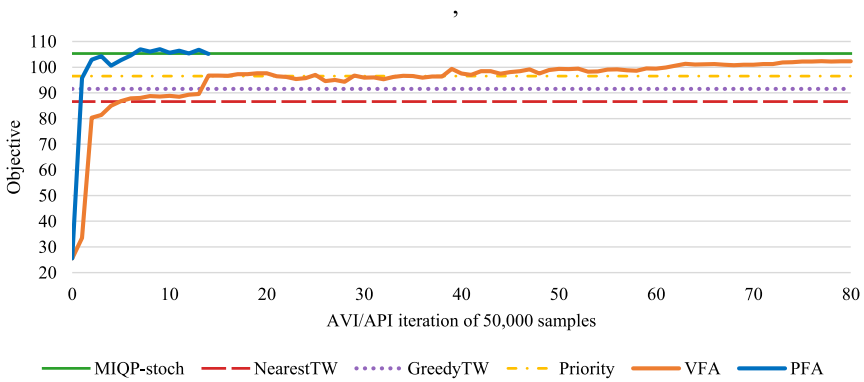


Fig. 5. Convergence during training of the VFA and PFA for the C25 instance with trucks and UAVs. PFA only performs 15 iterations in the 4-hour time limit, but convergences in much fewer iterations than VFA, which improves more gradually.

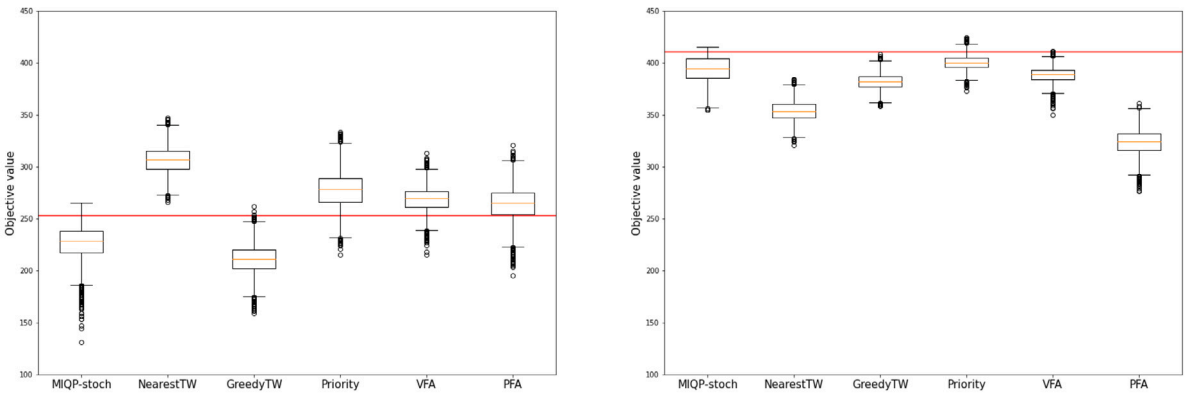
Fig. 5 illustrates the convergence of VFA and PFA for the C25 instance with 1 truck and 1 UAV. As sampling is computationally extensive for the PFA, only 15 iterations are performed. Nonetheless, it converges in 8 iterations and outweighs the objective value of the VFA, which collects substantially more samples due to its computational efficiency. For both PFA and VFA, displayed convergence behavior is representative for the other small instances.

7.1.2. Large instances

The results for the large instances are shown in Table 6. Among all methods, no consistently dominant method was found. As the large instances are computationally challenging, PFA was not able to learn a performant policy within the computational time limit. VFA, which is more sample-efficient than PFA, generated more competitive results, especially for the truck-UAV instances. The MIQP solutions perform better in the cases with both trucks and UAVs than in the truck-only instances. Since UAVs only perform trips to a single demand node, this problem is easier to solve than the problems with more trucks (including more complex routes). Furthermore, the difference in performance between MIQP-stat and MIQP-stoch is significantly smaller for the truck-UAV instances

Table 6
Results of the different solution methods for the Solomon instances with 100 demand nodes. The potential maximum reward when covering all demand without lateness is 600. The gap is the percentage difference between the objective value and objective bound found by Gurobi when reaching the time limit. The best solutions are marked in bold.

Instance	Fleet	MIQP-stat	Gap	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
C100	8 trucks	252.6	138%	226.8	**306.9**	211.4	278.3	271.1	281.3
C100	4 trucks, 4 UAVs	410.2	46%	394.7	353.6	382.3	**400.8**	390.4	323.9
R100	8 trucks	226.5	165%	198.0	**290.4**	202.0	266.8	260.9	261.1
R100	4 trucks, 4 UAVs	419.3	43%	414.1	362.8	419.1	420.0	**429.6**	350.4
RC100	8 trucks	251.9	138%	218.5	**279.6**	229.2	276.6	265.5	227.5
RC100	4 trucks, 4 UAVs	374.4	60%	357.6	309.1	**370.3**	357.2	355.0	310.8



(a) C100 - 8 trucks (b) C100 - 4 trucks, 4 UAVs

Fig. 6. Objective values for all solution approaches for the C100 instances. For reference, a red line has been added corresponding to the objective value of the MIQP-stat solution objectives. NearestTW is considerably better in the truck instance, whereas the Priority rule performs best in the truck-UAV variant.

compared to the truck-only instances. While only obtaining the best result once, the Priority rule never scores more than 10% below the best outcome, thus providing the most consistent performance. The NearestTW heuristic performs best for the truck-only instances. Since not all locations can be visited within the time horizon, it is efficient to visit as many locations as possible close to the depot. Furthermore, since travel time variability scales with the travel time, shorter trips, e.g., to the nearest location, have less absolute variance, reducing both early and late arrivals. Similar to the smaller instances, GreedyTW performs better than NearestTW in the truck-UAV instances, as UAVs benefit from visiting locations with a high reward disregarding travel times. Again, the deployment of UAVs achieves the largest improvement over truck-only fleets in the random instances (48%) and also shows significant improvements in the other instances. For clustered instances, trucks can more easily combine locations, whereas with scattered locations, the faster UAVs that fly back-and-forth are having an advantage.

Fig. 6 shows the boxplots of outcomes for each method for the larger C100 instances, in which NearestTW performs best in terms of both objective value and robustness for truck-only fleets. For mixed truck-UAV fleets, the Priority heuristic yields marginally higher objective values than MIQP and VFA; the heuristics and VFA provide slightly more robust results than the MIQP solutions. PFA achieves noticeably worse results than for the small instances and a larger spread in outcomes. The poor performance of PFA is due to the computational limits, which did not provide sufficient training time to learn a well-performing policy. Appendix B shows the boxplots for the other R100 and RC100 instances, with comparable results.

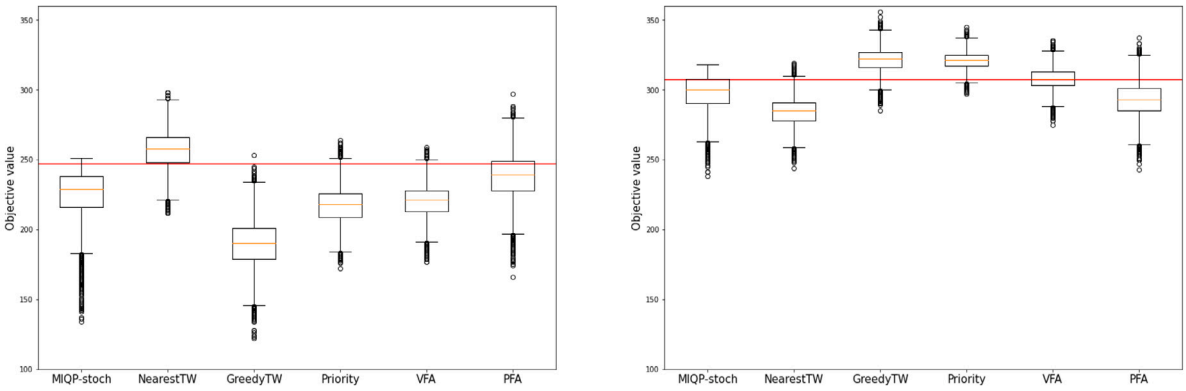
7.2. Analysis of real-world cases

This section analyzes the results of the real-world cases consisting of the 2015 earthquake in Nepal earthquake and the 2018 tsunami in Indonesia. We again present the performance and robustness of all methods, and also provide insight into the individual components of the objective function and differences in decision-making between the solution methods.

The results for the Nepal- and Indonesia cases are presented in Table 7. Insights for the Nepal case (84 nodes) are roughly comparable to those of the large Solomon-based instances. NearestTW performs best for the truck-only fleet and GreedyTW performs best for the mixed-fleet settings. For the large Indonesia instance (343 nodes), Gurobi could not obtain feasible solutions within the computational time limit. The learning-based methods also experienced difficulty in finding competitive policies within the time limit. Similar to the larger Solomon instances, NearestTW shows the best results for the truck-only instances. Given the large number of locations, trucks can efficiently visit multiple locations in one trip. GreedyTW obtains the best results for fleets with UAVs, as GreedyTW dispatches UAVs to unvisited locations, thereby obtaining additional location rewards. These results show a clear distinction between vehicle types and the type of routing. The Priority rule performs reasonably well with a performance

Table 7
Results of the different solution methods for the Nepal and Indonesia cases. Theoretical maximum scores are 504 and 2058. A feasible solution could not be found within the time limit for the MIQP of the Indonesia case. The best solutions are marked in bold.

Instance	Fleet	MIQP-stat	Gap	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
Nepal	6 trucks	246.8	104%	225.2	**257.5**	190.2	218.5	220.9	239.1
Nepal	3 trucks, 3 UAVs	307.4	64%	298.7	284.7	**321.8**	321.3	308.5	293.3
Indonesia	16 trucks	n.a.	n.a.	n.a.	**1184.1**	866.7	1014.7	707.8	919.9
Indonesia	8 trucks, 8 UAVs	n.a.	n.a.	n.a.	1221.2	**1313.4**	1292.6	1076.8	805.1



(a) Nepal - 6 trucks (b) Nepal - 3 trucks, 3 UAVs

Fig. 7. Objective values for all solution approaches for the Nepal instances. For reference, a red line has been added corresponding to the objective value of the MIQP-stat solution objectives.

falling in between the other heuristics. For the RL training on the Nepal instance, convergence patterns for both VFA and PFA are comparable to those shown in Fig. 5. In the Indonesia instances, the convergence patterns were erratic as policies did not converge within the computational time limit.

Boxplots showing the robustness in objective values of the Nepal instances are presented in Fig. 7. GreedyTW displays the worst robustness in the truck instance, whereas it performs as one of the best in the truck-UAV instance. In both instances, the Priority rule provides the smallest interquartile range. For mixed fleets, GreedyTW and the Priority rule outperform static solution values most of the time. The same observation holds for NearestTW applied on truck-only fleets.

To gain insights into the different methods and the role of UAVs, we perform an additional analysis on the Nepal instances. We decompose the objective values into the individual objectives and analyze the decisions made by the different methods. Recall that the objective value consists of (i) the number of goods delivered, (ii) the number of different locations visited, and (iii) the lateness penalty. For each method, these partial objective values are shown in Table 8. The MIQP performance heavily relies on the number of locations visited and shows relatively high lateness penalties. The substantial difference between MIQP-stat and MIQP-stoch in the truck-only instance is mainly attributed to the increase in lateness penalties, whereas the difference between those in the truck-UAV variant remains limited due to a much smaller increase in lateness penalties. The MIQP benefits from the freedom to decide on the delivery amount at each location and thus can visit more locations with a truck. In the truck-only instance, the MIQP solution visits on average 52 locations and delivers 1.7 units per location, whereas the other methods visit on average 37 locations and deliver 3.2 units per location. Due to the action space reduction detailed in Section 5.4, the reinforcement learning methods (as well as the heuristic methods) always deliver as much as possible at each node given the remaining demand and vehicle payload. Compared to MIQP, the dynamic heuristics and learning-based methods incur considerably lower lateness penalties, approximately 80% reduction for the truck-only instance, and around 90% in the case with UAVs. Hence, both sequential decision-making and the deployment of UAVs considerably improve the reliability of humanitarian operations. As NearestTW always selects the nearest feasible node, travel time variability is relatively low, thereby limiting both earliness and lateness. NearestTW is the best-performing method in the truck-only case and covers about 52% of the locations and 51% of the demand. In the UAV instance, GreedyTW covers 82% of the locations and 47% of the demand. Comparing the best solution of the truck instance to the mixed fleet instance, the mixed fleet visits 25 additional locations against a slight reduction in the amount delivered. From an egalitarian perspective, this is a significant improvement. However, since the number of goods delivered is slightly lower, the performance decreases from a utilitarian perspective.

Table 9 reports for each method (i) the average distance to the depot, (ii) the average demand of visited nodes, and (iii) the average uncertainty levels of traversed arcs by trucks and UAVs. By comparing the differences between methods and between the truck-only instance and the truck-UAV instance, we provide insight into the decisions of the methods for both fleet types. First, all methods generally assign locations nearby the depot to trucks and farther locations to UAVs. Naturally, this differentiation is least visible for NearestTW, which assigns all vehicles, including UAVs, to their closest feasible locations. The differentiation is

Table 8
Individual objective values of the methods for the Nepal instances.

6 trucks	MIQP-stat	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
Total reward	246.8	225.2	**257.5**	190.2	218.5	220.9	239.1
# locations visited	**53**	52.08	43.71	28.27	37.18	36.09	40.41
Amount delivered	93	89.14	**127.65**	111.28	108.45	116.16	124.53
Weighted lateness penalty	5.20	20.16	**1.26**	5.92	1.51	3.51	6.68
3 trucks, 3 UAVs	MIQP-stat	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
Total reward	307.4	298.7	284.7	**321.8**	321.3	308.5	293.3
Locations visited	**75.00**	74.35	47.70	68.63	69.99	64.41	54.63
Amount delivered	97.00	94.92	**141.87**	118.50	114.75	117.66	131.35
Weighted lateness penalty	14.60	19.25	**0.27**	2.53	3.42	2.34	1.97

Table 9
Distance, demand, and uncertainty analysis of the decisions of all methods on the Nepal instances. VFA has learned similar characteristics of decisions for the truck-UAV instance as the best-performing GreedyTW heuristic.

	Fleet: vehicle	MIQP-stat	MIQP-stoch	NearestTW	GreedyTW	Priority	VFA	PFA
Average distance to depot (km)	Truck-only: truck	65.4	66.4	52.8	68.4	53.7	48.8	58.8
	Trucks & UAVs: truck	62.6	63.5	47.9	64.5	47.0	46.3	52.2
	UAV	72.6	72.6	58.5	70.2	77.4	79.3	77.4
Average demand of visited node	Truck-only: truck	2.96	2.94	3.12	4.08	3.13	3.31	3.29
	Trucks & UAVs: truck	2.88	2.87	3.24	4.68	3.27	4.25	3.75
	UAV	3.12	3.12	3.72	2.69	2.97	2.82	2.87
Average uncertainty level between nodes (low = 1, high = 5)	Truck-only: truck	3.36	3.34	3.22	3.41	3.62	3.44	2.97
	Trucks & UAVs: truck	3.75	3.75	3.31	3.49	3.47	3.37	3.46
	UAV	3.14	3.13	3.84	3.73	3.74	3.69	3.64

strongest for the VFA, but does not aid in achieving the best results. Second, we observe a similar relation between the average demand per visited node and the visiting vehicle type. For GreedyTW, the Priority rule, and the learning-based methods, trucks tend to visit high-demand nodes and UAVs low-demand nodes. NearestTW does not exhibit this behavior. Notably, there is also little differentiation in the demand of visited locations for the MIQP, as trucks perform multiple smaller deliveries such that differentiation by location demand becomes less impactful. Third, we do not observe differentiation with respect to uncertainty levels for the MIQP, NearestTW, and GreedyTW methods, as these do not consider this. For the RL methods and the Priority rule, we observe a slight difference with UAVs visiting locations subject to higher travel time uncertainty, which is explicitly modeled in the Priority rule and learned implicitly by the RL methods. However, PFA visits high-uncertainty locations with both trucks and UAVs in the mixed fleet instance. One could argue that a UAV can better carry out trips with higher uncertainties, as a truck taking that trip has a higher probability of arriving outside the time window than taking low-uncertainty trips. Similarly, a UAV might be more suited to visit nodes with narrow time windows. Nonetheless, these decisions are not trivial, because they also depend on the trip length and the planned arrival times of trucks. Overall, the in-depth behavioral analysis illustrates complex and nuanced trade-offs, highlighting the need for advanced decision-making methods for the problem.

8. Conclusions

After natural disasters such as floods and earthquakes, the local road infrastructure is often heavily damaged. The resulting stochastic travel times pose a significant challenge for trucks performing humanitarian relief distribution operations. This challenge may be mitigated by (i) employing dynamic decision-making methods that adapt operations based on disruptions and (ii) deploying unmanned aerial vehicles (UAVs) that are not affected by damaged infrastructure. To study this, we introduced a multi-vehicle, multi-trip, split-delivery vehicle routing problem with soft time windows and stochastic travel times for last-mile relief distribution. We formulated the problem as a stochastic dynamic program and analyze two deep reinforcement learning (RL) methods: value function approximation (VFA) and policy function approximation (PFA). As a benchmark, we solved a deterministic mathematical integer quadratic program (MIQP) assuming expected travel times. We subsequently evaluated the static solution of the MIQP with samples of stochastic travel times using Monte Carlo simulation. We further compared the RL methods with three dynamic heuristics based on a nearest-neighbor principle, a greedy decision, and a priority rule. We analyzed the impact of utilizing cargo UAVs that operate independently from trucks to mitigate uncertainty in humanitarian missions. The weighted multi-criteria objective is to maximize (i) the number of relief goods delivered and (ii) the number of different locations visited, while (iii) penalizing late arrivals. We performed our experiments on adapted Solomon instances of 25 and 100 demand nodes, as well as two real-world cases originating from the Nepal earthquake in 2015 (84 demand nodes) and the Indonesia tsunami in 2018 (343 demand nodes). UAV specifications are derived from a real-world humanitarian logistics partner.

Experimental results show that for instances with 25 demand nodes, PFA outperforms the other methods. PFA's overall objective improves on average 5% compared to the stochastic evaluation of the static solution and improves 32% compared to the heuristics.

Furthermore, the RL methods and the dynamic heuristics show higher levels of robustness with respect to the objective function compared to the evaluated MIQP solutions. Moreover, late arrivals can be significantly reduced by adopting dynamic methods; the analysis of the individual components of the objective function in the Nepal case shows a reduction in lateness penalties of 85%. Nonetheless, the static solutions perform competitively compared to the other methods, also when evaluated with stochastic travel times. The route sequences prescribed by the static solution can be followed to a large extent, and the increase in lateness penalties remains limited due to the fact that travel time uncertainty is partly mitigated by multiple consecutive trips between nodes. In situations with limited uncertainty, these predetermined plans are therefore still useful. For instances with 100 demand nodes, both the RL methods and the MIQP struggle to identify high-quality policies within the imposed computational time limit of 4 h, mimicking time constraints in real-life daily relief operations. In the largest instance, the real-world case of Indonesia with 343 demand nodes, the heuristics achieve the best performance, whereas the MIQP does not yield a feasible solution within the time limit.

Humanitarian logistics often exhibit a tension between egalitarianism and utilitarianism, as sending a truck to small remote locations improves performance from an egalitarian perspective, but decreases performance from a utilitarian perspective. Substituting a part of the truck fleet with cargo UAVs mitigates the tension between visiting as many locations as possible and delivering as many goods as possible. Considering the best solutions found in all instances, the deployment of UAVs yields performance increases between 11% and 56% compared to truck-only fleets. Furthermore, replacing half of the trucks with UAVs reduces lateness penalties during operations by approximately 60% compared to truck-only instances, increasing the reliability of humanitarian response operations. With UAVs, larger numbers of nodes are visited (e.g., 57% more nodes in the Nepal case) with only a minor decrease in the total amount delivered (7% in the Nepal case). The deployment of UAVs in humanitarian logistics operations thus provides clear benefits regarding effective operations, taking over more challenging locations from trucks and visiting hard-to-reach locations. The experiments show that UAVs are more likely to visit relatively remote locations with low demand and high travel time uncertainties. Accordingly, a specialized division of tasks between vehicle types materializes for the RL methods in particular. However, this task division is not straightforward, given the delicate nuances and trade-offs in different instances.

The insights from our study fuel future research. The results indicate that UAVs are suitable to (i) visit remote locations, (ii) visit locations with small demand, and (iii) perform trips in which the travel time uncertainty is relatively high. These insights can be used to improve the heuristics, provide input for feature engineering for improving convergence of RL methods, and advance adapted learning mechanisms (e.g., multi-agent learning). Another potential avenue for future research could be to generalize the action space of the RL methods. In the current methods, the amount to deliver at a node is determined by a fixed decision rule, but could also be learned by the RL methods. Other suggestions include the consideration of prioritized locations, handling ad-hoc demand requests, and the analysis of deprivation costs in relief distribution with UAVs to justify the potential increased operational costs of using UAVs. Lastly, we have considered a problem that spans a single day of a humanitarian operation. As response operations usually span many days (up to multiple months), future work should also consider the allocation of resources over longer time horizons.

CRediT authorship contribution statement

Robert van Steenbergen: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Martijn Mes:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision. **Wouter van Heeswijk:** Methodology, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data available at https://data.mendeley.com/datasets/j36zgxnc5p/2.

Acknowledgments

We thank Akshayat Koirala for his scrutiny and perseverance during the local collection of field data about the 2015 earthquake by visiting government officials, NGOs, local agencies, VDC chiefs, and ward members at various villages in the districts of Nuwakot and Dhading in Nepal in August 2021. We thank the three anonymous reviewers for their valuable suggestions and insightful comments during the review process which has greatly enhanced this paper. We also thank Anne Zander for her knowledgeable feedback during the review process. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Details of the benchmark methods

This appendix provides additional details about the benchmark methods. First, we discuss the mixed integer quadratic program (MIQP) and thereafter the dynamic heuristics.

Table 10
Notation for the mathematical formulation.

Sets:		
$\mathcal{N} = \{0, 1, \dots, \lvert\mathcal{N}\rvert\}$		Set of demand nodes in the disaster area, where node 0 represents the depot
$\mathcal{V} = \{1, \dots, \lvert\mathcal{V}\rvert\}$		Set of vehicles
$\mathcal{P} = \{1, \dots, \lvert\mathcal{P}\rvert\}$		Set of trip indices for each vehicle
Parameters:		
Q_v		Capacity of vehicle v, $v \in \mathcal{V}$
$t^{\max}$		Total amount of time within the time horizon
$t_{v,i,j}$		Expected travel time for vehicle v from node i to node j, $v \in \mathcal{V}$, $i \in \mathcal{N}$, $j \in \mathcal{N}$
d_n		Demand requested at node n, $n \in \mathcal{N}$
a_n		Start of time window at node n, $n \in \mathcal{N}$
b_n		End of time window at node n, $n \in \mathcal{N}$
M_1, M_2, M_3		Large positive numbers for Big M constraints
Decision variables:		
$x^{\text{amount}}_{v,p,n}$	$\in \mathbb{Z}_{\geqslant 0}$	Number of goods delivered by vehicle v in trip p at location n, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $n \in \mathcal{N}$
$x^{\text{dest}}_{v,p,i,j}$	$\in \{0, 1\}$	Equals 1 if vehicle v in trip p travels from node i to node j and 0 otherwise, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $i \in \mathcal{N}$, $j \in \mathcal{N}$
$t^{\text{arr}}_{v,p,n}$	$\in \mathbb{R}_{\geqslant 0}$	Arrival time of vehicle v in trip p at location n, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $n \in \mathcal{N}$
$e_{v,p,n}$	$\in \mathbb{R}_{\geqslant 0}$	Minutes early when vehicle v in trip p arrives at location n, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $n \in \mathcal{N}$
$l_{v,p,n}$	$\in \mathbb{R}_{\geqslant 0}$	Minutes late when vehicle v in trip p arrives at location n, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $n \in \mathcal{N}$
$f_{v,p,n}$	$\in \{0, 1\}$	Equals 1 if vehicle v in trip p is the first vehicle to visit node n and 0 otherwise, $v \in \mathcal{V}$, $p \in \mathcal{P}$, $n \in \mathcal{N}$

A.1. Mixed integer quadratic program

The deterministic formulation of the problem using a mixed integer quadratic programming (MIQP) model does not entail stochastic elements of the road network and is instead formulated as a static variant using the expected values of the travel times. To improve the readability, we denote the expected travel times by $t_{v,i,j} = \mathbb{E}[T_{v,i,j}]$. For the mathematical formulation of the multi-trip variant of the VRP, we follow the 4-Index formulation as presented in Cattaruzza et al. (2016). With this formulation, we omit the decision epoch variable k, as we solve the problem for the full horizon at once. Alternatively, we add an additional trip index p to the decision variables, e.g., $x^{\text{dest}}_{v,p,i,j}$, which indicates whether vehicle v in trip p travels from node i to node j. A trip is a single round-trip of a vehicle that starts and ends at the depot and visits one or more demand nodes. With this additional index, we can model multiple trips and split-deliveries while respecting the time limit $t^{\max}$. The maximum number of trips with the trip index p is equal to the sum of all demand $|\mathcal{P}| = \sum_{n \in \mathcal{N}} d_n$. This could occur when a single vehicle performs split-deliveries to fulfill all demand with a single relief unit each trip. When a node receives a vehicle visit for the first time, the number of node visits increases by 1, any additional vehicle visit increases the number of goods delivered, but not the number of visited locations. Hence, to keep track of the number of nodes visited, we introduce the variable $f_{v,p,n}$, which indicates whether vehicle v in trip p is the first vehicle arriving at node n. Table 10 presents an overview of the mathematical notation used in the problem formulation.

The objective function consists of three terms weighted by ξ_1, ξ_2, and ξ_3. The first term is the total number of goods delivered, the second is the number of demand nodes visited at least once, and the third is the weighted lateness of all vehicle arrivals. The lateness penalty is a quadratic term that multiplies the number of minutes late, the number of goods delivered and a binary variable that indicates the first vehicle arrival at the location. In this way, the objective function will yield the same reward as the cumulative reward function in Eq. (2). The formulation of the problem is as follows:

$$\max \xi_1 \sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{P}} \sum_{n \in \mathcal{N}} x^{\text{amount}}_{v,p,n} + \xi_2 \sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{P}} \sum_{n \in \mathcal{N}} f_{v,p,n} - \xi_3 \sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{P}} \sum_{n \in \mathcal{N}} l_{v,p,n}(\xi_1 x^{\text{amount}}_{v,p,n} + \xi_2 f_{v,p,n}) \tag{22}$$

subject to

$$\sum_{i \in \mathcal{N}} x^{\text{dest}}_{v,p,i,n} = \sum_{i \in \mathcal{N}} x^{\text{dest}}_{v,p,n,i} \qquad \forall v \in \mathcal{V}, p \in \mathcal{P}, n \in \mathcal{N} \tag{23}$$

$$\sum_{i \in \mathcal{N}} x^{\text{dest}}_{v,p,i,n} \leqslant 1 \qquad \forall v \in \mathcal{V}, p \in \mathcal{P}, n \in \mathcal{N} \tag{24}$$

$$\sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{P}} x^{\text{amount}}_{v,p,n} \leqslant d_n \qquad \forall n \in \mathcal{N} \tag{25}$$

$$\sum_{n \in \mathcal{N}} x^{\text{amount}}_{v,p,n} \leqslant Q_v \qquad \forall v \in \mathcal{V}, p \in \mathcal{P} \tag{26}$$

$$\sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} t_{v,i,j} x^{\text{dest}}_{v,p,i,j} \leqslant t^{\max} \qquad \forall v \in \mathcal{V} \tag{27}$$

$$x^{\text{amount}}_{v,p,n} \leqslant M_1 \sum_{i \in \mathcal{N}} x^{\text{dest}}_{v,p,i,n} \qquad \forall v \in \mathcal{V}, p \in \mathcal{P}, n \in \mathcal{N} \tag{28}$$

$$M_1\left(\sum_{i\in\mathcal{N}} x^{\text{dest}}_{v,p,i,n} - 1\right) + 1 \leqslant x^{\text{amount}}_{v,p,n} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{29}$$

$$a_n \leqslant t^{\text{arr}}_{v,p,n} + e_{v,p,n} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{30}$$

$$t^{\text{arr}}_{v,p,n} - l_{v,p,n} \leqslant b_n \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{31}$$

$$x^{\text{dest}}_{v,1,0,j}\left(t^{\text{arr}}_{v,1,j} - t_{v,0,j}\right) = 0 \qquad \forall v\in\mathcal{V}, j\in\mathcal{N} \tag{32}$$

$$x^{\text{dest}}_{v,p,0,j}\left(t^{\text{arr}}_{v,p,j} - t^{\text{arr}}_{v,p-1,0} - e_{v,p-1,0} - t_{v,0,j}\right) = 0 \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}\setminus\{1\}, j\in\mathcal{N} \tag{33}$$

$$x^{\text{dest}}_{v,p,i,j}\left(t^{\text{arr}}_{v,p,j} - t^{\text{arr}}_{v,p,i} - e_{v,p,i} - t_{v,i,j}\right) = 0 \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, i\in\mathcal{N}\setminus\{0\}, j\in\mathcal{N} \tag{34}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x^{\text{dest}}_{v,p,i,j} \leqslant M_2 \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x^{\text{dest}}_{v,p-1,i,j} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}\setminus\{1\} \tag{35}$$

$$\sum_{v\in\mathcal{V}}\sum_{p\in\mathcal{P}} f_{v,p,n} \leqslant 1 \qquad \forall n\in\mathcal{N} \tag{36}$$

$$f_{v,p,n} \leqslant \sum_{i\in\mathcal{N}} x^{\text{dest}}_{v,p,i,n} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{37}$$

$$(f_{v,p,n}-1)M_3 \leqslant t^{\text{arr}}_{w,q,n} - t^{\text{arr}}_{v,p,n}\sum_{i\in\mathcal{N}} x^{\text{dest}}_{w,q,i,n}) \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N}, w\in\mathcal{V}, q\in\mathcal{P} \tag{38}$$

$$x^{\text{amount}}_{v,p,n}, t^{\text{arr}}_{v,p,n}, l_{v,p,n}, e_{v,p,n} \in \mathbb{Z}^+ \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{39}$$

$$x^{\text{dest}}_{v,p,i,j} \in \{0,1\} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, i\in\mathcal{N}, j\in\mathcal{N} \tag{40}$$

$$f_{v,p,n} \in \{0,1\} \qquad \forall v\in\mathcal{V}, p\in\mathcal{P}, n\in\mathcal{N} \tag{41}$$

The objective function including the quadratic expected lateness is formulated in Eqs. (22). A penalty for earliness is not included, as this is implicitly penalized through waiting times of vehicles. Constraints (23) present the flow constraints such that the number of movements towards a node is equal to the number of movements departing from that node. Constraints (24) assure that nodes cannot be visited more than once by a single vehicle in the same trip. Nodes can be skipped and also visited multiple times (i.e., split-deliveries). Constraints (25) limit the sum of all deliveries at one node, such that it cannot exceed the demand at that node. Vehicle capacity constraints within one trip are kept by constraints (26). The restrictions that vehicles can only perform trips within the time limit $t^{\max}$ are considered in constraints (27). Constraints (28) ensure that if the location is not visited by vehicle v in trip p that the delivery is 0 and (29) ensure that when a node is visited, the number of goods delivered is at least 1, where M_1 is a sufficiently large positive number. Constraints (30) and (31) represent the soft time window constraints, which check whether deliveries arrive early (i.e., $e_{v,p,n} > 0$) or late (i.e., $l_{v,p,n} > 0$). Constraints (32)–(34) ensure the time flow between and within trips, such that arrivals of a vehicle follow each other chronologically and according to the vehicle flow decisions, travel times, and waiting times due to earliness. Constraints (32) ensure the correct arrival time at the first destination of the first trip of each vehicle. Constraints (33) record the arrival times at the first destination of each additional trip, and constraints (34) keep track of the arrival times between nodes within each trip of each vehicle. Constraints (35) establish the trip order, such that each additional trip p follows after trip $p-1$, with M_2 as a sufficiently large positive number, and this ensures that arrival times within constraints (32)–(34) are respected. Constraints (36) assert that considering all vehicles and all trips, there can at most be one first visit of each node. Constraints (37) ensure that a first visit is only recorded at node n if vehicle v visits node n during trip p. Constraints (38) enforce that among all actual vehicle visits at node n, the visit can only be marked first (i.e., $f_{v,p,n} = 1$) if vehicle v in trip p is the first to arrive. M_3 is a sufficiently large positive number. Constraints (39)–(41) define the integer and binary restrictions on the variables.

Algorithm 3: NearestTW heuristic

initialize arrived vehicle $v \leftarrow \hat{v}_k$, current location $i \leftarrow i_{k,\hat{v}_k}$, shortest travel time $t^{nearest} \leftarrow \infty$, and nearest node $n^{nearest} \leftarrow 0$
for $j \in N \setminus \{i\}$ **do**
  if $0 < d_{k,j}$ **and** $a_{k,j} < t_{v,i,j} < b_{k,j}$ **and** $t_k + t_{v,i,j} + t_{v,j,0} < t^{max}$ **then**
    if $t_{v,i,j} < t^{nearest}$ **then**
      $t^{nearest} \leftarrow t_{v,i,j}$
      $n^{nearest} \leftarrow j$
    end
  end
end
$x^{dest} \leftarrow n^{nearest}$
$x^{amount} \leftarrow \min(d_{k,n^{nearest}}, q_v)$
return x^{dest}, x^{amount}

A.2. Dynamic heuristics

We defined three dynamic heuristics, that take decisions at each decision epoch k, based on the state of the system at that moment (S_k). As the heuristics use real-time information, dynamic decisions are made concerning the arrival time of the vehicle and time windows of locations. Algorithms 3, 4, and 5 present the NearestTW, GreedyTW, and Priority heuristic rules, respectively.

Algorithm 4: GreedyTW heuristic

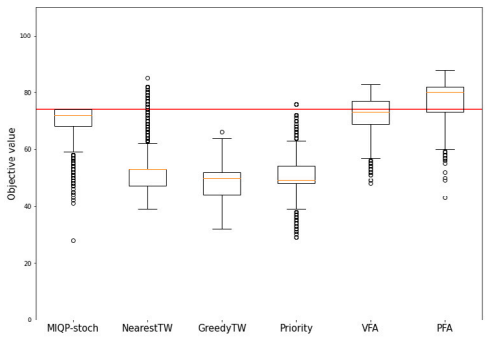
initialize arrived vehicle $v \leftarrow \hat{v}_k$, current location $i \leftarrow i_{k,\hat{v}_k}$, best reward r^{best}, best travel time $t^{best} \leftarrow \infty$, and best node $n^{best} \leftarrow 0$
for $j \in N \setminus \{i\}$ **do**
  if $0 < d_{k,j}$ **and** $a_{k,j} < t_{v,i,j} < b_{k,j}$ **and** $t_k + t_{v,i,j} + t_{v,j,0} < t^{max}$ **then**
    if $\xi_1 \min(d_{k,j}, q_v) + \xi_2 f_{k,j} > r^{best}$ **then**
      $r^{best} \leftarrow \xi_1 \min(d_{k,j}, q_v) + \xi_2 f_{k,j}$
      $t^{best} \leftarrow t_{v,i,j}$
      $n^{best} \leftarrow j$
    else if $\xi_1 \min(d_{k,j}, q_v) + \xi_2 f_{k,j} = r^{best}$ **and** $t_{v,i,j} < t^{nearest}$ **then**
      $t^{best} \leftarrow t_{v,i,j}$
      $n^{best} \leftarrow j$
  end
end
$x^{dest} \leftarrow n^{best}$
$x^{amount} \leftarrow \min(d_{k,n^{best}}, q_v)$
return x^{dest}, x^{amount}

Algorithm 5: Priority heuristic

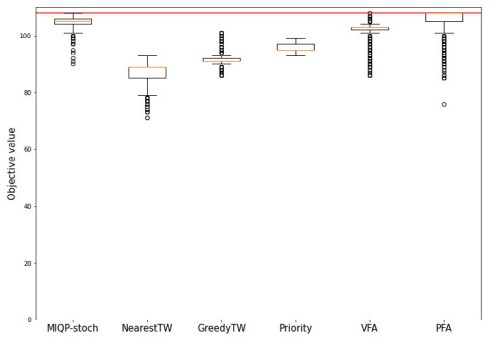
initialize arrived vehicle $v \leftarrow \hat{v}_k$, current location $i \leftarrow i_{k,\hat{v}_k}$, highest priority p^{best}, best travel time $t^{best} \leftarrow \infty$, and best node $n^{best} \leftarrow 0$, earliness penalty $e^{penalty} \leftarrow 0.078125$
for $j \in N \setminus \{i\}$ **do**
  if $0 < d_{k,j}$ **and** $t_k + t_{v,i,j} + t_{v,j,0} < t^{max}$ **then**
    $r \leftarrow \xi_1 \min(d_{k,j}, q_v) + \xi_2 f_{k,j}$
    $r \leftarrow r - P(T_{v,i,j} > b_{k,j}) \cdot [\mathbb{E}(T_{v,i,j} | T_{v,i,j} > b_{k,j}) - b_{k,j}] \cdot \xi_3 \cdot r$
    $r \leftarrow r - P(T_{v,i,j} < a_{k,j}) \cdot (a_{k,j} - \mathbb{E}(T_{v,i,j} | T_{v,i,j} < a_{k,j})) \cdot e^{penalty}$
    if $v =$ truck **then**
      if $r/t_{v,i,j} > p^{best}$ **then**
        $p^{best} = r/t_{v,i,j}$
        $n^{best} = j$
      end
    else
      if $r > p^{best}$ **then**
        $p^{best} = r$
        $t^{best} \leftarrow t_{v,i,j}$
        $n^{best} = j$
      else if $r = p^{best}$ **and** $t_{v,i,j} < t^{nearest}$ **then**
        $t^{best} \leftarrow t_{v,i,j}$
        $n^{best} = j$
      end
    end
  end
end
$x^{dest} \leftarrow n^{best}$
$x^{amount} \leftarrow \min(d_{k,n^{best}}, q_v)$
return x^{dest}, x^{amount}

Appendix B. Additional results

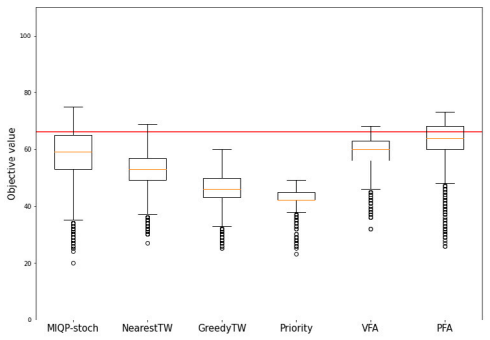
Fig. 8 shows the spread in objective values (i.e., robustness) for each method in the C25 and RC25 cases with trucks only and with trucks and UAVs. Fig. 9 presents the same results for the R100 and RC100 instances.



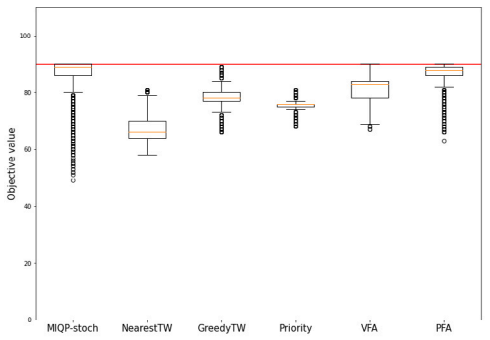
(a) C25 - 2 trucks



(b) C25 - 1 truck, 1 UAV

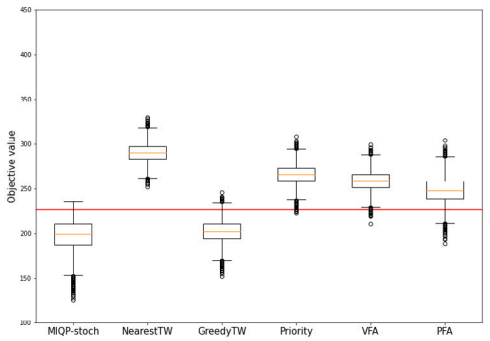


(c) RC25 - 2 trucks

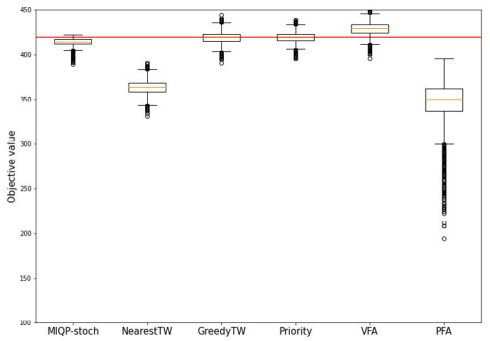


(d) RC25 - 1 truck, 1 UAV

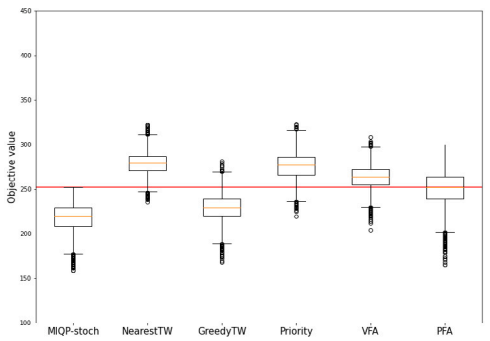
Fig. 8. Objective values for all solution approaches for the C25 and RC25 instances. For reference, a red line has been added corresponding to the objective value of the MIQP-stat solution objectives.



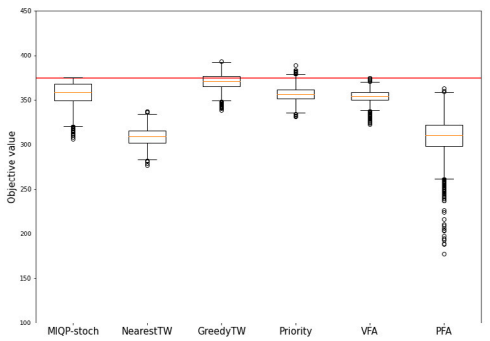
(a) R100 - 8 trucks



(b) R100 - 4 trucks, 4 UAVs



(c) RC100 - 8 trucks



(d) RC100 - 4 trucks, 4 UAVs

Fig. 9. Objective values for all solution approaches for the R100 and RC100 instances. For reference, a red line has been added corresponding to the objective value of the MIQP-stat solution objectives.

References

Adan, I.J.-B.F., Boxma, O.J., Resing, J.A.C., 2001. Queueing models with multiple waiting lines. Queueing Syst. 37, 65–98.

Ahmadi, M., Seifi, A., Tootooni, B., 2015. A humanitarian logistics model for disaster relief operation considering network failure and standard relief time: A case study on san francisco district. Transp. Res. E 75, 145–163.

Akkerman, F., Begnardi, L., Lo Bianco, R., Temizoz, T., Dijkman, R., Iacob, M., Mes, M., Zhang, Y., Van Jaarsveld, W., 2022. Dynaplex. URL http://www.dynaplex.nl/, Visited: 2022-10-05.

Anuar, W.K., Lee, L.S., Pickl, S., Seow, H.-V., 2021. Vehicle routing optimisation in humanitarian operations: A survey on modelling and optimisation approaches. Appl. Sci. 11 (2), 667.

Anuar, W.K., Lee, L.S., Seow, H.-V., Pickl, S., 2022. A multi-depot dynamic vehicle routing problem with stochastic road capacity: An MDP model and dynamic policy for post-decision state rollout algorithm in reinforcement learning. Mathematics 10 (15), 2699.

Balcik, B., Beamon, B.M., 2008. Facility location in humanitarian relief. Int. J. Logist. 11 (2), 101–121.

Bamsey, I., 2021. Dossier: Wings for aid MiniFreighter. Unmanned Syst. Technol. (35), 22–33, URL https://ust-media.com/ust-magazine/UST035/22/, Visited: 2022-07-05.

Beck, C.L., Srikant, R., 2012. Error bounds for constant step-size Q-learning. Systems Control Lett. 61 (12), 1203–1208.

Bellman, R., 1957. A Markovian decision process. J. Math. Mech. 6 (5), 679–684.

Bertsekas, D., 2019. Reinforcement Learning and Optimal Control. Athena Scientific.

Bertsekas, D., Tsitsiklis, J.N., 1996. Neuro-Dynamic Programming. Athena Scientific.

Besiou, M., Van Wassenhove, L.N., 2020. Humanitarian operations: A world of opportunity for relevant and impactful research. Manuf. Serv. Oper. Manag. 22 (1), 135–145.

Bravo, R., Leiras, A., 2015. Literature review of the application of UAVs in humanitarian relief. In: Proceedings of the XXXV Encontro Nacional de Engenharia de Producao, Fortaleza, Brazil. pp. 13–16.

Bruni, M., Khodaparasti, S., Beraldi, P., 2020a. The selective minimum latency problem under travel time variability: An application to post-disaster assessment operations. Omega 92, 102154.

Bruni, M.E., Khodaparasti, S., Nucamendi-Guillén, S., 2020b. The bi-objective minimum latency problem with profit collection and uncertain travel times. In: ICORES. pp. 109–118.

Bruni, M.E., Toan, D.Q., et al., 2021. The multi-vehicle profitable pick up and delivery routing problem with uncertain travel times. Transp. Res. Procedia 52, 509–516.

Cattaruzza, D., Absi, N., Feillet, D., 2016. Vehicle routing problems with multiple trips. 4OR 14 (3), 223–259.

Chang, K.-H., Hsiung, T.-Y., Chang, T.-Y., 2022. Multi-commodity distribution under uncertainty in disaster response phase: Model, solution method, and an empirical study. European J. Oper. Res..

Chauhan, D., Unnikrishnan, A., Figliozzi, M., 2019. Maximum coverage capacitated facility location problem with range constrained drones. Transp. Res. C 99, 1–18.

Chen, W., Mes, M., Schutten, M., Quint, J., 2019. A ride-sharing problem with meeting points and return restrictions. Transp. Sci. 53 (2), 401–426.

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S.G., Bian, L., 2017. Drones for disaster response and relief operations: A continuous approximation model. Int. J. Prod. Econ. 188, 167–184.

Elçi, Ö., Noyan, N., 2018. A chance-constrained two-stage stochastic programming model for humanitarian relief network design. Transp. Res. B 108, 55–83.

Fan, J., Chang, X., Mišić, J., Mišić, V.B., Kang, H., 2022. DHL: Deep reinforcement learning-based approach for emergency supply distribution in humanitarian logistics. Peer-to-Peer Netw. Appl. 15 (5), 2376–2389.

Fereiduni, M., Shahanaghi, K., 2017. A robust optimization model for distribution and evacuation in the disaster response phase. J. Ind. Eng. Int. 13 (1), 117–141.

Fikar, C., Gronalt, M., Hirsch, P., 2016. A decision support system for coordinated disaster relief distribution. Expert Syst. Appl. 57, 104–116.

Gentili, M., Mirchandani, P.B., Agnetis, A., Ghelichi, Z., 2022. Locating platforms and scheduling a fleet of drones for emergency delivery of perishable items. Comput. Ind. Eng. 168, 108057.

Ghannadpour, S.F., Noori, S., Tavakkoli-Moghaddam, R., Ghoseiri, K., 2014. A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application. Appl. Soft Comput. 14, 504–527.

Ghelichi, Z., Gentili, M., Mirchandani, P.B., 2021. Logistics for a fleet of drones for medical item delivery: A case study for Louisville, KY. Comput. Oper. Res. 135, 105443.

Ghelichi, Z., Gentili, M., Mirchandani, P.B., 2022. Drone logistics for uncertain demand of disaster-impacted populations. Transp. Res. C 141, 103735.

Golabi, M., Shavarani, S.M., Izbirak, G., 2017. An edge-based stochastic facility location problem in UAV-supported humanitarian relief logistics: a case study of Tehran earthquake. Nat. Hazards 87 (3), 1545–1565.

Gralla, E., Goentzel, J., 2018. Humanitarian transportation planning: Evaluation of practice-based heuristics and recommendations for improvement. European J. Oper. Res. 269 (2), 436–450.

Hildebrandt, F.D., Thomas, B.W., Ulmer, M.W., 2022. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. Comput. Oper. Res. 106071.

Holguín-Veras, J., Jaller, M., Van Wassenhove, L.N., Pérez, N., Wachtendorf, T., 2012. On the unique features of post-disaster humanitarian logistics. J. Oper. Manage. 30 (7–8), 494–506.

Hoyos, M.C., Morales, R.S., Akhavan-Tabatabaei, R., 2015. OR models with stochastic components in disaster operations management: A literature survey. Comput. Ind. Eng. 82, 183–197.

Hu, S., Han, C., Dong, Z.S., Meng, L., 2019. A multi-stage stochastic programming model for relief distribution considering the state of road network. Transp. Res. B 123, 64–87.

International Organization for Migration, 2019. Indonesia displacement data sulawesi earthquake site assessment [IOM DTM]. URL https://data.humdata.org/dataset/indonesia-displacement-data-sulawesi-earthquake-site-assessment-iom-dtm, Visited: 2022-09-03.

Karunakaran, D., Mei, Y., Zhang, M., 2019. Multitasking genetic programming for stochastic team orienteering problem with time windows. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, pp. 1598–1605.

Kim, G., Ong, Y.S., Cheong, T., Tan, P.S., 2016. Solving the dynamic vehicle routing problem under traffic congestion. IEEE Trans. Intell. Transp. Syst. 17 (8), 2367–2380.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Klaine, P.V., Nadas, J.P., Souza, R.D., Imran, M.A., 2018. Distributed drone base station positioning for emergency cellular networks using reinforcement learning. Cogn. Comput. 10 (5), 790–804.

Köster, F., Ulmer, M.W., Mattfeld, D.C., 2015. Cooperative traffic control management for city logistic routing. Transp. Res. Procedia 10, 673–682.

Köster, F., Ulmer, M.W., Mattfeld, D.C., Hasle, G., 2018. Anticipating emission-sensitive traffic management strategies for dynamic delivery routing. Transp. Res. D 62, 345–361.

Laporte, G., Louveaux, F., Mercure, H., 1992. The vehicle routing problem with stochastic travel times. Transp. Sci. 26 (3), 161–170.

Liberatore, F., Pizarro, C., Blas, C., Ortuño, M., Vitoriano, B., 2013. Uncertainty in humanitarian logistics for disaster management. a review. In: Decision Aid Models for Disaster Management and Emergencies. Springer, pp. 45–74.

Lin, L.-J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. Mach. Learn. 8 (3), 293–321.
Liu, Y., 2019. An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. Comput. Oper. Res. 111, 1–20.
Liu, Z., Li, X., Khojandi, A., 2022. The flying sidekick traveling salesman problem with stochastic travel time: A reinforcement learning approach. Transp. Res. E 164, 102816.
Lorini, S., Potvin, J.-Y., Zufferey, N., 2011. Online vehicle routing and scheduling with dynamic travel times. Comput. Oper. Res. 38 (7), 1086–1090.
Macias, J.E., Angeloudis, P., Ochieng, W., 2020. Optimal hub selection for rapid medical deliveries using unmanned aerial vehicles. Transp. Res. C 110, 56–80.
Mao, C., Shen, Z., 2018. A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. Transp. Res. C 93, 179–197.
Mills, A.F., Argon, N.T., Ziya, S., 2018. Dynamic distribution of patients to medical facilities in the aftermath of a disaster. Oper. Res. 66 (3), 716–732.
Mohammadi, S., Darestani, S.A., Vahdani, B., Alinezhad, A., 2020. A robust neutrosophic fuzzy-based approach to integrate reliable facility location and routing decisions for disaster relief under fairness and aftershocks concerns. Comput. Ind. Eng. 148, 106734.
Mondal, A., Roy, S.K., 2021. Multi-objective sustainable opened-and closed-loop supply chain under mixed uncertainty during COVID-19 pandemic situation. Comput. Ind. Eng. 159, 107453.
Munos, R., 2005. Error bounds for approximate value iteration. In: Proceedings of the National Conference on Artificial Intelligence. Vol. 20, (2), Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1006.
Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transp. Res. C 54, 86–109.
Nadi, A., Edrisi, A., 2016. A reinforcement learning approach for evaluation of real-time disaster relief demand and network condition. Int. J. Econ. Manag. Eng. 11 (1), 5–10.
Nadi, A., Edrisi, A., 2017. Adaptive multi-agent relief assessment and emergency response. Int. J. Disaster Risk Reduct. 24, 12–23.
Nedjati, A., Vizvari, B., Izbirak, G., 2016. Post-earthquake response by small UAV helicopters. Nat. Hazards 80 (3), 1669–1688.
Nezhadroshan, A.M., Fathollahi-Fard, A.M., Hajiaghaei-Keshteli, M., 2021. A scenario-based possibilistic-stochastic programming approach to address resilient humanitarian logistics considering travel time and resilience levels of facilities. Int. J. Syst. Sci.: Oper. Logist. 8 (4), 321–347.
Oruc, B.E., Kara, B.Y., 2018. Post-disaster assessment routing problem. Transp. Res. B 116, 76–102.
Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L., 2013. A review of dynamic vehicle routing problems. European J. Oper. Res. 225 (1), 1–11.
Potvin, J.-Y., Xu, Y., Benyahia, I., 2006. Vehicle routing and scheduling with dynamic travel times. Comput. Oper. Res. 33 (4), 1129–1137.
Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, second ed. John Wiley & Sons.
Powell, W.B., 2019. A unified framework for stochastic optimization. European J. Oper. Res. 275 (3), 795–821.
Powell, W.B., 2022. Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions. Wiley.
Powell, W.B., Meisel, S., 2016. Tutorial on stochastic optimization in energy — Part II: An energy storage illustration. IEEE Trans. Power Syst. 31 (2), 1468–1475.
Pureza, V., Laporte, G., 2008. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. INFOR: Inf. Syst. Oper. Res. 46 (3), 165–175.
Puterman, M.L., 1990. Markov decision processes. In: Handbooks in Operations Research and Management Science. Vol. 2, Elsevier, pp. 331–434.
Rabta, B., Wankmüller, C., Reiner, G., 2018. A drone fleet model for last-mile distribution in disaster relief operations. Int. J. Disaster Risk Reduct. 28, 107–112.
Raghu, T., Rajendran, C., 1993. An efficient dynamic dispatching rule for scheduling in a job shop. Int. J. Prod. Econ. 32 (3), 301–313.
Rejeb, A., Rejeb, K., Simske, S., Treiblmaier, H., 2021. Humanitarian drones: A review and research agenda. Int. Things 16, 100434.
Rennemo, S.J., Rø, K.F., Hvattum, L.M., Tirado, G., 2014. A three-stage stochastic facility routing model for disaster response planning. Transp. Res. E 62, 116–135.
Ritzinger, U., Puchinger, J., Hartl, R.F., 2016. A survey on dynamic and stochastic vehicle routing problems. Int. J. Prod. Res. 54 (1), 215–231.
Schilde, M., Doerner, K.F., Hartl, R.F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. European J. Oper. Res. 238 (1), 18–30.
Shavarani, S.M., 2019. Multi-level facility location-allocation problem for post-disaster humanitarian relief distribution: a case study. J. Humanitarian Logist. Supply Chain Manag. 9 (1), 70–81.
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362 (6419), 1140–1144.
Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 (2), 254–265.
Stumpf, J., Besiou, M., Wakolbinger, T., 2022. Assessing the value of supply chain management in the humanitarian context–an evidence-based research approach. J. Humanitarian Logist. Supply Chain Manag. (ahead-of-print).
Su, Z.-P., Jiang, J.-G., Liang, C.-Y., Zhang, G.-F., 2011. Path selection in disaster response management based on Q-learning. Int. J. Autom. Comput. 8 (1), 100–106.
Taniguchi, E., Shimamoto, H., 2004. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. Transp. Res. C 12 (3–4), 235–250.
Taş, D., Dellaert, N., van Woensel, T., De Kok, T., 2014. The time-dependent vehicle routing problem with soft time windows and stochastic travel times. Transp. Res. C 48, 66–83.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Thomas, B.W., 2020. On modeling stochastic dynamic vehicle routing problems. EURO J. Transp. Logist. 9 (2), 100008.
Ulmer, M.W., Thomas, B.W., 2020. Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. European J. Oper. Res. 285 (1), 183–195.
Van Jaarsveld, W., 2020. Deep controlled learning of dynamic policies with an application to lost-sales inventory control. CoRR.
Van Wassenhove, L.N., 2006. Humanitarian aid logistics: supply chain management in high gear. J. Oper. Res. Soc. 57 (5), 475–489.
Van Woensel, T., Kerbache, L., Peremans, H., Vandaele, N., 2008. Vehicle routing with dynamic travel times: A queueing approach. European J. Oper. Res. 186 (3), 990–1007.
Van Wynsberghe, A., Comes, T., 2020. Drones in humanitarian contexts, robot ethics, and the human–robot interaction. Ethics Inf. Technol. 22 (1), 43–53.
Wang, G., Giannakis, G.B., 2020. Finite-time error bounds for biased stochastic approximation with applications to Q-learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 3015–3024.
Xiang, Z., Chu, C., Chen, H., 2008. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European J. Oper. Res. 185 (2), 534–551.
Yan, Y., Chow, A.H., Ho, C.P., Kuo, Y.-H., Wu, Q., Ying, C., 2022. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. Transp. Res. E 162, 102712.
Yang, Z., Nguyen, L., Zhu, J., Pan, Z., Li, J., Jin, F., 2020. Coordinating disaster emergency response with heuristic reinforcement learning. In: 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, pp. 565–572.
Yu, G., Yang, Y., 2019. Dynamic routing with real-time traffic information. Oper. Res. 19, 1033–1058.
Yu, L., Zhang, C., Jiang, J., Yang, H., Shang, H., 2021. Reinforcement learning approach for resource allocation in humanitarian logistics. Expert Syst. Appl. 173, 114663.
Zhang, J., Liu, Y., Yu, G., Shen, Z.-J., 2021. Robustifying humanitarian relief systems against travel time uncertainty. Nav. Res. Logist. 68 (7), 871–885.
Zhou, H., Wang, X., Umehira, M., Chen, X., Wu, C., Ji, Y., 2021. Wireless access control in edge-aided disaster response: A deep reinforcement learning-based approach. IEEE Access 9, 46600–46611.