mathematics

MDPI

Article

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

Wadi Khalid Anuar [1,2], Lai Soon Lee [2,3,*], Hsin-Vonn Seow [4] and Stefan Pickl [5]

[1] Department of Logistics and Transportation, School of Technology Management and Logistics, Universiti Utara Malaysia, Sintok 06010, Kedah, Malaysia; wadikhalidanuar@gmail.com
[2] Laboratory of Computational Statistics and Operations Research, Institute for Mathematical Research, Universiti Putra Malaysia, Serdang 43400, Selangor, Malaysia
[3] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, Serdang 43400, Selangor, Malaysia
[4] Faculty of Arts and Social Sciences, Nottingham University Business School, University of Nottingham Malaysia Campus, Semenyih 43500, Selangor, Malaysia; hsin-vonn.seow@nottingham.edu.my
[5] Fakultät für Informatik, Universität der Bundeswehr München, 85577 Neubiberg, Germany; stefan.pickl@unibw.de
* Correspondence: lls@upm.edu.my

Abstract: In the event of a disaster, the road network is often compromised in terms of its capacity and usability conditions. This is a challenge for humanitarian operations in the context of delivering critical medical supplies. To optimise vehicle routing for such a problem, a Multi-Depot Dynamic Vehicle-Routing Problem with Stochastic Road Capacity (MDDVRPSRC) is formulated as a Markov Decision Processes (MDP) model. An Approximate Dynamic Programming (ADP) solution method is adopted where the Post-Decision State Rollout Algorithm (PDS-RA) is applied as the lookahead approach. To perform the rollout effectively for the problem, the PDS-RA is executed for all vehicles assigned for the problem. Then, at the end, a decision is made by the agent. Five types of constructive base heuristics are proposed for the PDS-RA. First, the Teach Base Insertion Heuristic (TBIH-1) is proposed to study the partial random construction approach for the non-obvious decision. The heuristic is extended by proposing TBIH-2 and TBIH-3 to show how Sequential Insertion Heuristic (SIH) (I1) as well as Clarke and Wright (CW) could be executed, respectively, in a dynamic setting as a modification to the TBIH-1. Additionally, another two heuristics: TBIH-4 and TBIH-5 (TBIH-1 with the addition of Dynamic Lookahead SIH (DLASIH) and Dynamic Lookahead CW (DLACW) respectively) are proposed to improve the on-the-go constructed decision rule (dynamic policy on the go) in the lookahead simulations. The results obtained are compared with the matheuristic approach from previous work based on PDS-RA.

Keywords: reinforcement learning; Markov decision processes; approximate dynamic programming; rollout algorithm; constructive base heuristic; vehicle routing problem.

MSC: 90C40; 90B15; 90C59

Citation: Anuar, W.K.; Lee, L.S.; Seow, H.-V.; Pickl, S. 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* **2022**, *10*, 2699. https://doi.org/10.3390/math10152699

Academic Editors: Francois Rivest and Abdellah Chehri

Received: 2 June 2022
Accepted: 26 July 2022
Published: 30 July 2022

1. Introduction

Recent events have shown that the occurrence of a disaster continues to claim many lives despite the growing number of relief organisations to support and help the victims throughout the world. In the case of the 2015 Nepal earthquake, for example, nearly 9000 lives were lost, and 23,000 people were injured [1]. In the event, critical medical supplies and health personnel were far from lacking, given aid rushed into Nepal as soon

as the news went out. The relief supplies received worldwide were so large in volume that the Kathmandu airport was overwhelmed with the sudden increase in air traffic [2]. However, these life-changing resources could not be distributed accordingly and in a timely manner given the lack of coordination between the local and international relief aid providers. This led to inefficiencies in executing relief operations [1,2]. The fact that Nepal is a landlocked country without any sea access also exacerbated the logistical challenges further, as seen from the bottleneck problem in Kathmandu airport [3]. The hilly topography of Nepal is prone to landslides [1,2,4], while continuous aftershocks damaged the road infrastructure, such as the Pasang Lhamu and Araniko highways [5], which compromised the road network to the disaster zone [4,6]. Additionally, the sudden onset of the disaster meant that there was limited availability of vehicles and little preparation for emergency logistics operations [4]. The viability of the long-term humanitarian operations with the available vehicles, with regards to safety and logistics and asset management, were also in question since the road network was compromised [4]. This hindered efficient relief operations in terms of transport and delivery [4,7].

Meanwhile, urgent local medical supplies were rapidly diminishing as both field and local hospitals were overrun by victims seeking immediate treatment [2,8]. If the relief aid and supplies that were laying dormant at the Tribhuvan Airport could have been channelled through effectively, it would certainly have helped alleviate the problem of the urgent need for medical supplies and treatment.

In terms of disaster management preparedness, this case study serves to highlight the critical role of transportation in the event of a disaster. Transportation service is a crucial element when it comes to humanitarian logistics operation, in particular the in-country transportation for delivery of relief supplies [4]. From the 2015 Nepal earthquake event, some observations have been made with regards to ensuring efficient delivery of goods through vehicle routing. The study in [4] pointed out that the geographical topology and mountainous landscapes of Nepal as well as the second earthquake tremors and weather conditions during the event heavily impacted the delivery speed, especially involving the last mile of the delivery. In the disaster event, the road condition and capacity were compromised by major accidents, causing traffic density and loss of cargo [4]. Furthermore, there was an overwhelming demand for the limited number of trucks in terms of capacity and availability due to critical delay and backlogs. This increased the price of transport vehicle procurement to as high as 40%. Additionally, the lack of a decision support system (DSS) to monitor and track transport vehicles, coupled with untrained drivers, also led to a serious shortage of reliable transportation. Landslides, in particular, limited the routing to certain areas. In addition, the risk of accidents due to a unfamiliar route was significant and not helped by landslides which are sensitive to weather conditions. As such, intelligent routing and communication access is pivotal for this particular vehicle routing problem.

The proposed Multi-Depot Dynamic Vehicle-Routing Problem with Stochastic Road Capacity (MDDVRPSRC) model addresses such delivery problems in the setting of relief humanitarian operations by incorporating the aforementioned challenges. The problem of a bottleneck could be solved if "mini" airports were erected at strategic locations (multi-depot). The problem of limited transport vehicles in terms of availability and capacity could be addressed at some length through transport vehicles performing split deliveries which is known to be a cheaper alternative by almost 50% [9]. Furthermore, these vehicles could also perform multi-trip deliveries. Communication among Logistics Service Providers (LSPs) and their local contacts helps in updating the road network conditions which may hint towards dynamic problem modelling, with information updates at regular intervals. The uncertainty which is the pivotal aspect of efficient delivery operations should be addressed in terms of dynamic and stochastic settings of road capacity and conditions within the road network [4,7].

The Markov Decision Processes (MDP) modelling framework offers a natural way to address these dynamic and stochastic aspects of the problem. However, incorporating these aspects leads to an exponential growth of state, action and outcome spaces which is needed when solving such a real and complex problem. To deal with the curse of dimensionality [10], the Approximate Dynamic Programming (ADP) approach is commonly applied to solve the problem from the Machine Learning (ML) (Reinforcement Learning, RL) perspective. To address the large outcome space, the Post-Decision State (PDS) is applied as an extended version of the basic MDP modelling framework [11].

The ADP approach is usually applied to approximate the value of the next state s_{k+1} or the value of the action a_k when solving the Bellman equation [12]. In ADP, the lookahead approach is suitable when dealing with the large action and outcome space that constitutes part of the curse of dimensionality. In this paper, the known Post-Decision State Rollout Algorithm (PDS-RA) [13] is applied as part of the lookahead approach in the ADP.

A base heuristic in the PDS-RA is taken as the guiding policy for the decision rule applied as the state transitions within the lookahead horizon. In modelling the VRP through the MDP framework, the decision rule is generally the assignment of vehicles when the state transitions. In the rollout, the transition occurs in the lookahead horizon beginning from the potential next state s_{k+1} to the lookahead end state s_K. In other words, the base decision rule is a route computed for all vehicles to navigate within the future lookahead horizon. In the case of MDDVRPSRC, however, assigning vehicles based on a computed route, which is computed once, becomes problematic. This is due to the stochastic road capacity which may render the computed route unusable in the next lookahead update. One way to navigate around this is to have the route build dynamically on the go by a simple constructive heuristic known as a base heuristic.

From the decision rule or route obtained through this approach at each iteration, only the first assignments of the constructed route are applied while the rest are ignored. This method is feasible and practical due to the less expansive computation performed by the simple constructive heuristic. Here, the Teach Base Insertion heuristic (TBIH-1) is proposed to balance the random exploration and to guide the exploitation by dictating obvious assignments. Extending from this, the authors apply two known constructive heuristics: Sequential Insertion Heuristic SIH(I1) and Clarke and Wright (CW), in a dynamic setting and embed them into the TBIH-1(as proposed TBIH-2 and TBIH-3). To the best of our knowledge, no paper has shown how these constructive heuristics can be executed, as proposed, in a PDS-RA setting. We further derived from these two heuristics: TBIH-4 and TBIH-5 that seek, within their algorithm, promising vehicle assignments by looking up to two steps ahead. To deal further with the large action space, most research provides some mechanism to segregate or cluster vehicles with a specific set of customers to serve. Due to the stochastic road capacity, such mechanism is difficult to adopt. Thus, further approximation is made when computing the optimal action $a_k^\star$ with regards to executing the PDS-RA. Through this proposed method, each PDS-RA is executed for each vehicle for every potential assignment that can be given to that vehicle.

The contributions made in this paper are as follows: first, the novel MDDVRPSRC is proposed in a disaster event setting based on the modelling framework of MDP. Next, TBIH-1 heuristic is proposed in this work along with four extended variants, namely TBIH-2, TBIH-3, TBIH-4, and TBIH-5. By doing so, it is shown how SIH(I1) and CW can be applied in the dynamic setting of route-based MDP for PDS-RA [14]. The authors also show how these two can be extended by looking up to two steps ahead. Finally, it is also shown and validated how the near optimal decision can be approximated further by disintegrating the collective assignment decisions to an individual near optimal decision. To the best of our knowledge, both the MDDVRPSRC MDP model and the five base heuristics applied in the PDS-RA algorithm are novel and have not yet been proposed.

This work is the extension of research published by [15], where the damage determination of the roads within the road network is referred from. Furthermore, the Poisson stochastic distribution of a stochastic road capacity is also referred from. In this paper, the dynamic and stochastic MDDVRPSRC MDP model is presented to complement the earlier research in [15] that modelled the Deterministic Multi-Depot VRP with Road Capacity (D-MDVRPRC) as well as the two-stage Stochastic Integer Liner Programming (SILP) model of a Multi-Depot VRP with Stochastic Road Capacity (MDVRPSRC-2S). To solve the MDP Model presented in this paper via the PDS-RA algorithm, the five base heuristics are proposed as an alternative to the "cluster first, route second" approach that cannot be applied. Meanwhile, to benchmark these heuristics, the matheuristic rollout presented in [15] is applied where tractable.

This paper is organised as follows: Section 2 describes the literature review focusing on the proposed model and base heuristics. Section 3 describes the problem of MDDVRPSRC where some elements of the models have been referenced from the authors' previous work. The known PDS-RA approach is briefly described in Section 4 as well as how the optimal decision is approximated through the proposed mechanism. Additionally, variants of the proposed base heuristics are presented here. Section 5 presents the computational results. while Section 6 synthesises the findings. Finally, Section 7 concludes the paper.

2. Literature Review

An extensive synthesis of the literature on works adapting VRP for humanitarian operations can be found in [16,17]. In [17], numerous papers within the last decade (as of 2020) have been reviewed in terms of the application of VRPs for three selected humanitarian operations. Various modelling aspects, such as dynamic and stochastic problem, multi-disaster phase, multi-objectives, multi-trips, multi-depots, split delivery and more, which are relevant to the model proposed in this work, are elaborated. Furthermore, the solution approaches applied are also discussed in detail, especially the challenges when dealing with stochastic and dynamic VRPs. Ref. [15] extends the findings by discussing some papers applying VRP outside the field of humanitarian operation settings but are still relevant to the model that was proposed. Meanwhile, the RL adaptation in solving Supply Chain Management (SCM) is discussed in [18]. Additionally, the adoption of RL in VRP and TSP is discussed in [19].

2.1. Stochastic Vehicle Routing Problem for Humanitarian Operations

The survey in [20] discussed the recent dynamic VRP for various applications from 2015 to 2021. From the analysis of their work, it could be observed that the research focusing on dynamic problems usually also address stochastic problems. The opposite, however, may not necessarily be true. As such, some discussion on research works regarding stochastic VRP for humanitarian operations is warranted to complement those discussed in [17]. For example, the recent work of [21] addressed the Inventory Routing Problem (IRP) with an uncertain traffic network. Here the distribution of essential multi-commodities in the chaotic post-disaster phase among relief shelters and distribution centers is modelled through the network flow model. The uncertain traffic network is due to the magnitude of the earthquake's attributes, such as the earthquake's magnitude and the time of its occurrence. Such attributes also affect the vehicle speeds when making split deliveries. Apart from the optimized routing decision, the efficiency of the commodities distribution is further improved through the optimized inventory decision. Both are computed via the simulation optimization technique where the Sample Averaging Approximation (SAA) method is applied.

On the other hand, Ref. [22] presented a two-stage Location Routing Problem (LRP) of distributing first aid relief materials post-earthquake disaster where the complex demand uncertainty is addressed. Here the mixture of uncertain demand from the perspective of randomness (probabilistic theory) and fuzziness (possibility theory) due to the merging of subjective and objective data forms the hybrid demand uncertainties. A scenario-

based stochastic demand over a specified interval per scenario is considered for stochastic probabilistic programming due to the strong relationship between events/scenarios such as post-earthquakes and aftershocks and the demands' uncertainty. As such, the demand parameter is considered a fuzzy random variable. In the two-stage robust programming model, the decision for locating a warehouse among existing warehouses is first determined, while the routing and distribution decision is computed in the second stage. The objective is to minimize the cost of warehouse location and the penalty induced by unsatisfied demand. Here the equivalent crisp model is represented by the Basic Possibilistic Chance Constraint Programming (BPCCP) model and later modified as the two-stage robust programming model to overcome the drawback of BPCCP. A small-scale instance problem based on the actual case study of Hamadan province of Iran, a location prone to earthquakes, is applied to verify the model proposed. The solution obtained using CPLEX indicates that the demand satisfaction level is significantly higher than in conventional scenario-based stochastic programming.

Meanwhile, the work proposed in [23] focused on the redistribution of food to charitable agencies, such as homeless shelters and soup kitchens, by picking up the resource (food) from donors such as grocery stores and restaurants. Here the objective is to assure fairness in distributing the donated foods while also considering the waste implicitly through the constraints introduced. The demand from charitable agencies and the resource (donation) are stochastic in this problem. Additionally, equity is the rate between allocated amounts to respective agencies over the total demands. Some assumptions are made, such as unlimited vehicle capacities and that all donors must be visited before distribution is performed. A decomposition strategy in the form of a heuristic is applied to solve the problem where the recipients and donors are clustered first. Then the route is computed for respective clusters, and resources are allocated for each recipient.

Another multi-objective problem is addressed in [24] to deliver both non-perishable and perishable items to demand points considering uncertainty, such as the location and number of relief centers that should be established at the demand points as well as the delivery means of the relief item post-disasters. A Mixed-Integer Non-Linear programming (MINLP) model is formulated to minimise the total distance travelled, the maximum travelled distance between relief centers and demand points, and the total cost associated with acquiring the relief items and vehicles utilized as well as the inventory cost. This model is solved by GAMS software for small-scale instances. Meanwhile, the larger instances are solved by a Grasshopper Optimization Algorithm (GOA) metaheuristic.

Ref. [25] addressed another multi-objective problem involving the COVID-19 pandemic. In this problem, multi-period collection and delivery of multi-products in a single open and close loop Supply Chain (SC) is modelled through the formulation of transportation problems and the Pick and Delivery VRP (PDVRP). Here the open and close supply chain system involving reusable and non-reusable products is transferred from hybrid depots that produce and recycle the products through a forward and reverse flow. The transfer collection centers located in the affected COVID-19 areas acted as the intermediary between the depots and the hospitals. Heterogeneous vehicles traverse the forward and reverse flow via PDVRP from the transfer collection centers to the hospitals when delivering products through split delivery while receiving the old product for reproduction. Various uncertain parameters are defined involving vehicle cost, production cost, the demand from the hospitals, and the returned products when computing for multiple decision variables focusing on the number and routing of the vehicles, production and return of products, shipping, and inventory of products. Finally, a robust optimization approach is applied where the Tchebycheff method is adopted to solve the complex problem.

A complex problem of relief distribution within the humanitarian logistics network and victim evacuation is addressed by [26]. This problem is based on the Facility Location Problem (FLP) and VRP, where the uncertain demand, transportation time, miscellaneous cost, injured victims, and facility capacity are considered. Moreover, the formulated problem involves stakeholders, such as the suppliers of relief materials (e.g., charitable

organizations), the distribution centers, the emergency centers that distribute the aids, distribution units where the evacuated victims are located, and hospitals to which they are sent to for further treatment. Thus, the decisions are based on locating and distributing relief materials and allocating evacuated victims assigned to respective routed vehicles. Multi-objectives are also addressed where the total humanitarian logistics cost, the total time of relief operations, and the variation between the lower and upper bound of the transportation cost of the distribution centers are minimized. The uncertainties in the form of inconsistencies and unclear and inaccurate information are captured through the neutrosophic fuzzy set. These uncertainties are prioritized and dealt with, respectively, through the neutrosophic set and the robust optimisation, where the latter deals with the uncertainties associated with worst-case planning involving victims, facility capacity, and relief supply.

The work by [27] similarly addressed the LRP through the Conditional Value at Risk with Regret (CVaR-R) bi-objective mixed nonlinear programming model in dealing with relief distribution post-disaster. In this problem, the optimal decisions include selecting distribution centers to be made operational among all existing distribution centers, the number of vehicles that should be assigned, the assignment of vehicles to respective demand points, and the allocation of relief aids delivered to each of these demand points. The concept of regret when making these decisions due to inaccurate expectations of demands allows for a novel chance constraint programming approach introduced through the CVaR-R measure. The regret value for each objective to: (1) minimise the total waiting time of demand points and (2) minimise the total system cost is measured by defining possible demand scenarios with respective probability. For each demand scenario, the difference between the ideal objective values (from the deterministic model) and the objective values computed given the demands in each scenario is determined, from which the worst-case scenarios are identified and applied to compute the CVaR-R. To solve the problem model, the Nash Bargaining Solution (NBS) approach is applied with the help of a hybrid Genetic Algorithm (GA) when solving the single LRP version of the problem (via the GA) as well as determining the Pareto frontier (via the Non-Dominated Sorting GA Algorithm II (NSGA-II)) used to compute the NBS.

Finally, Ref. [28] also addressed the risk decision factor regarding the relief distribution with uncertain travel time. This problem is formulated based on the multi-level network via a mixed integer programming model to minimise the total arrival time of vehicles delivering relief materials to only selected demand points instead of satisfying demand for all demand points. Furthermore, the near-optimal delivery route computed ensures that a particular service level is reached throughout the operation by formulating the associated constraints. By introducing the risk-averse approach, the objective function is adapted by incorporating the standard deviation term representing the risk of the decision regarding uncertain time travel. Meanwhile, the non-additional term is the expected total arrival that needs to be minimized. Both are weighted to balance the importance of risk to the decision maker when computing for the near-optimal decision. Here, the Variable Neighbourhood Search (VNS) is employed to solve the problem based on the data obtained from the Haiti earthquake case study in 2010.

2.2. Markov Decision Processes Model for Humanitarian Operations

In this paper, the current trends of VRP in the scope of MDP modelling are discussed. It could be derived from [17] that MDP modelling for VRPs in the setting of humanitarian operations is scarce. This is supported by the finding that only [29] addressed the problem in humanitarian operations. Ref. [29] looked at multiple humanitarian operations, such as delivery and search and rescue, by incorporating the Relief Assessment Team (RAT) and the Emergency Relief Team (ERT), using the Decision-Making Agent (DMA) to coordinate the former two. The problem is modelled as an MDP with multiple random parameters, such as the demand and stability of the transport link. Here, the RAT is tasked to assess affected areas and dynamically report the demand situation for each zone. ERT, on the

other hand, is tasked to serve the zones assigned by the DMA. Meanwhile, decisions are composed by assigning various combinations of aid organisations to a specified affected zone, as well as routing decisions for both the ERT and RAT. Finally, Q-learning is applied to solve the problem involving a maximum of 7 RAT and 10 ERT.

Other than papers reviewed by [17], there are works such as [30] which plan evacuation routing for the last minute disaster preparedness phase. In this problem, residents are evacuated prior to disaster occurrence via buses that pick up stochastic resident evacuees at bus stops. The problem is modelled as a single trip operation with a homogeneous fleet of vehicles within a finite horizon. Here, the action or decision is the assignment of the next pick-up point for each of the vehicles and the number of evacuees taken, suggesting a split delivery operation. The small instance of the problem is solved exactly using structured value iteration. Meanwhile, dynamic re-routing is applied for a large-scale problem through a reduced network flow MIP and Robust MIP (RMIP) model. Beyond these aforementioned works, related works on modelling a VRP through the framework of the MDP for humanitarian operations are sparse.

Instead, the MDP application is used to address humanitarian operations that revolve around the coverage problem, the allocation problem, the path planning, and the scheduling problem. Ref. [31] computed the evacuation routes during a disaster by modelling the problem as an MDP model. However, the work cannot be regarded as a VRP as the target application is not specified in terms that would constitute a VRP, such as the vehicles' availability or their capacity. Similarly, Ref. [32] used the MDP model to address the problem of clearing the debris from blocked edges or roads in an optimal assignment with uncertain clearance resources. In the post-disaster event, the optimal decision was computed considering the delivery of aid or service for demand nodes. Then we have [33] who addressed the problem of congestion in terms of hospital facilities as well as limited ambulances to rescue patients in the aftermath of a disaster. Considering the stochastic treatment time and transportation availability, the decision for such planning was to allocate ambulances to affected patients and to choose which medical facility the ambulance should be headed for. Two types of vehicles are considered: a dedicated ambulance for a location and another ambulance with flexibility in terms of the location. The latter might suggest split delivery. However, neither capacity nor comprehensive routing were considered in this problem. Dynamic patient treatment times were updated, and the problem was solved based on proposed heuristics that applied a myopic approach with policy improvement.

2.3. Markov Decision Processes Model for Industrial Problems and the Application of Approximate Dynamic Programming

Apart from the application for humanitarian operations, the MDP modelling framework has also been adopted for industrial problems since 2000. Interestingly, most of the early works are dedicated to solve the single VRP with stochastic demand such as in [34–37]. The study in [38] is among the first to look into the theoretical aspect of the pick-up and delivery of a single vehicle-routing problem with stochastic request using MDP modelling. The focus on the multi-vehicle problem is seen later in [39–41] for a VRP with stochastic demands. On the other hand, Ref. [42] deviated from stochastic demands by addressing the problem with stochastic customers or stochastic requests. In this type of application, various different and diverse solution strategies were adopted. This contrasted with the problem with stochastic demands which mainly applied the lookahead approach such as the PDS-RA.

Meanwhile, in [14,43] the route-based MDP was introduced as opposed to the conventional formulation of an MDP addressing a VRP while incorporating a dynamic route plan at every decision epoch. This framework was applied in [44] to solve the problem of maximising the number of services within the same period for stochastic service requests using the Value Function Approximation (VFA). The problem was to decide which new stochastic request should be accepted and which should be postponed to the next operation period. Apart from the fixed working time, multi-periods would be considered as multi-trip

operations. VFA was applied as one of the ADP solution approaches allowing for online computation while dealing with the decision and outcome spaces. Through the VFA, the state space was segregated based on common features. The resulting MDP model led to a huge number of decision points which was dealt with by introducing a classification of multi-period operations. The application of VFA as shown here is commonly known to ignore some details of the state due to the aggregation mechanism within. The same authors in [45] alleviated this problem by proposing a hybrid of two ADP approaches: the VFA and the Rollout Algorithm (RA) to address the stochastic request for a single vehicle-routing problem. The authors later addressed the same problem in [46] with a different hybrid mechanism of VFA and RA by having the second part of the simulation horizon driven by a base policy dictated by the VFA. This method was effective in increasing the solution quality while reducing the computation time. A single vehicle-routing problem was also addressed by [47] with regards to taxi routing when searching for a passenger. In this MDP model, the transition probability is derived from the taxi–passenger matching probability on a link. Here, an enhanced value iteration was applied to solve the problem by reformulating Bellman's Equation into a series of matrix operations.

2.4. Approximate Dynamic Programming in Machine Learning

As can be seen from the example mentioned above, aside for [47], ADP is commonly adopted as a solution approach for problems formulated in MDP. The ADP emerged as a means to solve real and practical MDP problem models. This is due to the complexity of the MDP model that increases exponentially due to the explosion of state, outcome, and action spaces [48,49]. Such problem renders the exact solution to be prohibitive as shown in [50,51]. Afterwards, Ref. [52] coined the term ADP which is otherwise known as RL or neuro–dynamic programming. The interest in ADP sparked around the mid–1990s when it was extensively written on by [53,54]; although the ideas and concepts could be found dating back to the 1950s. For instance, Ref. [55] described the concept of approximating the value of a position in a chess game which is based on the state of the board. He likened the concept to an experienced player evaluating a move roughly but not based on all possible scenarios. Later, Ref. [12], when introducing dynamic programming, hinted at the idea with the mention of value space and approximation. Then [56] not only applied the idea of [55] in evaluating a position move, but also showcased the practical use of the lookahead approach while approximating the value of a move in the game of checkers. At the same time, he considered the possibility of remembering all the positions and moves, much like the concept of a dynamic lookup table. The work was further improved in [57] with regards to a tree search lookahead with an improved alpha–beta pruning scheme based on the memorisation of a board position, known as the book-learning procedure. The basic ideas from the aforementioned works were explored further resulting in pivotal works of [58–60] which formed what is known as the modern era of ADP [53]. Following that, Refs. [61–63] in his experiments with the game Backgammon afterwards demonstrated the practical application of ADP to solve real and complex problems.

2.5. Rollout Algorithm and Post-Decision State Rollout Algorithm in Approximate Dynamic Programming

The authors take advantage of the well-known operations of the VRP from the humanitarian operations' perspective and propose that the model based on MDP is adopted for MDDVRPSRC with a known variant of RA (PDS-RA) being applied to solve the problem online. The intuition for the lookahead approach as elaborated above can be identified back to the work of [55] who thought of evaluating a move by thinking ahead in a lookahead manner. The RA is based on the same principle but is refined by means of the quality of the lookahead which depends on the base policy applied. Furthermore, the horizon of the lookahead plays an important role as was mentioned by [64,65]. Another crucial aspect of the rollout is the Monte Carlo sampling that would enable the multiple lookaheads to account for the stochastic parameter for the simulated episode. Finally, the number of

simulations performed would also contribute to a better mean approximation of the state or PDS value. The RA is proposed by [48] belonging to the class type of policy iteration in reinforcement learning. Note that [34] was among the first to develop a specified RA in solving sequencing problems. This work was extended in [35] for a single VRP. In [49], the performance of the rollout policy is compared to the optimistic approximate policy iteration for the single VRP. The former showed a better performance. Cyclic heuristic is applied as the base policy for the RA in [66] to solve the problem of of a VRP with stochastic demand (VRPSD) as well as the problem of the Travelling Salesmen Problem (TSP) with stochastic travel time. The former was also addressed by [36] who presented the two–stage stochastic programming solution approach and compared it to the online approach using RA. The outcome space was addressed exactly for all these problems up to this point. However, Ref. [37] managed to tackle this challenging computation approach by applying Monte Carlo sampling instead in a single- and two-stage RA. The computation was shown to have shrunk by 65% when compared with [35].

In [39], the multi-vehicle VRPSD is finally addressed. The challenges that come with the multi-vehicle problem is addressed through clustering to dedicate a set of different customers to each vehicle. An offline a priori route is computed for respective vehicles, and RA is only applied if a route failure occurred. Although RA was not applied from the get-go, this work highlighted the potential of a clustering mechanism when dealing with a multi-vehicle VRP. This was seen in [40] who made the clustering mechanism dynamic at every decision point as the status of stochastic demands were being updated. The problem was then solved by proposed variants of RA making use of the extension of the MDP framework, pre-decision state (PRE) ,and post-decision state (PDS) , advocated by [67]. Here, the base policy applied was the fixed route heuristic. This heuristic was relaxed forming a known restocking fixed route heuristic in [41], from which policies were iterated and obtained through a local search. These policies were evaluated with the help of the optimal value computed by dynamic programming to help with pruning the search space in the search for a more effective rollout base policy. This base policy was then applied in the RA to obtain a more optimal policy for the problem. The same concept was applied in [68] for the same problem, apart from the duration limit for a single vehicle, where a hybrid of backward and forward recursive dynamic programming was applied instead. In the work of [42], the authors applied RA with the cheapest insertion heuristic as the base policy for the problem of single VRP with stochastic requests. The solution was compared with the solution obtained from a greedy heuristic and VFA, respectively. Meanwhile, Ref. [69] proposed a framework for applying RA for the dynamic and stochastic VRP as an ADP solution approach. In [45], PDS–RA was applied with the VFA method driving the decision rule for the lookahead.

2.6. Rollout Algorithm as Matheuristic

An RA with which the base heuristic is driven by the policy of applying a mathematical programming method could be regarded as a matheuristic method. Among those who applied such a technique are [15,70–73]. In the work of [70], the authors addressed the scheduling problem with RA, where the decision rule was obtained using the quadratic programming approach. In [71], the Mixed Integer Linear Programming (MILP) was applied to obtain the decision rule for the RA in solving the problem of inventory routing with a single vehicle. Such approaches were also seen in [72,73] for solving the inventory routing problem. In other works, Ref. [15] proposed a matheuristic RA method to solve the multi-vehicle routing problem for humanitarian delivery operations by reducing the two-stage stochastic programming model to two reduced models that was dependent on the vehicle's mode of operation: replenishment or serving an emergency shelter.

2.7. Knowledge Gap and Research Contribution

By referring to Section 2.2, it is very clear that a humanitarian VRP, which is being addressed through machine learning, is still lacking. Meanwhile, from the industrial problem's perspective, the literature available shows that solving the VRP problem via reinforcement learning and RA are largely limited to stochastic demands, customers, or request problems. No such approach applied has addressed the problem with stochastic road capacity. The authors intend to fill this gap.

Furthermore, addressing the VRP through MDP formulation and the RA solution method is often limited to those of a single-vehicle problem as seen in the work [35–37,41,42,45,49,66,68]. This is evident even in the published works of the last five years. Those who solve multi-vehicle problems such as the work [39–41] often resort to a clustering or decomposition method of the vehicles to a set of different customers. Such a method, with the exception of a dynamic decomposition method, could not be performed when addressing MDDVRPSRC as the road capacity; thus the route is uncertain. Furthermore, it is not clear how a dynamic clustering or decomposition method could address the problem with stochastic road capacity while also accounting for the split delivery and multi-trip operations. To the best of the authors' knowledge, there is no literature that has proposed the method of building the decision rule on the go, guided by the constructive heuristic as the policy while performing the rollout. Similarly, no known variants of such heuristics have been introduced to allow for decision ruling on the go. Here the authors intend to fill the research gaps by proposing the application of proposed heuristics (TBIH-1–TBIH-5) within the PDS-RA adopted from [13]. In terms of the modelling approach, to the best of the authors' knowledge, no literature addresses the MDDVRPSRC, whether in the form of a mathematical programming model or in the MDP formulation, especially in humanitarian operations settings. Most literature for MDP formulation in VRPs addressed multi-trip operations only for on-route failure occasions. For example, the work such as [74] addressed the split operation also when triggered by the event of route failure due to stochastic demands. Literature such as [44,45] which addressed the multi-period operations, however, do not include the possibility of split delivery operations. Furthermore, most of these models only described operations involving a single vehicle. We differ from existing literature by intentionally allowing multi-trip operation as well as split delivery to address the limitation of delivery trucks during a disaster event rather than a result of route failure. Addressing multi-vehicles leads to the explosion of action space and is often very difficult to solve without resorting to a clustering approach. The authors therefore present here how with a moderate number of destination nodes in various simulated road network, the MDDVRPSRC could be solved without utilising a clustering or dynamic clustering method.

3. MDDVRPSRC MDP Model

3.1. Problem Statement

The problem of MDDVRPSRC focuses on the delivery problem, one of the crucial humanitarian operations during a disaster and post-disaster event. Here, the road network is represented as an undirected incomplete graph $G = (H, E)$ in graph theory. H is the set of nodes in graph G such that $H = \{D\} \bigcup \{S\} \bigcup \{N\}$ where D, S, and N are the set of depots, emergency shelters, and connecting nodes respectively. The connecting nodes represent the junction connecting the edges $(i, j) \in E$, representing the roads such that $E = \{(i, j) : i, j \in N \bigcup S \bigcup D, i \neq j\}$. Note that emergency shelters whose demands are satisfied can act as connecting nodes for vehicles to travel through.

In MDDVRPSRC, the medical supplies are to be delivered to temporary erected emergency shelters, $s \in S$, with different demands, w_s, by a homogeneous fleet of vehicles, $m \in M$, with capacity, q_m. The delivery of medical supplies is conducted via split delivery to account for the limited number of vehicles during the sudden onset of a disaster. Vehicles are allowed to perform multi-trip deliveries throughout the humanitarian operations to satisfy all the demands. The vehicle capacity, q_m, can be replenished to a full capacity, Q,

as soon as they return to the depot, $d \in D$. All vehicles must be dispatched throughout the operation until the total demand is satisfied as there is no guarantee that any assigned vehicles might reach their designated emergency shelter. More on this point, a vehicle is considered stranded when it is unable to travel to the next node on the way to the depot for the consecutive decision points of ST once the total demand is satisfied. Unless such an event occurs, all vehicles must return to the depot. However, they are not constrained as to which depot they should return to, advocating the flexibility needed for the humanitarian operations.

All throughout the operation, the road capacity, $r_{i,j}$, is uncertain. The mean road capacity for each road, (i, j), as well as the capacity distribution deteriorates as the operation time progresses for all damaged roads [15]. The deteriorating road capacity mean is due to the damages inflicted by the subsequent post-disaster events, such as tremors of an earthquake. This is simulated as the outward radial circles originating from the epicentre of the earthquake [15]. For more information of road capacity distribution, road damage determination, and simulation of earthquake tremors, refer to the work of [15].

3.2. Agent Solving MDDVRPSRC in Reinforcement Learning

In reinforcement learning, an agent learns to make decisions based on given information of the system. The information is given in the form of the state representation of the system as well as the transition of the state when making a constrained action or decision. An agent learns to make optimal decisions based on the series of interactions it has made from making decisions and in return obtained rewards. An MDP model formulates how an agent sees the system through: (i) the state representation, (ii) how the system transitions, (iii) the constrained actions or decisions it is allow to make as well, and (iv) the reward it received from making a decision. In this work, the agent perceived the problem as a MDDVRPSRC and will make a near optimal decision at every decision point. This agent then becomes a part of the DSS for delivering critical medical supplies during a disaster.

The agent learns of the aforementioned delivery operations of the MDDVRPSRC through a series of discrete states it observes at each decision point, k, beginning with the initial state, s_0, until the end state, s_K, representing the end of delivery operations. The Pre-Decision State (PRE), s_k, represents the state of the operations at decision point, k, prior to decision, a_k, computed by the agent. The state observed by the agent after decision, a_k, is made and denoted as the Post-Decision State (PDS), $s_k^{a_k}$. These states (PRE and PDS) are described through the state variables l, t, q, w, e, and r, respectively:

- l: current location of all vehicles;
- t: the next arrival time to next destination of all vehicles;
- q: the capacity status of all vehicles;
- w: the demand status for all nodes;
- e: the occupancy status of road (i, j) in relation to vehicle m;
- r: the road capacity of all roads or edges.

In the MDDVRPSRC MDP formulation, an agent computes a decision, a_k, to send a fleet of vehicles to a destination node based on the state that it observed, s_k. Once the decision a_k is executed, the PRE transitions to PDS, $s_k^{a_k}$, deterministically and waits for the next decision point, $k+1$, which is triggered at T_{k+1} by the arrival of one or more vehicles $m \in M'_{k+1}$ simultaneously at the emergency shelter $s \in S$ or connecting node $n \in N$ or depot $d \in D$.

Once the decision point $k+1$ is triggered, the random road capacities $\hat{r_{i,j}}$ are observed for all roads $(i, j) \in E$ through the dynamic update from the locals at the arrival destinations. At this point, the PDS transitions to PRE, s_{k+1}, stochastically and the PRE state variables e_k and r_k are updated. This new random information is now known to the agent (via the updates of e_k and r_k) who then uses it to compute the next decision, a_{k+1}, for all vehicles. Once the decision a_{k+1} is computed, the PRE transitions to PDS, $s_{k+1}^{a_{k+1}}$. At the same time, demand $w_h, \forall h \in S$ is served or the vehicle's capacity is replenished or neither (when a vehicle arrives at connecting node $h \in H \cap \{S + D\}$) depending on where the vehicles

arrived. Hence, the variables of PDS representing the next destination, l^{a_k}, arrival time to next destination t^{a_k}, capacity, q^{a_k}, status edges travelled, e^{a_k} and demand status, w^{a_k} are updated accordingly. The MDDVRPSRC formulation that follows is developed by referring to [75] and the work of [13]. All parameters, state variables. and decision variables are listed in Tables 1 and 2.

Table 1. MDDVRPSRC: Parameters.

Parameters	
N	connecting node set
D	depot set
S	shelter set
H	$N \bigcup S \bigcup D$
E	set of edges $E = \{(i,j) : i,j \in N \bigcup S \bigcup D, i \neq j\}$
M	set of vehicles
K	end decision point
s_k	state at decision point k prior to decision a_k being made, PRE
$s_k^{a_k}$	state at decision point k after decision a_k is made, PDS
s_K	termination PRE state
k	decision point such that $k \in \mathbb{N} \cap \{0, K\}$
M'_k	set of vehicles that arrived at their assigned destination at decision point k where $M'_k \subset M$
T_k	time at decision point k such that $T_k \in \mathbb{R}$
t_{wait}	waiting time for vehicle arrived at a node but is assigned to remain stationary at current node such that $t_{\text{wait}} \in [0, \text{inf})$
$A(s_k)$	decision set of all possible decisions at decision point k given s_k
$A^\pi(s_k)$	MDP decision rule at decision point k following policy π given s_k
$\varrho^\pi(s_k)$	on-the-go constructed lookahead decision rule at decision point k following policy π given s_k
$\eta^\pi(s_k)$	decision rule computed through construction heuristic or CPLEX determined by policy π at decision point k given s_k in the lookahead horizon
O'_m	set of all potential destination for vehicle m at i for all $(i,j) \in E$ where $O'_m \subset H$
O_m	reduced decision space set for a single vehicle m in rollout given that $O_m \subset O'_m$
Q	maximum capacity of vehicles after replenishment at depot
$c_{i,j}$	cost incurred if edge (i,j) is travelled such that $c_{i,j} \in \mathbb{R}$
$t_{i,j}$	time travelled of edge (i,j) such that $t_{i,j} \in \mathbb{R}$
$W(k)$	random information at decision point k
$r_{i,j}$	deterministic road capacity $r_{i,j} \in \mathbb{Z}^\star$, where $Z^\star = \mathbb{Z}^+ \bigcup \{0\}$ observed at decision point k
$\hat{r_{i,j}}$	stochastic road capacity $\hat{r_{i,j}} \in \mathbb{Z}^\star$
$p_{i,j}$	damage unit sustained by road (i,j) [15], such that $p_{i,j} \in \mathbb{N}$
Z	a large negative arrival time for vehicles resting at depot when all demand is served
$G2$	a larger constant acting as reward or penalty
ST	a limit on how many times a vehicle is allowed to be stationary (stranded) consecutively in terms of decision points
$F(m,i)$	function that adds consecutive decision points for all stranded vehicle m at i when all demand is served and all other vehicle is at depot consecutively starting from decision point k_{strand_0} to current decision point k_{strand_k} such that $F : (m,i) \rightarrow \{k_n \vert k_{n-1} - k_n = 1\}_{n=k_{\text{strand}_0}}^{k_{\text{strand}_k}} : \quad \forall k_{\text{strand}_k} \leq ST \quad m \in M, \quad i \in H, \quad r_{i,j} = 0 \quad \forall (i,j) \in E$
λ	discount factor in Bellman Equation [12]
π	policy $\pi \in \Pi \subset \mathbb{N}$ that affect the decision rule $\varrho^\pi(s_k) : s_k \rightarrow a_k$
$R_k(s_k, a_k)$	reward for the agent for making decision a_k at decision point k when observing(given) the state s_k
$R_{k,m}(s_k, a_k)$	individual reward of vehicle m at decision point k

Table 2. MDDVRPSRC: State and Decision Variables.

PRE and PDS Variables	
l_k,	vector of next destination $\forall m \in M$ at decision point k, such that $l_k \in H^{\|M\|} = [l_0, l_1, \ldots, l_{\|M-1\|}]$
l^{a_k},	vector of next destination $\forall m \in M$ at decision point k, such that $l^{a_k} \in H^{\|M\|} = [l_0^{a_k}, l_1^{a_k}, \ldots, l_{\|M-1\|}^{a_k}]$
t_k,	vector of arrival time $\forall m \in M$ to assigned destination at decision point k, such that $t_k \in \mathbb{R}^{\|M\|} = [t_0, t_1, \ldots, t_{\|M-1\|}]$
t^{a_k}	vector of arrival time $\forall m \in M$ to assigned destination at decision point k, such that $t^{a_k} \in \mathbb{R}^{\|M\|} = [t_0^{a_k}, t_1^{a_k}, \ldots, t_{\|M-1\|}^{a_k}]$
q_k,	vector of vehicle capacity $\forall m \in M$ at decision point k, such that $q_k \in \mathbb{R}^{\|M\|} = [q_0, q_1, \ldots, q_{\|M-1\|}]$
q^{a_k},	vector of vehicle capacity $\forall m \in M$ at decision point k, such that $q^{a_k} \in \mathbb{R}^{\|M\|} = [q_0^{a_k}, q_1^{a_k}, \ldots, q_{\|M-1\|}^{a_k}]$
w_k,	vector of demand for all node $i \in H$ at decision point k, such that $w_k \in \mathbb{R}^{\|M\|} = [w_0, w_1, \ldots, w_{\|H-1\|}]$
w^{a_k},	vector of demand for all node $i \in H$ at decision point k, such that $w^{a_k} \in \mathbb{R}^{\|M\|} = [w_0^{a_k}, w_1^{a_k}, \ldots, w_{\|H-1\|}^{a_k}]$
r_k,	vector of road capacity for all edges $(i,j) \in E$ at decision point k, such that $r_k \in \mathbb{R}^{\|E\|} = [r_{i,j}]_{\forall (i,j) \in E}$
r^{a_k},	vector of road capacity for all edges $(i,j) \in E$ at decision point k, such that $r^{a_k} \in \mathbb{R}^{\|E\|} = [r_{i,j}]_{\forall (i,j) \in E}$
e_k,	vector of road occupancy of each vehicle for all $(i,j,m) \in \{(i,j,m) : \forall m \in M, \quad \forall (i,j) \in E\}$ at decision point k, such that $e_k \in \{0,1\}^{\|\{(i,j,m):\ \forall m \in M,\ \ \forall (i,j) \in E\}\|} = [e_{i,j,m}]_{\forall (i,j,m) \in \{(i,j,m):\ \forall m \in M,\ \ \forall (i,j) \in E\}}$ 1 if edge (i,j) travelled by specific vehicle m, 0 otherwise
e^{a_k},	vector of road occupancy of each vehicle for all $(i,j,m) \in \{(i,j,m) : \forall m \in M, \quad \forall (i,j) \in E\}$ at decision point k, such that $e^{a_k} \in \{0,1\}^{\|\{(i,j,m):\ \forall m \in M,\ \ \forall (i,j) \in E\}\|} = [e_{i,j,m}^{a_k}]_{\forall (i,j,m) \in \{(i,j,m):\ \forall m \in M,\ \ \forall (i,j) \in E\}}$ 1 if edge (i,j) travelled by specific vehicle m, 0 otherwise
Decision Variable	
a_k	vector of next assigned destination for vehicles, such that $a_k \in H^{\|M\|} = [a_1, a_2, \ldots, a_{\|M\|}] \in A(s_k)$

3.3. MDDVRPSRC Formulation

The pre-decision state (PRE) s_k is a multi dimensional vector consisting of other vectors representing each state variable, respectively. Within this vector are the state variables defined in Table 2:

$$s_k = [l_k, t_k, q_k, w_k, r_k, e_k]. \tag{1}$$

The PDS representation shares the same features as the PRE differs only by annotation and that its variables are updated after decision a_k is made:

$$s_k^{a_k} = [l^{a_k}, t^{a_k}, q^{a_k}, w^{a_k}, r^{a_k}, e^{a_k}]. \tag{2}$$

Once the decision point is triggered, based on the minimum current values within the arrival time vector t^{a_k}, at:

$$T_k = \min_{m \in M,\, t_m^{a_k} \geq 0} t^{a_k}, \tag{3}$$

the respective decision/assignment is computed for all vehicles including those that arrived as shown in Equation (4) below:

$$M'_k = \underset{m \in M,\, t_m^{a_k} \geq 0}{\arg\min} \quad t^{a_k}. \tag{4}$$

Here, the vehicle that is still en route to its destination during the decision point k is denoted by $\{m \in M \backslash M'_k\}$.

In the MDDVRPSRC, the decision a_k is a $|M|$ dimensional vector in a decision space (a set) $A(s_k)$ given the state s_k. The decision involves assigning the next destination for all vehicles at every decision point k,

$$a_k = a \in H^{|M|} = [a_0, a_1, \ldots, a_{|M-1|}] \in A(s_k). \tag{5}$$

However, the decision space $A(s_k)$ for MDDVRPSRC is too large to obtain a good solution within reasonable computation efforts. Therefore, for every decision point k, the decision by the agent is computed as proposed in [15] where the reduced decision set for a single vehicle O_m is as defined for the rollout in Table 1:

$$\begin{aligned}
A(S_k) = \{a_k \in H^{|M|} :\ & \\
& a_m = j, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} > 0, w_j \neq 0, q_m \neq 0, j \in O_m : O_m \subset S\} \\
& a_m = j, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} > 0, \sum_{h \in H} w_h \neq 0, q_m = 0, j \in O_m : O_m \subset D\} \\
& a_m = j, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} > 0, \sum_{h \in H} w_h = 0, i \notin D, j \in O_m : O_m \subset D\} \\
& a_m = i, \forall \{m \in M'_k : i = l_m, j, d \neq i, r_{i,d} > 0, r_{i,j} = 0, \sum_{h \in H} w_h \neq 0, q_m \neq 0, \\
& \qquad j, d \in O'_m, d \in D\} \\
& a_m = i, \forall \{m \in M'_k : i = l_m, j, s \neq i, r_{i,j} > 0, r_{i,s} = 0, \sum_{h \in H} w_h \neq 0, q_m \neq 0, \\
& \qquad j, s \in O'_m, s \in S\} \\
& a_m = i, \forall \{m \in M'_k : i = l_m, \sum_{h \in H} w_h = 0, i \in D\} \\
& a_m = i, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} = 0 \quad \forall (i,j) \in E\} \\
& a_m = j, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} > 0, \sum_{h \in H} w_h \neq 0, q_m \neq 0, j \in O_m : O_m \subset (H \backslash S)\} \\
& a_m = j, \forall \{m \in M'_k : i = l_m, j \neq i, r_{i,j} > 0, \sum_{h \in H} w_h \neq 0, q_m = 0, j \in O_m : O_m \subset (H \backslash D)\} \\
& a_m = l_m, \forall \{m \in M \setminus M'_k\}\}
\end{aligned} \tag{6}$$

The state S_k transitions deterministically to PDS, $s_k^{a_k}$:

$$s_k^{a_k} = S^{M,a}(s_k, a_k), \tag{7}$$

where the decision is made by the agent ($l^{a_k} = a_k$). This is where the next destination l^{a_k}, arrival time t^{a_k}, capacity of vehicle q^{a_k}, travelled edges status by vehicles e^{a_k}, as well as the demands status w^{a_k} are updated. At this point, the stochastic road capacity is not known; hence r^{a_k} is not updated.

The time of arrival $t_m^{a_k} \in \mathbb{R}$, $\forall m \in M$ is updated to:

$$t_m^{a_k} = \begin{cases} t_m - Z, & \forall\{m \in M'_k : l_m \in D, \sum_{h\in H} w_h = 0\} \\ t_m - Z, & \forall\{m \in M'_k : i = l_m, r_{i,j} = 0, \quad \forall(i,j) \in E, \\ & |F(m,i)| = ST\} \\ t_{wait}, & \forall\{m \in M'_k : a_m = l_m, l_m \notin D, w_h = 0, \forall h \in H, \} \\ T_k + t_{l_m,a_m}, & \forall\{m \in M'_k : l_m \neq a_m, l_m \notin D, \sum_{h\in H} w_h = 0, \\ & p_{l_m,a_m} = 0\} \\ T_k + t_{l_m,a_m} + \left(t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}\right), & \forall\{m \in M'_k : l_m \neq a_m, l_m \notin D, \sum_{h\in H} w_h = 0, \\ & p_{l_m,a_m} \neq 0\} \\ t_{wait}, & \forall\{m \in M'_k : a_m = l_m, \sum_{h\in H} w_h \neq 0, \} \\ T_k + t_{l_m,a_m}, & \forall\{m \in M'_k : l_m \neq a_m, \sum_{h\in H} w_h \neq 0, p_{l_m,a_m} = 0\} \\ T_k + t_{l_m,a_m} + \left(t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}\right), & \forall\{m \in M'_k : l_m \neq a_m, \sum_{h\in H} w_h \neq 0, p_{l_m,a_m} \neq 0\} \\ t_m, & \text{otherwise} \end{cases} \tag{8}$$

where t_{wait} is defined as:

$$t_{wait} = \begin{cases} t_2, & t'_k = (t_1, t_2, ...t_n), \quad t_i \in t_k : t_i \geq 0, t_i - t_{i-1} > 0, n \geq 2 \\ \min\limits_{\forall(i,j)\in E} t_{i,j}, & \text{otherwise} \end{cases}, \tag{9}$$

and t'_k is an n-tuple with an increasing order of arrival time.

Meanwhile, the capacity of all vehicles $m \in M$ is updated to:

$$q_m^{a_k} = \begin{cases} Q, & \forall m \in \{M'_k : l_m \in D\} \\ \max(q_m - w_{l_m}, 0), & \forall m \in \{M'_k : l_m \in S\} \\ q_m, & \text{otherwise} \end{cases}. \tag{10}$$

The travelled edges e^{a_k} are updated $\forall(i,j) \in E$, $\forall m \in M$:

$$e_{i,j,m}^{a_k} = \begin{cases} 1, & \forall m \in \{M'_k : i = l_m, j = a_m, i \neq j\} \\ e_{i,j,m}, & \text{otherwise} \end{cases}. \tag{11}$$

Finally, the demand of emergency shelter is also updated $\forall h \in H$:

$$w_h^{a_k} = \begin{cases} \max(w_{l_m} - q_m, 0), & \forall m \in \{M'_k : l_m \in S\} \\ w_h, & \text{otherwise} \end{cases}. \tag{12}$$

At decision point T_{k+1}, the uncertainty of the road capacity $\hat{r_{i,j}}\forall(i,j) \in E$ is now observed by the agent which leads to the transition from PDS to PRE, s_{k+1}:

$$s_{k+1} = S^{M,W}(s_k^{a_k}, W_{k+1}). \tag{13}$$

The road capacity at this point is no longer uncertain ($r_{i,j}\forall(i,j) \in E$) since it has been sampled/known to vehicles that have arrived at their destinations. This information is thus known to the agent. The next destination $l_m = l_m^{a_k} \quad (\forall m \in M)$, the arrival time $t_m = t_m^{a_k} \quad (\forall m \in M)$, capacity of the vehicle $q_m = q_m^{a_k} \quad (\forall m \in M)$, as well as the shelter demand $w_h = w_h^{a_k} \quad (\forall h \in H)$ remain the same.

The travelled edges status, e_k is updated $\forall m \in M$, $\forall (i,j) \in E$:

$$e_{i,j,m} = \begin{cases} 0, & \forall m \in M'_k \\ e^{a_k}_{i,j,m}, & \text{otherwise} \end{cases}. \tag{14}$$

The road capacity r_k is at this point observed and updated:

$$r_{i,j} = \min((\hat{r}_{i,j} - \sum_{m \in M} e_k(i,j,m)), 0) \quad \forall (i,j) \in E, \tag{15}$$

where the random road capacity $\hat{r}_{i,j} \forall (i,j) \in E$ is obtained from a random Poisson distribution as described in [15].

When transitioning to PDS, $s^{a_k}_k$, the agent receives a reward $R_k(s_k, a_k)$ contributed by all vehicles $m \in M$ at decision point k:

$$R_k(s_k, a_k) = \sum_{m \in M} R_{k,m}(s_k, a_k), \tag{16}$$

where $R_{k,m}(s_k, a_k), \quad \forall m \in M$ is given by:

$$R_{k,m}(s_k, a_k) = \begin{cases} 0, & \forall m \in \{M \backslash M'_k\} \\ 0, & \forall m \in \{M'_k : i = l_m, l_m = a_m, \sum_{h \in H} w_h \neq 0\} \\ (max(w_{l_m} - q_m, 0) \times G2) - c_{l_m,a_m} - t_{l_m,a_m}, & \\ \qquad \forall m \in \{M'_k : q_m \neq 0, l_m \in S, w_{l_m} > q_m, p_{l_m,a_m} = 0\} & \\ (max(q_m - w_{l_m}, 0) \times G2) - c_{l_m,a_m} - t_{l_m,a_m}, & \\ \qquad \forall m \in \{M'_k : q_m \neq 0, l_m \in S, w_{l_m} < q_m, p_{l_m,a_m} = 0\} & \\ (max(w_{l_m} - q_m, 0) \times G2) - c_{l_m,a_m} - (t_{l_m,a_m} + t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}), & \\ \qquad \forall m \in \{M'_k : q_m \neq 0, l_m \in S, w_{l_m} > q_m, p_{l_m,a_m} \neq 0\} & \\ (max(q_m - w_{l_m}, 0) \times G2) - c_{l_m,a_m} - (t_{l_m,a_m} + t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}), & \\ \qquad \forall m \in \{M'_k : q_m \neq 0, l_m \in S, w_{l_m} < q_m, p_{l_m,a_m} \neq 0\} & \\ G2 - c_{l_m,a_m} - t_{l_m,a_m}, & \\ \qquad \forall m \in \{M'_k : l_m \neq a_m, q_m \neq 0, l_m \in S, w_{l_m} = q_m, p_{l_m,a_m} = 0\} & \\ \qquad \text{or} \quad \forall m \in \{M'_k : l_m \neq a_m, q_m = 0, a_m \in D, \sum_{h \in H} w_h \neq 0, & \\ \qquad\qquad p_{l_m,a_m} = 0\} & \\ \qquad \text{or} \quad \forall m \in \{M'_k : l_m \neq a_m, a_m \in D, \sum_{h \in H} w_h = 0, p_{l_m,a_m} = 0\} & \\ G2 - c_{l_m,a_m} - (t_{l_m,a_m} + t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}), & \\ \qquad \forall m \in \{M'_k : l_m \neq a_m, q_m \neq 0, l_m \in S, w_{l_m} = q_m, p_{l_m,a_m} \neq 0\} & \\ \qquad \text{or} \quad \forall m \in \{M'_k : l_m \neq a_m, q_m = 0, a_m \in D, \sum_{h \in H} w_h \neq 0, & \\ \qquad\qquad p_{l_m,a_m} \neq 0\} & \\ \qquad \text{or} \quad \forall m \in \{M'_k : l_m \neq a_m, a_m \in D, \sum_{h \in H} w_h = 0, p_{l_m,a_m} \neq 0\} & \\ -c_{l_m,a_m} - t_{l_m,a_m}, & \text{otherwise}, p_{l_m,a_m} = 0, \quad \forall m \in M'_k \\ -c_{l_m,a_m} - (t_{l_m,a_m} + t_{l_m,a_m} \times \frac{p_{l_m,a_m}}{10}), & \text{otherwise}, p_{l_m,a_m} \neq 0, \quad \forall m \in M'_k \end{cases} \tag{17}$$

Here, a policy of decisions denotes the guiding principle on which the decision is based. For example, a policy $\pi_{\mathcal{B}} \in \Pi$ could be a heuristic if a decision function $A^{\pi_{\mathcal{B}}}(s_k)$ is mapped from that heuristic. In this model formulation, the decision $a_k = A^{\pi}(s_k) \subset A(s_k)$ [67] is selected among other potential decisions in the decision space, $A(s_k)$ following a certain policy $\pi \in \prod$ such that the decision rule function $A^{\pi}(s_k) : s_k \rightarrow a_k$.

The objective (Equation (18)) is to find an optimal policy $\pi^{\star}$ such that the expected total rewards are maximised (objective function for the Bellman optimality equation [67]):

$$\max_{\pi \in \Pi} \quad \mathbb{E}^{\pi} \left\{ \sum_{k=0}^{K} (R_k(s_k, A^{\pi}(s_k)) \right\}. \tag{18}$$

Hence:

$$\pi^{\star} = \arg\max_{\pi \in \Pi} \quad \mathbb{E}^{\pi} \left\{ \sum_{k=0}^{K} (R_k(s_k, A^{\pi}(s_k)) \right\}, \tag{19}$$

where for every decision point k, the optimal decision $a_k^{\star}$ is chosen: $a_k^{\star} = A^{\pi^{\star}}(s_k)$ by following the optimal policy $\pi^{\star}$.

4. MDDVRPSRC Solution Approach

To solve an MDP is to seek an optimal policy $\pi^{\star}$. Through this optimal policy, every decision made by the agent is optimal: $A^{\pi^{\star}}(s_k) : s_k \rightarrow a_k^{\star}$ as stated by the principal of optimality [12]. To obtain the optimal policy, the Bellman Equation is solved [12] such that the optimal decision $a_k^{\star}$ is computed for every decision point k for each given state s_k observed by the agent. This series of optimal computed decisions is said to be guided by the optimal policy $\pi^{\star} \in \Pi$, and therefore the problem formulated in MDP is solved. To compute the optimal decision, $a_k^{\star}$, the Bellman Equation could be written as Equation (20) [67]:

$$a_k^{\star} = \arg\max_{a_k \in A(s_k)} (R_k(s_k, a_k) + \lambda^k \mathbb{E}\{V_{k+1}(s_{k+1})\}). \tag{20}$$

Due to the curse of dimensionality as seen in Equation (20), computing for an optimal decision is often intractable. To alleviate the curse associated with the outcome or transition space, the PDS is introduced [67] (Equation (21)) and the equation was rewritten as in Equation (22):

$$V_k^{a_k}(s_k^{a_k}) = \mathbb{E}\{V_{k+1}(s_{k+1})\}, \tag{21}$$

$$a_k^{\star} = \arg\max_{a_k \in A(s_k)} (R_k(s_k, a_k) + \lambda^k V_k^{a_k}(s_k^{a_k})). \tag{22}$$

Even with PDS introduced as above, the computation for an optimal decision is usually challenging, especially when computing the value of the PDS, $V_k^{a_k}(s_k^{a_k})$ in Equation (22). The ADP approach approximates the value of PDS instead. This is to deal with the large state space in the MDP. Through the ADP approach, Equation (22) is rewritten as Equation (23), where the value of PDS is approximated and thus the decision is computed to near optimality instead:

$$\widetilde{a_k^{\star}} = \arg\max_{a_k \in A(s_k)} (R_k(s_k, a_k) + \lambda^k \overline{V_k^{a_k}}(s_k^{a_k})). \tag{23}$$

For MDDVRPSRC, this equation is still challenging to solve since the decision space $A(s_k)$ in Equation (23) is too large for a practical number of vehicles to be involved. Furthermore, the decisions consist of combinations of vehicle assignments that would require a long rollout horizon as well as a large number of Monte Carlo simulations. The concern is that the reward obtained for one vehicle may exaggerate the value of the decision for all vehicles collectively if computed prematurely. This is seen in the initial experiments

where, given limited computation capabilities on the machine, inefficient assignments for vehicles resulted.

Alternatively, to cope with this challenge, the near optimal decision could be further approximated as in Equation (24) as proposed in work [15]:

$$\widetilde{a_k^\star} \in H^{|M|} \approx [\widetilde{a_1^\star}, \widetilde{a_2^\star}, \ldots, \widetilde{a_{|M|}^\star}] \quad \forall k, \tag{24}$$

where the decision space, made of combinations of vehicle assignments (which could be astronomical), could be restricted to a decision space of possible assignments for each vehicle O_m instead, as shown in Equation (25):

$$\widetilde{a_m^\star} = \underset{a_m \in O_m}{\arg\max}(R_k(s_k, a_k) + \lambda^k \overline{V_k^{a_m}}(s_k^{a_m})) \quad \forall (k, m). \tag{25}$$

Even with such measures, given the machine's capabilities, the computation is only limited to a small number of rollout horizons and Monte Carlo simulations. However, it is shown in this work that the decisions computed are applicable to this type of problem.

To compute Equation (25), the PDS value, $\overline{V_k^{a_m}}(s_k^{a_m})$ is approximated using PDS-RA, as proposed by [40]. However, different base heuristics can be applied to solve for the MDP problems with characteristics such as the one in this work. Finally, this approach is applied to reciprocate the model for the agent's decision in Equation (6).

4.1. PDS-RA Algorithm

PDS–RA is one of the RA families first introduced in [40] as an ADP solution algorithm to the dynamic VRP with stochastic demand. PDS–RA takes advantage of the PDS structures that alleviate the problem associated with the outcome or transition space. This thus reduces the number of rollout executions compared to the conventional RA to approximate the value of PDS in a modified Bellman Equation effectively. The general PDS-RA could be referred to in Algorithm 1. Here, the rollout transitioned PDS (simulation) is denoted as s^a to avoid confusion with the real-time transitioned PDS, $(s_k^{a_k})$ observed by the agent. In this algorithm, the values of PDS, $V_k^{a_k}(s_k^{a_k})$ associated with each respected $a_k \in A(s_k)$ is approximated $(\overline{V_k^{a_k}}(s_k^{a_k}))$. For each possible PDS associated with the next decision $a_k \in A(s_k)$, the PDS-RA is executed, and by the end of the execution, the approximated value of PDS, $\overline{V_k^{a_k}}(s_k^{a_k})$ is obtained. In every execution of PDS-RA, the base policy $\pi_{\mathcal{B}(s_k^{a_k})} \in \Pi$ is first assigned (policy to apply heuristic $\mathcal{B}$), computing the decision rule function $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$ for the rollout simulation is based on the heuristic $\mathcal{B}$ performed given the PDS, $s_k^{a_k}$. Based on this specific decision rule (which is normally the assignment of vehicles to next destination for VRP), the lookahead into the future as far as horizon K is performed. Transiting from the simulated PRE to PDS is enabled by referring to $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$ when making a decision during the lookahead. This decision is followed by a stochastic transition, transitioning PDS back to PRE ($s_k = S^{M,W}(s^a, W_{k+1}(\omega(k+1)))$) in the lookahead simulation. Here, the random information of road capacities for all roads, $(W_{k+1}(\omega(k+1)))$, is known by sampling $\omega(k+1) \in \Omega$ through a known distribution as part of the Monte Carlo simulation approach. By sampling $\omega(k+1) \in \Omega$, the exhaustive computation for all random transitions of outcomes in the outcome space Ω is prevented as first observed by [37] in her application of RA.

Each time the transition occurs within the lookahead along the horizon, rewards $R_k(s_k, a_k)$ are consecutively amassed. At the end of a one-episode lookahead simulation, the sum of rollout rewards $\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K)$ is obtained. This value is used to update the approximated PDS value of the respective potential decision a_k through an incremental mean approach (Algorithm 1: line 14). The process repeats for N Monte Carlo simulation episodes. After this number of Monte Carlo simulations, the resulting updated approximated PDS value, $\overline{V_k^{a_k}}^N(s_k^{a_k})$ is considered good enough an estimation for the respective

decision a_k. This approximated PDS value is mapped to the respective decision by a function: $f : a_k \leftarrow \overline{V_k^{a_k}}^N(s_k^{a_k})$.

Algorithm 1 Compute $\overline{V_k^a}(s_k^{a_k})$ (as shown in [75] based on PDS-RA proposed by [13] and highlighted in [15]).

Require: s_k, λ, a_k
Ensure: $\overline{V_k^a}(s_k^{a_k})$
Initialise $n, k, R_k(s_k, a_k), B^n$
$s_k^{a_k} = S^{M,a}(s_k, a_k)$
$\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k) \leftarrow \pi_{\mathcal{B}(s_k^{a_k})} \leftarrow \mathcal{B}(s_k^{a_k})$
while $n \leq N$ **do**
  $s^a \leftarrow s_k^{a_k}$
  while $k \neq K$ **do**
    $R_k(s_k, a_k) = R_k(s_k, a_k) + \lambda^k R_k(s_k, a_k)$
    $s_k = S^{M,W}(s^a, W_{k+1}(\omega(k+1)))$
    $a_k \leftarrow \varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$
    $s^a = S^{M,a}(s_k, a_k)$
    $k = k + 1$
  end while
  $\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) \leftarrow R_k(s_k, a_k)$
  $\overline{V_k^a}^n(s_k^{a_k}) = \overline{V_k^a}^{n-1}(s_k^{a_k}) + \frac{1}{n}\left(\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) - \overline{V_k^a}^{n-1}(s_k^{a_k})\right)$
  $n = n + 1$
end while
return $\overline{V_k^{a_k}}^N(s_k^{a_k})$

The PDS-RA is then executed for the next possible decision $a_k \in A(s_k)$ after which the process repeats. Finally, when the PDS-RA is executed $\forall a_k \in A(s_k)$, Equation (23) can now be computed based on all approximated PDS values associated with each respective potential next decision for agent to make. Based on the computation of Equation (23), a decision is then made.

It should be noted that Algorithm 1 is applied to the rollout and looks into the future of each potential decision where each decision a_k revolves on the assignments of all vehicles $a_m = a_k[m], \forall m \in M$ simultaneously per PDS-RA. The base policy $\pi_{\mathcal{B}(s_k^{a_k})}$ guides the construction of the decision function $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$ in one go, per PDS-RA execution $\forall a_k \in A(s_k)$.

In the applied solution, the rollout is executed for every potential next destination for each vehicle $a_m, \forall m \in M$, and the value for each PDS associated with the potential next destination is computed by PDS-RA, such that near optimal assignments $\widehat{a_m^\star}$ would be computed in Equation (25). These near optimal individual assignments form the near optimal decision as described in Equation (24). The base policy $\pi_{\mathcal{B}(s_k^{a_k})}$ is based on an iterative policy $\pi_{\mathcal{B}(s_k)}$ applied at each lookahead decision point s_k to construct the decision rule $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ on-the-go using constructive base heuristics $\mathcal{B}$. The decision rule $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ is constructed on the go such that $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ where $\eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ is the decision rule computed at every decision epoch k in the lookahead horizon when applying heuristic $\mathcal{B}$ based on the rollout state s_k according to the iterative policy $\pi_{\mathcal{B}(s_k)}$.

In Algorithm 2, for example, CPLEX (denoted as $CPLEX$) is applied instead as the base heuristic. Here, CPLEX is run for a Stochastic Linear Integer Programming (SILP) version of the reduced MDDVRPSRC to construct $\varrho^{\pi_{CP(s_k^{a_k})}}(s_k)$ on the go given the current rollout state s_k and this results in $\eta^{\pi_{CP(s_k)}}(s_k)$. Since here the policy is to apply CPLEX, we denote that the policy that the decision rule follows is of $\pi_{CP(s_k^{a_k})}$.

A detailed explanation on this matheuristic is given in [15] (Algorithm 2). The authors used this exact configuration (with $CPLEX$ as the base heuristic) as a benchmark where tractable. As a solution approach in general, the Algorithm 3 is referred. The authors first introduced the TBIH-1 heuristic (Algorithm 4) based on a pure random insertion for the non-obvious decisions. Furthermore, other constructive heuristics were applied dynamically (TBIH-2 and TBIH-3), extended from TBIH-1, to construct the decision rule on the go ($\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$) as shown in Algorithm 3, in contrast to Algorithm 2. Additionally from these constructive heuristics, the authors propose another two new variants (TBIH-4 and TBIH-5) for this problem by introducing the exploitation mechanism on both TBIH-2 and TBIH-3.

Algorithm 2 Matheuristic Extended from Algorithm 1 to Compute $\widetilde{a_m^\star}$ [15].

Require: s_k, λ, a_k
Ensure: $a_m^\star$
for $a_m \in O_m$ at decision point k **do**
  Initialise $n, k, R_k(s_k, a_k), B^n$
  while $n \leq N$ **do**
    $s_k^{a_m} = S^{M,a}(s_k, a_m)$
    $s^a \leftarrow s_k^{a_m}$
    while $k \neq K$ **do**
      $R_k(s_k, a_k) = R_k(s_k, a_k) + \lambda^k R_k(s_k, a_k)$
      $s_k = S^{M,W}(s^a, W_{k+1}(n(k+1)))$
      $\pi_{CP_k(s_k)} \leftarrow CPLEX(s_k)$
      $\eta^{\pi_{\mathcal{B}(s_k)}} \leftarrow \pi_{CP_k(s_k)}$
      $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$
      $a_m \leftarrow \varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$
      $s^a = S^{M,a}(s_k, a_m)$
      $k = k + 1$
    end while
    $\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) \leftarrow R_k(s_k, a_k)$
    $\overline{V_k^{a_m}}^n(s_k^{a_m}) = \overline{V_k^{a_m}}^{n-1}(s_k^{a_m}) + \frac{1}{n}\left(\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) - \overline{V_k^{a_m}}^{n-1}(s_k^{a_m})\right)$
    $n = n + 1$
  end while
  $f : a_m \leftarrow \overline{V_k^{a_m}}^N(s_k^{a_m})$
end for
$\widetilde{a_m^\star} = \underset{a_m \in O_m}{\arg\max}(R_k(s_k, a_k) + \lambda^k f(a_m)) \quad \forall (k, m)$
return $\widetilde{a_m^\star}$

Algorithm 3 Compute $\widetilde{a_m^\star}$ based on [15] using other Base Heuristic, $\mathcal{B}$.

Require: s_k, λ, a_k
Ensure: $a_m^\star$
for $a_m \in O_m$ at decision point k **do**
  Initialise $n, k, R_k(s_k, a_k), B^n$
  while $n \leq N$ **do**
    $s_k^{a_m} = S^{M,a}(s_k, a_m)$
    $s^a \leftarrow s_k^{a_m}$
    while $k \neq K$ **do**
      $R_k(s_k, a_k) = R_k(s_k, a_k) + \lambda^k R_k(s_k, a_k)$
      $s_k = S^{M,W}(s^a, W_{k+1}(n(k+1)))$
      $\pi_{\mathcal{B}(s_k)} \leftarrow \mathcal{B}(s_k)$
      $\eta^{\pi_{\mathcal{B}(s_k)}} \leftarrow \pi_{\mathcal{B}(s_k)}$
      $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$
      $a_m \leftarrow \varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$
      $s^a = S^{M,a}(s_k, a_m)$
      $k = k + 1$
    end while
    $\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) \leftarrow R_k(s_k, a_k)$
    $\overline{V_k^{a_m}}^n(s_k^{a_m}) = \overline{V_k^{a_m}}^{n-1}(s_k^{a_m}) + \frac{1}{n}\left(\widehat{B}^n(\pi_{\mathcal{B}(s_k^{a_k})}, k+1, K) - \overline{V_k^{a_m}}^{n-1}(s_k^{a_m})\right)$
    $n = n + 1$
  end while
  $f : a_m \leftarrow \overline{V_k^{a_m}}^N(s_k^{a_m})$
end for
$\widetilde{a_m^\star} = \arg\max_{a_m \in O_m}(R_k(s_k, a_k) + \lambda^k f(a_m)) \quad \forall (k, m)$
return $\widetilde{a_m^\star}$

4.2. Teach Base Insertion Heuristic (TBIH-1)

In this section, the base heuristics applied are described in general (Algorithm 4), and the elaboration of each is described in the subsections that follow. TBIH-1, TBIH-2, TBIH-3, TBIH-4, and TBIH-5 are the heuristics applied in this work to both validate the model and to cross-compare the performance for each of the models.

The algorithm for each heuristic applied here follows the same main structure of: (i) the teaching part (TP) and (ii) the seeking part (SP).

In the TP, the obvious decisions are chosen without running any heuristics to search for the best next destination. These obvious decisions are stated in Equation (6) and applied to each vehicle:

- To serve any shelter $s \in S$ randomly among possible shelters as the next destination;
- To replenish at any depots $d \in D$ randomly among possible depots as the next destination;
- To return to any depots $d \in D$ randomly among possible depots when all demands have been served;
- To remain stationary at the current arrival node i while still having capacity ($q_m \neq 0$) and demands that have not all been served, if the only next possible destination is to a depot;
- To wait at the current arrival node i while still having capacity ($q_m \neq 0$) if the road capacity to the next possible shelter is blocked $r_{i,j} = 0$;
- To remain resting at the depot if the current arrival node is a depot $i \in D$ and all demands have been served;
- To remain stationary at the current arrival node i if all road capacity to a neighbouring node j are blocked $r_{i,j} = 0, \forall (i, j) \in E$.

Algorithm 4 TBIH-1 and General Structure Algorithm. (Note: $|M'_k| \leq 1$ in rollout).

Require: s_k, M'_k, $M \backslash M'_k$
Ensure: *Decision* during lookahead

update $q_m, \forall m \in M$ and $w_h, \forall h \in H$ as in Equation (10) and Equation (12)
unserved shelters vector, $US = [h]_{\forall h \in H: w_h \neq 0}$
for $m \in M'_k$ **do**
    i = l_m
    potential next destination vector, $next = [h]_{\forall h \in H: r_{i,h} \in E, r_{i,h} > 0}$
    init empty list $nextS, nextD, nextND, Decision$
    $nextS = [i : i \in (next \cap US)]$
    $nextD = [i : i \in (next \cap D)]$
    $nextND = [i : i \in (next \notin D)]$
    if $q_m \neq 0$ AND len($nextS$!=0) **then**
        if len($nextS$)!=1 **then**
            a_m = random.choice($nextS$)
            while len($nextS$)!=0 **do**
                if $r_{i,a_m} > 0$ **then**
                    Decision.append(a_m)
                    $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
                    break
                else
                    $nextS$.remove(a_m)
                    if len($nextS$)!=0 **then**
                        a_m = random.choice($nextS$)
                    else
                        continue
                    end if
                end if
            end while
            if $Decision$!= 0 **then**
                break
            else
                continue
            end if
        else
            a_m = random.choice($nextS$)
            if $r_{i,a_m} > 0$ **then**
                Decision.append(a_m)
                $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
                break
            else
                continue
            end if
        end if
    else if $q_m == 0$ AND len($nextD$!=0) **then**
        if len($nextD$)!=1 **then**
            a_m = random.choice($nextD$)
            while len($nextD$)!=0 **do**
                if $r_{i,a_m} > 0$ **then**
                    Decision.append(a_m)
                    $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
                    break

Algorithm 4 *Cont.*

```
            else
                $nextD$.remove($a_m$)
                if len($nextD$)!=0 then
                    $a_m$ = random.choice($nextD$)
                else
                    continue
                end if
            end if
        end while
        if $Decision$!= 0 then
            break
        else
            continue
        end if
    else
        $a_m$ = random.choice($nextD$)
        if $r_{i,a_m} > 0$ then
            Decision.append($a_m$)
            $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
            break
        else
            continue
        end if
    end if
else if len($US$)==0 AND len($nextD$!=0) then
    if len($nextD$)!=1 then
        $a_m$ = random.choice($nextD$)
        while len($nextD$)!=0 do
            if $r_{i,a_m} > 0$ then
                Decision.append($a_m$)
                $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
                break
            else
                $nextD$.remove($a_m$)
                if len($nextD$)!=0 then
                    $a_m$ = random.choice($nextD$)
                else
                    continue
                end if
            end if
        end while
        if $Decision$!= 0 then
            break
        else
            continue
        end if
    else
        $a_m$ = random.choice($nextD$)
        if $r_{i,a_m} > 0$ then
            Decision.append($a_m$)
            $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
            break
```

Algorithm 4 *Cont.*

```
            else
                continue
            end if
        end if
    else if q_m!=0 AND len(US)!=0 AND len(nextD)!=0 AND len(nextS)== 0 AND len(nextND)== 0 then
        Decision.append(i)
        break
    else if q_m!=0 AND ANY(h, ∀h ∈ US ∩ next) AND len(nextS)== 0 AND ANY(w_h ≠ 0, ∀h ∈ next) then
        Decision.append(i)
        break
    else if ALL(w_h == 0, ∀h ∈ H) AND i ∈ D then
        Decision.append(i)
        break
    else if ALL(r_{i,j} == 0, ∀j ∈ next, if (i, j) ∈ E) then
        Decision.append(i)
        break
    else
        if len(nextND) > 1 then
            a_m = random.choice(nextND)
            while len(nextND)!=0 do
                if r_{i,a_m} > 0 then
                    Decision.append(a_m)
                    r_{i,a_m} = max(r_{i,a_m} − 1, 0)
                    break
                else
                    nextD.remove(a_m)
                    if len(nextND)!=0 then
                        a_m = random.choice(nextND)
                    else
                        continue
                    end if
                end if
            end while
            if Decision!= 0 then
                break
            else
                continue
            end if
        else
            Decision.extend(nextND)
            break
        end if
    end if
end for
if M\M'_k then
    for m ∈ M\M'_k do
        Decision[m] = l_m
    end for
end if
if len(Decision) == 0 then
    print("error")
end if
    return Decision
```

These decisions are considered obvious decisions where computation efforts should not be focused on. Instead, only when all of the above TP decisions are not applicable (no obvious decisions), will the heuristic be applied. Ideally, none of these obvious decisions should be specified. Instead, the agent should be able to figure out and learn the obvious decision based on the reward obtained. However, such an ideal mechanism would require a humongous amount of rollout episodes with horizons extending to the termination state on each of them. For practical purposes, limited by computation power, the obvious decisions are filtered out to avoid extensive computation efforts. Hence the term "teach" in TBIH-1's "teaching part" (TP).

For the SP, a purely random selection of the next destinations could be applied as in TBIH-1 (Algorithm 4 (line 122)). In the proposed TBIH-1, the obvious decisions are inserted by "teaching". The non-obvious decision is decided by a purely random selection among possible next destinations. The general structure of TBIH-1 is described in Algorithm 4 where the SP part is shown in line (121–147).

The TP consists of updating the capacity of all the vehicles as well as the demand of the shelters (Algorithm 4 (line 1)). This is performed so that the decision selected is based on the current status of the demand and capacity which are otherwise updated/observed by the agent during the transition from PRE to PDS. The next part involves determining the potential next destinations j for vehicle m and sorting these destinations whether they are emergency shelters, depots or non–depot nodes (line 5–9). Afterwards, the obvious decision selection follows as described in Equation (6) (line 10–120). The SP is executed if none of the obvious decisions are suitable (line 121–147). For en route vehicles, the decision is to remain at their current destination (line 149–153). Finally, a decision is selected and returned by the algorithm.

Instead of a purely random selection as in Algorithm 4 (line 121–147), there could be more meaningful guided approaches to select the next destination for the SP. From here, an extend Algorithm 4 (line 121–147) shows the possibilities of inserting a better possible next destination in the route by applying a dynamic SIH-I1 (DSIH) (Algorithm 5) in TBIH-2 and a DCW in TBIH-3 (Algorithm 6), in their respective SP. The authors also experimented with the proposed heuristics TBIH-4 (with an embedded Dynamic Lookahead SIH (DLASIH) in the SP) (Algorithm 7) and TBIH-5 (with an embedded Dynamic Lookahead CW (DLACW) in the SP) (Algorithm 8) to see if both aforementioned heuristics could be enhanced further for better insertion.

4.3. TBIH-2

Among the first to develop SIH is [76], whose work is based on the generalised savings algorithm. Ref. [77] then introduced three types of SIH to solve the VRP and scheduling problem with a time window. The proposed SIH (I1) constructs a route by considering two criteria: the first involves determining the best place for insertion, c_1 based on $c_{1,1}$ and $c_{1,2}$. The second is the consideration for the best un-routed node v to be inserted c_2. For the VRP considering time windows, the SIH (I1) is computed with the following equations [77]:

$$c_{1,1}(i,v,j) = c_{i,v} + c_{v,j} - \xi c_{i,j} \quad \xi \geq 0,$$

$$c_{1,2}(i,v,j) = b_{j_v} - b_j,$$

where $c_{1,1}$ is the generalised savings, and $c_{1,2}$ is the time difference between the new service time for j, b_{j_v} and the time prior to insertion of v. Together, the best insertion place of v is computed as $c(i(v),v,j(v))$ given by:

$$c(i(v),v,j(v)) = \min_{(i,v,j)\in c_1} c_1(i,v,j),$$

where c_1 is given as:

$$c_1 = \theta_1 c_{1,1}(i,v,j) + \theta_2 c_{1,2}, \quad \theta_1 + \theta_2 = 1.$$

Meanwhile, the best v insertion criterion c_2 is given by:

$$c_2(i, v, j) = \zeta c_{0,v} - c_1(i, v, j), \quad \zeta \geq 0,$$

where node 0 in the formulation is the depot; v is then chosen based on:

$$v^\star : (i, v^\star, j) = \underset{(i,v^\star,j)\in c_2}{\arg\max} \quad c_2.$$

Since the MDDVRPSRC does not consider time windows, the θ_2 value is given the value of zero and therefore $c_{1,2}$ is not considered. This turns c_1 into a generalisation of [76]. Furthermore, both ζ and θ_1 are given the value of one. Both node 0 and i are considered as the current position of vehicle m at s_k during the lookahead. In the DSIH, the seed j is chosen randomly by looking one step ahead beyond the next destination of m currently at i in a set such that $\{j : (v, j) \in E, r_{v,j} > 0, j \neq i, j \notin$ (set of the immediate neighbor of i), $\forall j \in H, \quad \forall v \in$ (set of the immediate neighbor of i)$\}$. This is illustrated in Figure 1a–c. Here, the road capacity is not considered to simplify the description of the DSIH. In Figure 1a, the current position of vehicle m is at node 0 (not a depot) with capacity to serve. Node S5 is an emergency shelter with demand, while the rest are connecting nodes. The purpose of the DSIH is to treat the next possible destination from the current position as v by treating j as the seed when constructing a route from node 0. In Figure 1b, the node j is identified as node S9, node 4, and node 3. In the DSIH, the seed j is then chosen randomly among these three potential seeds. In Figure 1c, node 3 is chosen randomly as the seed j. Here, two possible nodes could be inserted as the v: node 1 and node 5. After applying the SIH(I1) (without time window consideration, θ_1 and ζ is given the value one), node 5 is considered the best inserted node and the route $\eta^{\pi_{\mathcal{B}(s_k)}}$ is constructed from node 0–5–3. The next destination of the vehicle from node 0, $a_m = \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ is then selected as node 5. This also means that at the lookahead decision point k, the lookahead route $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ is constructed on the go/updated such that $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \to \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ with heuristic $\mathcal{B}$ as per TBIH-2. In addition, when applying TBIH-2, the route-based MDP concept is applied [14]. This means that at the lookahead decision point $k+1$, the vehicle may not necessarily move to node 3 ($\eta^{\pi_{\mathcal{B}(s_k)}}(s_{k+1})$) next upon arriving at node 5 as the DSIH will be executed at every decision point. In this example, the road capacity is ignored to simplify the explanation of the DSIH that is used in the SP of TBIH-2. In reality, if the edge $(0, 1)$ has no road capacity available ($r_{0,1} = 0$), then SIH(I1) will not be applied as the only possible v is node 5.

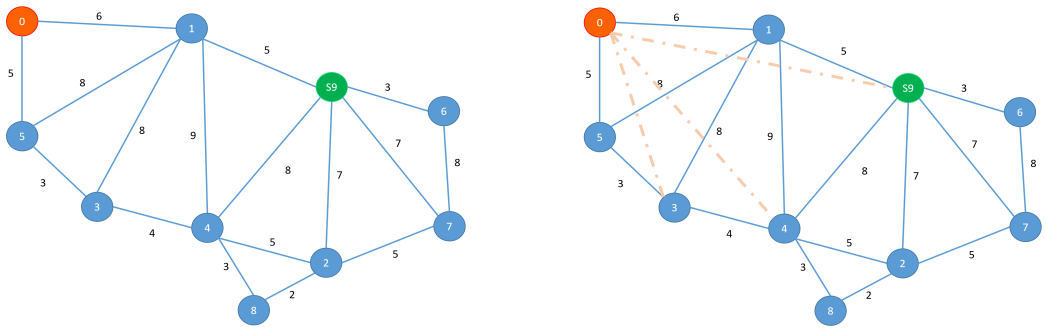
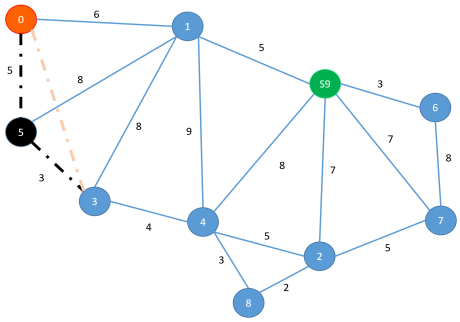


Figure 1. *Cont.*



(**c**)

Figure 1. Performing the DSIH in TBIH-2 in an example network (**a**) with node 0 as current position of vehicle m and node S9 as an emergency shelter. The potential seeds j are considered in (**b**) and chosen randomly in (**c**). As a result, node 5 (v) is inserted and route (0–5–3) is constructed.

The decision selection specific to the DSIH is highlighted in Algorithm 5. Here, the possible seed candidates are selected based on the neighbours of potential destination v for vehicle m (lines 3–5). The seed is then randomly selected (line 7) from which the number of potential destinations v is reduced (line 8). For each of the possible v, the insertion criteria are evaluated according to the SIH(I1) (lines 9–10), and the insertion of v is determined (lines 11–12) from which the decision is made and returned (line 18).

Algorithm 5 TBIH-2 (with Embedded DSIH) Algorithm.

Require: s_k, M'_k, $M \backslash M'_k$, current position of vehicle
Ensure: *Decision*

1: Perform Algorithm 4 (line 1–120)
2: **if** len($nextND$)> 1 **then**
3: $\quad dict : v \leftarrow \{j : \forall j \in H, (v,j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
4: $\quad newnextND = \{j : \forall j \in dict(v), \forall v \in nextND, dict(v) \neq \{\}\}$
5: $\quad$ remove duplicate node: list(set($newnextND$))
6: $\quad$ **if** len($newnextND$)!= 0 **then**
7: $\quad\quad seed$ = random.choice($newnextND$)
8: $\quad\quad$ list of potential inserted nodes based on the selected $seed$, $InsertList = \{v : \forall v \in nextND, (v, seed) \in E\}$
9: $\quad\quad dict$C11 : $(i, v, seed) \leftarrow (c_{i,v} + c_{v,seed} - c_{i,seed}), \quad \forall j \in InsertList$
10: $\quad\quad dict$C2 : $(i, v, seed) \leftarrow (\lambda c_{i,seed} - dict\text{C1}(i, v, seed)), \quad \forall v \in InsertList$
11: $\quad\quad$ choose v from $(i, v, seed) = \underset{(i,v,seed) \in dict\text{C2}}{\arg\max} (dict\text{C2})$
12: $\quad\quad a_m = v$
13: $\quad\quad Decision$.append(a_m)
14: $\quad\quad r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
15: $\quad\quad$ break
16: $\quad$ **else**
17: $\quad\quad a_m$ = random.choice($nextND$)
18: $\quad\quad Decision$.append(a_m)
19: $\quad\quad r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
20: $\quad\quad$ break
21: $\quad$ **end if**
22: **else**
23: $\quad Decision$.extend($nextND$)
24: $\quad$ break
25: **end if**
26: continue Algorithm 4 (148–156)
return *Decision*

4.4. TBIH-3

The CW is derived from the work of [78] as an alternative heuristic solution to the method proposed in [79] to solve the general VRP [80]. This algorithm which is also known as the savings algorithm [81] is based on the savings computed for two non-depot nodes (i, j) in a complete graph (where all non-depot nodes have arcs connecting them to the depot). The savings is computed as $\mathbb{S}_{i,j} = 2c_{i,0} + 2c_{j,0} - (c_{i,0} + c_{j,0} + c_{i,j})$ where 0 is the depot [81]. The route is then constructed based on the savings computed for all non-depot nodes in a decreasing order, provided that the capacity constraint is respected and the connection between edges are allowed (normally the theoretical application onto a complete graph). In the event that the capacity constraint is violated, a new route is constructed for the next vehicle in the same manner of the remaining savings pair. The concept of the CW algorithm is illustrated in Figure 2 [82]. One of the few surveys on the CW algorithm for VRP can be referred to in [83]. A good example of the CW application can be referred to in [84].

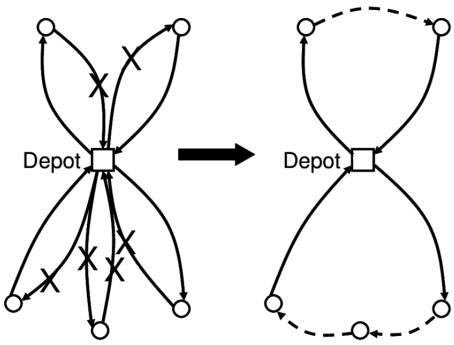


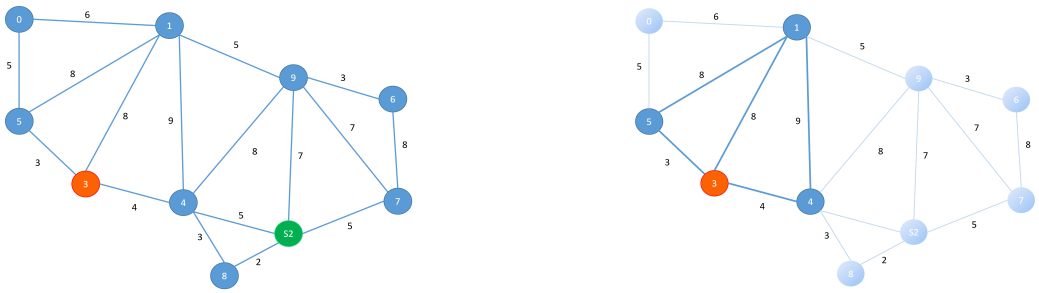
Figure 2. Concept of the Savings Algorithm. Reprinted/adapted with permission from Ref. [82]. 2015, ProQuest Information and Learning Company.

In this work, the application of the Dynamic Clarke and Wright Algorithm (DCW) is proposed and applied. By replacing the random selection of the TBIH-1 in the SP with the DCW, TBIH-1 is then modified to form TBIH-3 and is used as the base heuristic (among other heuristics) in the execution of the PDS–RA. In the DCW, the route-based MDP approach [14] is adopted during the rollout resulting in the on-the-go construction of the decision function $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$. The idea is to apply the CW iteratively ($\pi_{\mathcal{B}(s_k)}$) when a decision for a single vehicle's next assignment during the lookahead is required, given that the TBIH-3 has been selected as the base heuristic. Iteratively, the CW is applied when no obvious decision can be taken during the lookahead when transitioning. At the point of decision, the current position l_m is regarded as the single depot in the CW while the neighbouring nodes $j \in O_m : O_m \subset H$ are considered customers. The example for applying the DCW is illustrated in Figure 3a,b where node 3 is the current arrival spot of vehicle m. Additionally, a shelter in the network example is denoted as the node S2. Using the CW, route $\eta^{\pi_{\mathcal{B}(s_k)}}$ is constructed from the assumed depot (node 3) and back to the depot.

An example of a constructed route $\eta^{\pi_{\mathcal{B}(s_k)}}$ could be $(3-5-1-4-3)$ or $(3-4-1-5-3)$, or both if both edges (5,1) and (4,1) happen to have the highest savings. a_m would then be chosen as the first insertion $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ of the chosen route to transition within the lookahead horizon.

The algorithm for the TBIH-3 is shown in Algorithm 6 as the extension of Algorithm 4 by replacing lines 121–147. In this algorithm, a decision is computed if no other obvious decision can be chosen. To execute the CW, all possible pairs $(j, k) \in E$ are detected from the possible next destination nodes in the list $nextND$ (lines 3–4). If there are no edges that exist, a randomly selected node is chosen as the next destination for the vehicle m (lines 5–9). Otherwise, the savings are computed from these edges and sorted in decreasing order (lines 11–12) prior to constructing the temporary decision function $\eta^{\pi_{\mathcal{B}(s_k)}}$ (line 16)

which constructs the on-the-go base decision function $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ (line 17). Lines 148–156 are similar to Algorithm 4 where the computed decision is returned.



(**a**) (**b**)

Figure 3. Performing DCW in TBIH-3 in an example network (**a**) with node 3 as current position of vehicle m and node S2 as an emergency shelter. The components for performing CW are selected in (**b**).

Algorithm 6 TBIH-3 Algorithm.

Require: s_k, M'_k, $M \backslash M'_k$, current position of vehicle
Ensure: *Decision*
1: Perform Algorithm 4 (line 1–120)
2: **if** len($nextND$) > 1 **then**
3: Possible edges from the pair that could be formed in $nextND$, $Pairs = \binom{nextND}{2}$
4: Remove $(j,k) \in Pairs$, if $(j,k) \notin E$
5: **if** len($pairs == 0$) **then**
6: a_m = random.choice($nextND$)
7: *Decision*.append(a_m)
8: $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
9: break
10: **else**
11: compute savings: $(j,k) \leftarrow c_{i,j} + c_{i,k} - c_{j,k} \quad \forall (j,k) \in Pairs$
12: sort $(j,k) \in Pairs$ with decreasing savings
13: **if** len($Pairs$)==1 **then**
14: $a_m = j : (j,k) \in Pairs$
15: **else**
16: construct route $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ from $i = l_m$ by inserting $(j,k) \in Pairs$ as would be done in CW (decreasing savings)
17: $a_m \leftarrow A^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$
18: **end if**
19: *Decision*.append(a_m)
20: $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
21: break
22: **end if**
23: **else**
24: *Decision*.extend($nextND$)
25: break
26: **end if**
27: continue Algorithm 4 (148–156)
return *Decision*

4.5. TBIH-4

The application of the SIH in the MDDVRPSRC for the rollout is quite clear, given the chosen "seed customer" is the main driver of the method. In the MDDVRPSRC, the authors concur with [85] that choosing an appropriate seed is very important for insertion heuristics.

For this particular problem of depending on the capacity status of vehicle m, the seed is either the depot in the route to replenish capacity or the emergency shelter in the route to serve a shelter. This is different from Algorithm 5 where the seed is randomly chosen based on potential destinations v's neighbour. In Figure 1a–c, it is clear that more can be done to guide the vehicles towards fulfilling their task. In these figures, it could be argued that the selection of node 5 (while ignoring the road capacity condition in the network), which might occur through the random selection of seeds that occurs in Algorithm 5, may not be the best choice. Instead, node 1 could be the better choice as it would lead to serving node S9. In the TBIH-4 (Algorithm 7), the potential seeds j are reserved only for those which are either one of the depots, emergency shelters, or neighbours of either. In Figure 1a–c, node 1 will be chosen instead as v since this node would lead to serving shelter S9. Only node 1 could be inserted as only a one-time insertion procedure in DLASIH is performed at every decision point in the lookahead during the construction of the route. For the shelter or depot which is farther than a one-step lookahead, as seen in Figure 4a–c, the neighbour of either that shelter or depot is then chosen as the seed j. In this case where the vehicle is packed with delivery supplies, node 9 shall be chosen as the seed (j) since it is the neighbour of shelter S7. Since only node 1 can connect to node 9 from node 0 in the one-time insertion, node 1 is then regarded as the next destination v. Here, no SIH(I1) computation is necessary as the option is rather obvious. In this illustration case, recognising node 9 as the neighbour of emergency shelter S7 helped in trimming down potential seeds to be considered and thus also reduced the number of potential v.

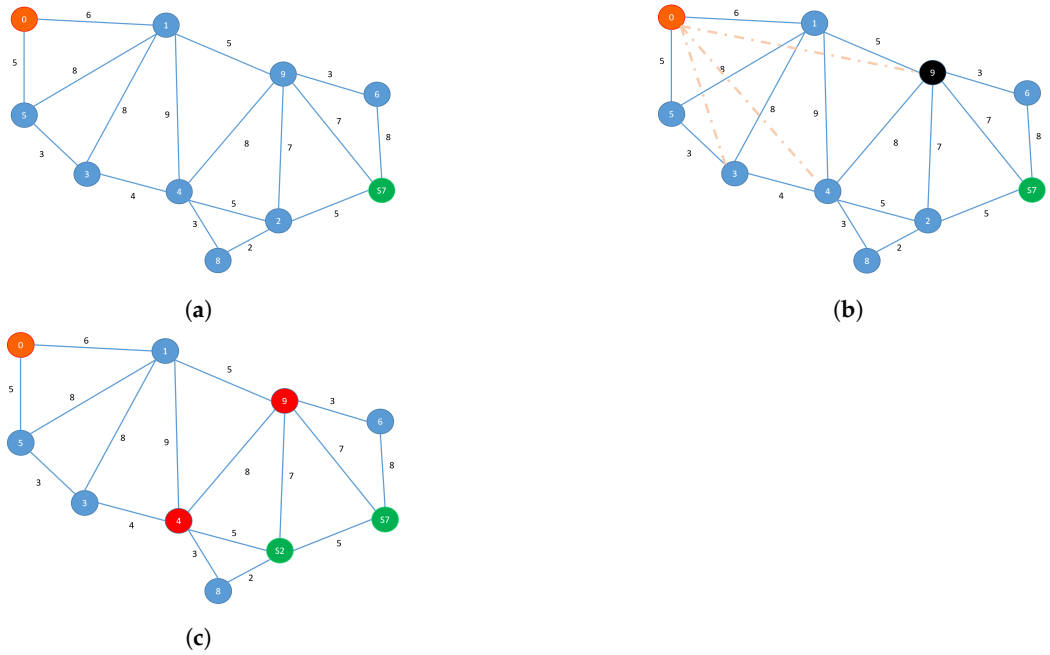


Figure 4. Performing the DLASIH in the TBIH-4 in an example network (**a**) with node 0 as current position of vehicle m ready to serve and node S7 as an emergency shelter. The potential seeds j are considered in (**b**) and Node 9 was selected since it is the neighbour of shelter S7 in (**c**). As a result, node 1 (v) is inserted and route (0–1–9–S7) is constructed. (**c**) shows two potential seeds j (node 4 and node 9) in which case a random selection between node 4 and node 9 as a seed is done.

However, if there are more potential v leading to the neighbours of emergency shelters, then one of these neighbours will be chosen randomly. If more than one possible v is connected to the chosen seed (j), the SIH(I1) will be executed.

In the TBIH-4 algorithm (Algorithm 7), the selection for j is restricted to those that would lead to either a shelter or depot, depending on q_m (lines 5 and 55). However, if such nodes are not available, another lookahead is performed to see if there are potential destinations j that could lead to neighbours of either a depot or shelter (lines 21 and 71). Depending on the case considered, the numbers of possible j from which the seed for the SIH could be

chosen from (lines 9, 37, 59, and 94) can be reduced. For either case, the insertion criteria are computed and evaluated such that the on-the-go lookahead route $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ is updated: $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$, and the decision $a_m = \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ are returned (line 98 onwards).

Algorithm 7 TBIH-4 (with an embedded DLASIH) Algorithm.

Require: s_k, M'_k, $M \backslash M'_k$, current position of vehicle
Ensure: *Decision*

1: Perform Algorithm 4 (line 1–120)
2: **if** len($nextND$)> 1 **then**
3: $\quad dict : v \leftarrow \{j : \forall j \in H, (v,j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
4: $\quad$ **if** $q_m \neq 0$ AND len(US)!= 0 **then**
5: $\quad\quad dictA : v \leftarrow \{j : \forall j \in US, (v,j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
6: $\quad\quad$ **if** len($dictA$)!= 0 **then**
7: $\quad\quad\quad newnextNDS = \{j : \forall j \in dictA(v), \forall v \in nextND, dictA(v) \neq \{\}$
8: $\quad\quad\quad$ remove duplicate node: list(set($newnextNDS$))
9: $\quad\quad\quad seed$ = random.choice($newnextNDS$)
10: $\quad\quad\quad$ list of potential inserted nodes based on selected $seed$, $InsertList = \{v : \forall v \in nextND, (v, seed) \in E\}$
11: $\quad\quad\quad dictC1 : (i, v, seed) \leftarrow (c_{i,v} + c_{v,seed} - c_{i,seed}), \quad \forall v \in InsertList$
12: $\quad\quad\quad dictC2 : (i, v, seed) \leftarrow (\lambda c_{i,seed} - dictC1(i, v, seed)), \quad \forall v \in InsertList$
13: $\quad\quad\quad$ choose v from $(i, v, seed) = \underset{(i,v,seed) \in dictC2}{\arg\max} \; (dictC2)$
14: $\quad\quad\quad a_m = v$
15: $\quad\quad\quad Decision$.append(a_m)
16: $\quad\quad\quad r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
17: $\quad\quad\quad$ break
18: $\quad\quad$ **else**
19: $\quad\quad\quad$ List of shelter's neighbours, $USN = \{n : n \in H, (s,n) \in E, \forall s \in US, n \neq i\}$
20: $\quad\quad\quad USN$ = list(set(USN))
21: $\quad\quad\quad dictB : v \leftarrow \{j : \forall j \in USN, (v,j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
22: $\quad\quad\quad$ **if** len($dictB$)!=0 **then**
23: $\quad\quad\quad\quad LNS = \{j : \forall j \in dictB(v), \forall v \in nextND, dictB(v) \neq \{\}$
24: $\quad\quad\quad\quad LNS$ = list(set(LNS))
25: $\quad\quad\quad$ **else**
26: $\quad\quad\quad\quad$ continue
27: $\quad\quad\quad$ **end if**
28: $\quad\quad\quad$ init $nextNDS$
29: $\quad\quad\quad$ **if** len(LNS)== 0 **then**
30: $\quad\quad\quad\quad nextNDS = \{j : \forall j \in dict(v), \forall v \in nextND, dict(v) \neq \{\}$
31: $\quad\quad\quad$ **else**
32: $\quad\quad\quad\quad nextNDS$.extend($LNS$)
33: $\quad\quad\quad$ **end if**
34: $\quad\quad\quad$ **if** len($nextNDS$)==0 **then**
35: $\quad\quad\quad\quad a_m$ = random.choice($nextND$)
36: $\quad\quad\quad\quad Decision$.append($a_m$)
37: $\quad\quad\quad\quad r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
38: $\quad\quad\quad\quad$ break
39: $\quad\quad\quad$ **else**
40: $\quad\quad\quad\quad$ pass
41: $\quad\quad\quad$ **end if**

Algorithm 7 *Cont.*

seed = random.choice($nextNDS$)
list of potential inserted nodes based on selected $seed$, $InsertList = \{v : \forall v \in nextND, (v, seed) \in E\}$
$dictC1 : (i, v, seed) \leftarrow (c_{i,v} + c_{v,seed} - c_{i,seed}), \quad \forall v \in InsertList$
$dictC2 : (i, v, seed) \leftarrow (\lambda c_{i,seed} - dictC1(i, v, seed)), \quad \forall v \in InsertList$
choose v from $(i, v, seed) = \underset{(i,v,seed) \in dictC2}{\arg\max} \ (dictC2)$
$a_m = v$
$Decision$.append(a_m)
$r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
break
end if
else
$dictC : v \leftarrow \{j : \forall j \in D, (v, j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
if len($dictC$)!= 0 **then**
$newnextNDS = \{j : \forall j \in dictC(v), \forall v \in nextND, dictC(v) \neq \{\}$
remove duplicate node: list(set($newnextNDS$))
seed = random.choice($newnextNDS$)
list of potential inserted nodes based on selected $seed$, $InsertList = \{v : \forall v \in nextND, (v, seed) \in E\}$
$dictC1 : (i, v, seed) \leftarrow (c_{i,v} + c_{v,seed} - c_{i,seed}), \quad \forall v \in InsertList$
$dictC2 : (i, v, seed) \leftarrow (\lambda c_{i,seed} - dictC1(i, v, seed)), \quad \forall v \in InsertList$
choose v from $(i, v, seed) = \underset{(i,v,seed) \in dictC2}{\arg\max} \ (dictC2)$
$a_m = v$
$Decision$.append(a_m)
$r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
break
else
List of Depot's neighbours, $UDN = \{n : n \in H, (d, n) \in E, \forall d \in D, n \neq i, n \notin D\}$
UDN = list(set(UDN))
$dictD : v \leftarrow \{j : \forall j \in UDN, (v, j) \in E, r_{v,j} > 0, j \neq i, j \notin nextND\} \quad \forall v \in nextND$
if len($dictD$)!=0 **then**
$LND = \{j : \forall j \in dictD(v), \forall v \in nextND, dictD(v) \neq \{\}$
LND = list(set(LND))
else
continue
end if
init $nextNDD$
if len(LND)== 0 **then**
$nextNDD = \{j : \forall j \in dict(v), \forall v \in nextND, dict(v) \neq \{\}$
else
$nextNDD$.extend(LND)
end if

Algorithm 7 *Cont.*

if len($nextNDD$)==0 **then**
  a_m = random.choice($nextND$)
  $Decision$.append(a_m)
  $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
  break
else
  pass
end if
$seed$ = random.choice($nextNDD$)
list of potential inserted nodes based on selected $seed$, $InsertList = \{v : \forall v \in nextND, (v, seed) \in E\}$
$dict\text{C}1 : (i, v, seed) \leftarrow (c_{i,v} + c_{v,seed} - c_{i,seed}), \quad \forall v \in InsertList$
$dict\text{C}2 : (i, v, seed) \leftarrow (\lambda c_{i,seed} - dict\text{C}1(i, v, seed)), \quad \forall v \in InsertList$
choose v from $(i, v, seed) = \underset{(i,v,seed) \in dict\text{C}2}{\arg\max} \; (dict\text{C}2)$
$a_m = v$
$Decision$.append(a_m)
$r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
break
end if
end if
else
  $Decision$.extend($nextND$)
  break
end if
continue Algorithm 4 (148–156)
return $Decision$

4.6. TBIH-5

In the previous section (Section 4.4), an example network (Figure 3a) is used to demonstrate how the DCW could be applied in constructing a temporary route $\eta^{\pi_{\mathcal{B}(s_k)}}$ with $\mathcal{B}$ being the heuristic from TBIH-3. From the temporary route, the first insertion is selected as the decision for the current lookahead state s_k based on the temporary route constructed: $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$. From the example, it is seen that either node 4 or 5 (Figure 3b) could be selected as the next destination since two routes could be computed from the CW heuristic (if two edges have similar highest savings). However, it is also seen in the example network that node S2 is next to node 4, while node 5 is much further than node S2. If the vehicle is with capacity, inserting node 4 for the on-the-go lookahead route $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ would make more sense.

If TBIH-3 (with an embedded DCW) could perform as a sort of lookahead (for non-obvious decisions, SP) for a nearby emergency shelter when a vehicle m has capacity, then the selection for the next destination would be more accurate. This is illustrated in Figure 5a,b. In Figure 5a, the current position of vehicle m is at node 3, and the nearby emergency shelter is node S9. With the exception of node 5, which is the neighbour of node 3, both nodes 1 and 4 are neighbours of node S9. As a result, only nodes 4 and 1 are considered when constructing $\eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ even though node 5 is also a neighbour of node 3. This leads to a more promising construction of $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ on the go. If node 5 is taken into consideration, there is a possibility of node 5 being selected as the next destination for vehicle m. This is undesirable as that would lead vehicle m, which has capacity, farther from serving S9. With this concept, the DCW is extended into DLACW (turning TBIH-3 into TBIH-5). The principle of the proposed DLACW is, for most parts, similar with the exception of a mechanism to detect a nearby shelter or depot depending on the capacity status of vehicle m.

The algorithm for TBIH-5 is presented in Algorithm 8. Similar to Algorithm 5, Algorithm 6 is extended ,resulting in a different base heuristic. When compared to Algorithm 6, some parts of this algorithm consist of detecting the potential next destination j of vehicle m that might lead to either a shelter of depot, depending on the current capacity status q_m (lines 3–9). Through this mechanism, the possible option for j is restricted to only those ideally more guided destinations. Edges are detected (line 10) and removed if they do not exist in the network (line 11), while savings are computed (line 18) and sorted (line 19). Finally the on-the-go base policy $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}}$ is updated for the lookahead state s_k, where $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ and the decision $a_m = \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ is returned.

Algorithm 8 TBIH-5 (with an embedded DLACW) algorithm.

Require: s_k, M'_k, $M \backslash M'_k$, current position of vehicle
Ensure: *Decision*

1: Perform Algorithm 4 (line 1–120)
2: **if** len($nextND$)> 1 **then**
3: | **if** $q_m \neq 0$ AND len(US)!= 0 **then**
4: | | $dict : j \leftarrow \{k : \forall k \in US, (j,k) \in E, r_{j,k} > 0, k \neq i, k \notin nextND\} \quad \forall j \in nextND$
5: | | $newnextND = \{j : \forall j \in nextND, dict(j) \neq \{\}\}$
6: | **else**
7: | | $dict : j \leftarrow \{k : \forall k \in D, (j,k) \in E, r_{j,k} > 0, k \neq i, k \notin nextND\} \quad \forall j \in nextND$
8: | | $newnextND = \{k : \forall k \in dict(j), \forall j \in nextND, dict(j) \neq \{\}\}$
9: | **end if**
10: | Possible edges from pair that could be formed in $newnextND$, $Pairs = \binom{newnextND}{2}$
11: | Remove $(j,k) \in Pairs$, if $(j,k) \notin E$
12: | **if** len($pairs == 0$) **then**
13: | | a_m = random.choice($nextND$)
14: | | $Decision$.append(a_m)
15: | | $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
16: | | break
17: | **else**
18: | | compute savings, $(j,k) :\leftarrow c_{i,j} + c_{i,k} - c_{j,k} \quad \forall (j,k) \in Pairs$
19: | | sort $(j,k) \in Pairs$ with decreasing savings
20: | | **if** len($Pairs$)==1 **then**
21: | | | $a_m = j : (j,k) \in Pairs$
22: | | **else**
23: | | | construct route $\varrho^{\pi_{\mathcal{B}(s_k^{a_k})}} : s_k \rightarrow \eta^{\pi_{\mathcal{B}(s_k)}}(s_k)$ from $i = l_m$ by inserting $(j,k) \in Pairs$ as would be done in CW (decreasing savings)
24: | | | $a_m \leftarrow A^{\pi_{\mathcal{B}(s_k^{a_k})}}(s_k)$
25: | | **end if**
26: | | $Decision$.append(a_m)
27: | | $r_{i,a_m} = max(r_{i,a_m} - 1, 0)$
28: | | break
29: | **end if**
30: **else**
31: | $Decision$.extend($nextND$)
32: | break
33: **end if**
34: continue Algorithm 4 (148–156)
return *Decision*

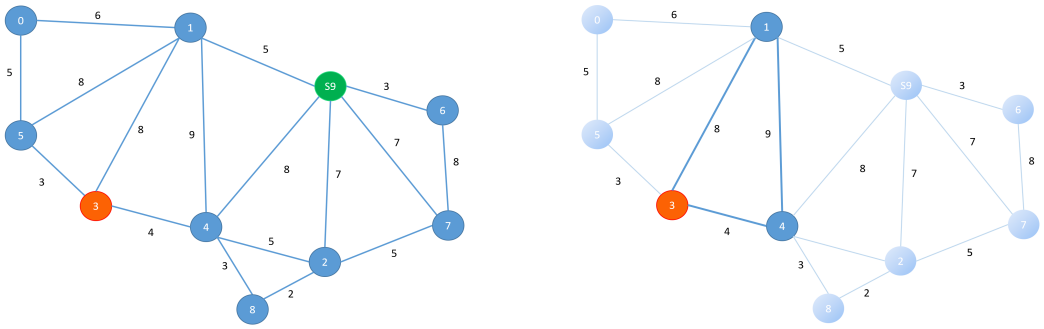


Figure 5. Performing the DLACW in TBIH-5 in an example network (**a**) with node 3 as the current position of vehicle m and node S9 as an emergency shelter. The components for performing the CW are selected in (**b**).

5. Computational Results

This study presents computational results for the following purposes:

- To validate the model MDDVRPSRC by observing through the simulation tool the ecosystem (emergency medical supplies delivery) simulated.
- To validate the reinforcement learning solution of the agent in conducting the medical delivery operations through decisions computed based on the ADP approach (PDS–RA with 5 proposed base heuristics).
- To study the quality of the learning solution through the resulting simulated data by means of a comparative approach against the matheuristic proposed in the work of [15] in the stochastic setting of road capacity and dynamic road damage.
- To extend the findings in the work of [15] which serves as a preliminary study for this research.

The experiment is conducted using the authors' developed MDDVRPSRC Decision Support System (DSS) program (Figure 6) with codes written in Python 2.7 programming language. Embedded in this program is also a network monitoring layout through which the simulation of medical supplies delivery operations in the setting of the MDDVRPSRC is observed in real time (live simulation). Both the MDDVRPSRC model and the computation of the agent's decision are also implemented at the heart of the DSS. As part of the computation of the agent's decision, this program also executes the matheuristic (upon selection) proposed in work of [15] with CPLEX computation executed through the DOCPLEX API for Python. For the experiment, the MDDVRPSRC DSS is run on a laptop computer running on Intel^{R} Core^{TM} i7-7500U CPU at 2.70–2.90 GHz with 20 GB RAM.

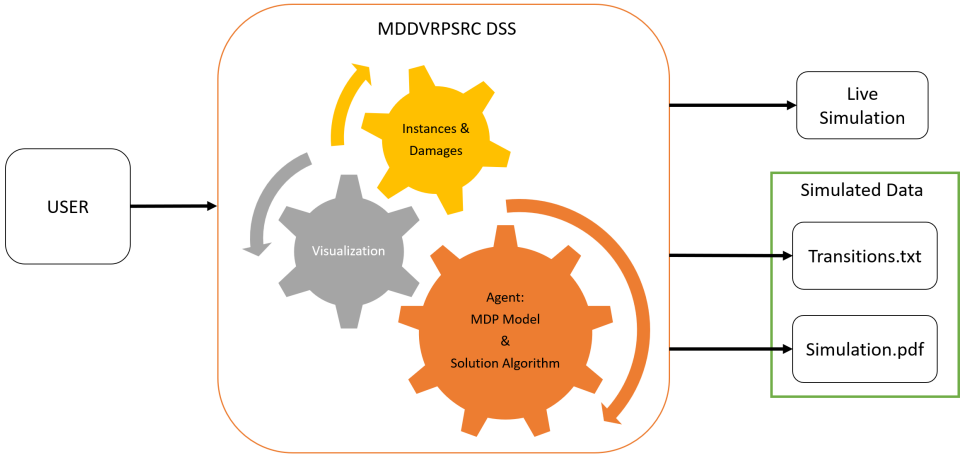


Figure 6. MDDVRPSRC DSS for medical delivery operation.

To test and validate the model and solutions algorithm for such a unique problem as the medical supplies delivery operations with compromising circumstances, the common benchmarks of Perl, Gaskell, and Christofides cannot be applied. So several test instances are designed [86,87], ranging from a small and simple road network, to medium and more challenging networks based on the available experimental apparatus (Figures A1–A10). Along with each test instance are the associated initial damage for each road within the network of the respective test instances [86,87]. These datasets, consisting of test instances and the associated damage files are available in [87] and are shown in Table 3. In Table 3, the test instances are ordered by increasing complexity levels, characterised in terms of total number of nodes as well as the ratio between connecting node to a depot and emergency shelter. The test instances associated with the road network D4N30S10 are placed last as it is hypothesised as the most challenging network among the test instances listed. Each network is comprised of multi-depots, multi emergency shelters with different demands and connecting nodes as described in Section 3.1. The placement of the nodes is based on the lessons learned from the 2015 Nepal earthquake.

In each of the road networks seen in Figures A1–A10, the blue, yellow, and brown nodes each represent the depots, emergency shelters, and connecting nodes, respectively. The violet circle lines represent the outward tremor that originated from an earthquake epicentre (coordinate (460, 180) in all the networks). The degree of the initial road damage is based on the intersection of these circles onto the edges, and the corresponding random road capacity is denoted in red at the center of the edges. The demands of each emergency shelter hovers above in a pink box. The green boxes represent vehicles that have arrived at the nodes where they are currently stationed. The blue boxes represent vehicles en route to each of their next assigned destinations. In Figure A11, the simulation example of an ongoing medical supplies operation is shown. In the road network D8N20S8, five vehicles are assigned to deliver medical supplies to eight emergency shelters with their respective demands. The full road capacity in this network for a city road, normal road, and highway is given in Table 3 as $(6, 7, 8)$, respectively. In all road networks, the highways are placed at the outer part of the network, while the city roads are placed at the innermost sections of the networks. Normal roads can be found connecting highways with city roads in most cases, especially in the larger networks. At decision point 104, which is at the simulated time of 3097 min (translated as 2:3:37:00), the road capacity for each road changes randomly based on the dynamic road capacity mean for the random distribution of each road. These dynamic deteriorating road capacities in turn depend on the initial damage sustained by the road (given in the damage file of each test instance in the repository [87]). Thus the road capacity for the edges with more interceptions with the radial earthquake tremor circles are seen with a tendency to have less road capacity at random when compared with the edges with less or zero intersections. Hence, vehicles travelling at these edges will suffer longer travel times proportional to the initial damage sustained by the edges as accounted for in the MDDVRPSRC model described in Section 3.3. The work [15] is referred to for more explanation on how the random road capacity is sampled at each decision point. The experiment settings for both the simulation and computation of the agent's decision (PDS-RA) is given in Table 4.

For the model and solution validation, simulated data is compiled (Figure 6). For each proposed base heuristics (TBIH-1, TBIH-2, TBIH-3, TBIH-4, and TBIH-5) applied for all test instances in Table 3, 10 complete simulations of a medical supplies delivery operation are performed. Out of the 10 complete simulations, there are 10 readings for 4 key measurements:

1. Total travelled distance (K1);
2. Total travelled time (K2);
3. Total computation time (K3);
4. Average decision computation time (K4).

Table 3. Simulated test instances applied to validate the model and solutions algorithm.

Instance	Depot	Shelter	Nodes	Vehicle	Total Demand	Max Road Capacity
D3N8S3_4	3	3	8	4	550	6,7,8
D3N8S3_8	3	3	8	8	550	6,7,8
D3N8S3_15	3	3	8	15	550	6,7,8
D3N8S3_30	3	3	8	30	550	6,7,8
D3N8S3_50	3	3	8	50	550	6,7,8
D4N11S4_4	4	4	11	4	550	6,7,8
D4N11S4_8	4	4	11	8	550	6,7,8
D4N11S4_15	4	4	11	15	550	6,7,8
D4N11S4_30	4	4	11	30	550	6,7,8
D4N11S4_50	4	4	11	50	550	6,7,8
D5N13S5_4	5	5	13	4	650	6,7,8
D5N13S5_8	5	5	13	8	650	6,7,8
D5N13S5_15	5	5	13	15	650	6,7,8
D5N13S5_30	5	5	13	30	650	6,7,8
D5N13S5_50	5	5	13	50	650	6,7,8
D6N16S6_4	3	3	8	4	950	6,7,8
D6N16S6_8	3	3	8	8	950	6,7,8
D6N16S6_15	3	3	8	15	950	6,7,8
D6N16S6_30	3	3	8	30	950	6,7,8
D6N16S6_50	3	3	8	50	950	6,7,8
D7N18S7_4	7	7	18	4	1250	6,7,8
D7N18S7_8	7	7	18	8	1250	6,7,8
D7N18S7_15	7	7	18	15	1250	6,7,8
D7N18S7_30	7	7	18	30	1250	6,7,8
D7N18S7_50	7	7	18	50	1250	6,7,8
D8N20S8_4	8	8	20	4	1350	6,7,8
D8N20S8_8	8	8	20	8	1350	6,7,8
D8N20S8_15	8	8	20	15	1350	6,7,8
D8N20S8_30	8	8	20	30	1350	6,7,8
D8N20S8_50	8	8	20	50	1350	6,7,8
D8N22S9_4	8	9	22	4	1600	6,7,8
D8N22S9_8	8	9	22	8	1600	6,7,8
D8N22S9_15	8	9	22	15	1600	6,7,8
D8N22S9_30	8	9	22	30	1600	6,7,8
D8N22S9_50	8	9	22	50	1600	6,7,8
D9N25S10_4	9	10	25	4	1650	6,7,8
D9N25S10_8	9	10	25	8	1650	6,7,8
D9N25S10_15	9	10	25	15	1650	6,7,8
D9N25S10_30	9	10	25	30	1650	6,7,8
D9N25S10_50	9	10	25	50	1650	6,7,8
D9N30S10_4	9	10	30	4	1650	6,7,8
D9N30S10_8	9	10	30	8	1650	6,7,8
D9N30S10_15	9	10	30	15	1650	6,7,8
D9N30S10_30	9	10	30	30	1650	6,7,8
D9N30S10_50	9	10	30	50	1650	6,7,8
D4N30S10_4	4	10	30	4	1650	6,7,8
D4N30S10_8	4	10	30	8	1650	6,7,8
D4N30S10_15	4	10	30	15	1650	6,7,8
D4N30S10_30	4	10	30	30	1650	6,7,8
D4N30S10_50	4	10	30	50	1650	6,7,8

The first three key measurements are self explanatory. The last key measurement is the average time taken for the agent to make one decision at decision point k based on the total computation time divided by the number of decisions made (decision points) to complete the delivery operations simulation. The PDS-RA with the proposed heuristic bases are benchmarked with the matheuristic rollout found in the work of [15] for all vehicle number settings (4, 8, 15, 30, and 50) for the road networks D3N8S8-D7N18S7. For

the road network D8N20S8-D4N30S10, however, the benchmarking is completed only up to the vehicle number settings of 4, 8, and 15. This is due to the resultant computational time which is far longer than considered reasonable when compared with the longest computation time obtained among the five proposed heuristics (Figures A14 and A18). With the resulting simulated data, the model is then validated based on the analysis of the output data produced (Figure 6). Furthermore, the performance of the proposed heuristics compared to the matheuristic rollout applied is observed through a descriptive and comparative analysis.

Table 4. Simulation and PDS-RA Configuration.

Parameter	Value
Deterioration Proportional Constant P	0.1
Ω	200
Vehicle Speed	90 km/h
Vehicle Capacity, Q	50
Monte Carlo Simulation	3
Lookahead Horizon K	7

The computational results in the Supplementary file (Tables S1–S81) are collected and recorded for a time span of more than two years; given the hardware available for the experiments. A total of 10 readings were taken for each of the proposed base heuristics applied in the PDS-RA for all test instances. This was for all key measurements (K1–K4) given the stochastic road capacity and dynamic deterioration of the mean road capacity in the problem addressed. From each 10 readings, the descriptive analysis is performed to measure the mean, standard deviation, variance, and covariance of the sample data. The Normality test is performed to determine that a suitable comparative analysis method is applied for benchmarking. A total of 11,600 key readings were recorded as a result of 2900 simulations performed for further analysis involving the key measurements of K1, K2, K3, and K4 mentioned in Section 5. The 2900 simulations consist of 290 sets of 10 readings per set, for each of the 4 key measurements which are then used to compute the average reading. Not all 290 sets tested were found to have a normal distribution based on the Shapiro–Wilk test [88] performed in the Excel [89]. The highest percentage for normal data (around 50%) is only seen in the K3 and K4 measurements. Furthermore, the 10 readings for each key measurement of a test instance is considered small for a parametric test. As such, a non-parametric test (Wilcoxon Signed Rank Test) was applied to test for significance in differences against the matheuristic solution (PDS-RA with CPLEX as base heuristic). Moreover, the Best So Far (BSF) measurement among the solution algorithms applied at each test instance was performed to observe the performance of each PDS-RA of the respective proposed heuristics against the matheuristic rollout.

The full computation results are presented in the supplementary file and the abbreviations applied are listed in Table 5. Furthermore, the general overview of the simulated data collected is shown in Table A1. Thorough investigation and synthesise of the resulting simulation data by means of cross-referencing key values were performed to ensure that no errors are presented.

The results obtained in Tables S1–S81 are further synthesised for numerical analysis focusing on model validation and base heuristics performances. The MDDVRPSRC model is validated based on the trends and patterns observed in Figures A12, A13, A16 and A17. Meanwhile, the performance of the proposed heuristics, as compared to the matheuristic rollout, can be seen in the remaining figures between Figures A12–A27 and in the supplementary file S1 (Figures S1–S62). Figures A12–A15 show the trends for average measurements of each of the 10 sample readings based on all four key measurements, while Figures A16–A19 shows the trends for best measurements among the 10 reading samples for each key measurement. Figures A20–A23 show the total numbers of BSF counts for each algorithm for all 40 instances with the matheuristic rollout benchmark. Figures A24–A27

depict the total numbers of BSF counts in percentage for each algorithm for all 50 instances with and without (omitting 10 test instances for matheuristic rollout due to computation time limitation) the matheuristic rollout benchmark. A more detailed breakdown per test instance of the percentage of BSF associated with each heuristics is shown in Figures S1–S40. Meanwhile, Tables A2–A7 give a more detailed breakdown on numbers of the BSF counts, normal distribution data, and the significant differences for each key measurements. Finally, a detailed performance of each PDS-RA with proposed heuristics for all key measurements is shown in Figures S41–S62.

Table 5. Abbreviation for Tables and Figures.

Base-H	Base Heuristic applied during rollout lookahead
TBIH-1	Teach Based Insertion Heuristic
TBIH-2	TBIH with dynamic SIH
TBIH-3	TBIH with dynamic CW
TBIH-4	TBIH with dynamic lookahead SIH
TBIH-5	TBIH with dynamic lookahead CW
CPLEX	DOCPLEX (Python): solving MDVRPSRC-2S1 and MDVRPSRC-2S2
SW(P)	P value: Shapiro Wilk Test for normality test
Wilcox(P)	P value: Wilcoxon Signed Rank Test for Significance test
BSF	Best So Far Measurement Value
Sig.	Significance
N	Normal
(% V2)	Percentage Performance based on 40 Measurements instead of 50 (CPLEX Application as Base Heuristic for benchmarking)

6. Discussion

In terms of the MDDVRPSRC MDP model validation, the behaviours plotted in Figures A12, A13, A16 and A17 are conforming to the natural expectation on how the humanitarian operational aspects will shape out based on the key measurements. Figures A12 and A16, for example, show a logical increase of total distance travelled with the increase in the number of vehicles. Here, the increase in total distance is also attributed to the policy that all vehicles must be dispatched for delivery to compensate for the potential risk that a vehicle might be stranded while en route due to the road damage incurred. Furthermore, a stochastic road capacity with multiple dispatches of vehicles might ensure a faster delivery time at the cost of an increase in total distance travelled.

For the road network D3N8S3, the increase of total distance is higher than that of networks D4N11S4, D5N13S5, and D6N16S6. This is comparable to that of network D7N18S7 onwards with operations involving 30 and 50 vehicles. This is due to the large amount of vehicles travelling on a road network with limited roads. The random road capacity as well as deteriorating road conditions cause a bottleneck at some connecting nodes. However, a steady increase of roads in more complex networks alleviates this problem, as shown in networks D4N11S4, D5N13S5, and D6N16S6. Given the increasing demands and more complex networks, a different observation could be made.

The road networks of D7N18S7, D8N20S8, D9N25S9, and D9N30S10, for example, indicate roughly the same trend of total distance travelled with an occasional peak at about 10,000 km for networks D8N20S8 and D9N25S10. However, an obvious increase of total distance travelled can be seen for the network D4N30S10, thereby confirming the hypothesis that this network is the most complex in terms of delivery operations. This is explained by the ratio of depots to connecting nodes where the vehicle has only a limited number of depots to replenish supplies in this network as compared to the other networks. Furthermore, the ratio of depots and shelters also contributes to this observation, showing the difficulties of completing the deliveries given the smaller number of depots to replenish.

Additionally, the reduced number of depots in this network also leads to more connecting options between the depots and connecting nodes which may not necessarily be advantageous to the delivery operations. This is especially true for networks that tend to

have a shorter route disabled due to random road capacity and a dynamic reduction of the road capacity due to damage to the road. As a result, a longer route is taken leading to the increase of total distance travel by all vehicles.

All of the observations for the total distance travelled shown in Figures A12 and A16 also apply to the observations seen in Figures A13 and A17. In general, the increase of the total vehicle numbers leads to the reduced delivery operations time (total travel time). For the network D3N8S3, a limited number of vehicles in a small network with high demands relative to the number of vehicles led to an increase in total travel time compared to network D4N11S4. This is due to the longer time required by the smaller number of vehicles to satisfy the total demands within the network. Moreover, the deteriorating road capacity for each damaged road may lead to lesser road availability for an already small road network. This leads to vehicles taking the longer route compared to that of network D4N11S4.

Vehicles may also travel back and forth along the same road due to connecting roads becoming increasingly less available. The bottleneck effect is also seen for the larger number of vehicles when comparing the total time travel within the road network of D3N8S3 with the road networks of D4N11S4, D5N13S5, and D6N16S6. It is also shown clearer here that the bottleneck effect could be alleviated through trends observed for networks D7N18S7, D8N20S8, D9N25S9, and D9N30S10. Similarly the reduced ratio between depots to shelters and depots to connecting nodes leads to a more complex network. This is despite not having the highest number of nodes that contributes to a higher total travel time for some of the algorithms. Interestingly, the matheuristic rollout approach does not show the same observations. This shows the potential of the matheuristic rollout in navigating more complex networks compared to the proposed heuristics.

This, however, comes at the cost of computation time as shown in Figures A14, A15, A18, A19, A22, A23, A26 and A27. This was observed when investigating the performance of the proposed base heuristics against the matheuristic rollout as a benchmark. The total computation time increases for the matheuristic rollout applying CPLEX at every lookahead decision point for road network D5N13S5 onwards for all vehicle settings (Figures A14 and A18) when compared with the results obtained with PDS-RA applying the proposed base heuristics. This trend is even more obvious in Figures A15 and A19, showing a clear increase in computation time for the agent in making a decision on average. As a result, no BSF count was ever obtained through the matheuristic rollout for the key measurement of K3 and K4 (Figures A22, A23, A26 and A27).

Apart from showing an exponential increase for both K3 and K4 (see Figures A15 and A19), it is also obvious that this trend depends on the total number of nodes that are involved in the network. This is evident when comparing the two key measurements for networks D9N30S10 and D4N30S10. However, the road networks sharing a similar number of nodes as D4N30S10, such as D9N25S10, do not indicate a similar magnitude of increment. Therefore, it could be concluded that both the number of nodes and complexity associated with each network affect the two key measurements for the matheuristic rollout.

Meanwhile, the performance of the proposed PDS-RA applying base heuristics is further investigated through the BSF count for all instances tested. Figures A22, A23, A26 and A27 confirm the observation made for the matheuristic rollout in terms of computation time (K3 and K4). However. the matheuristic rollout shows clear dominance in terms of the key measurements of K1 and K2 (Figures A20, A21, A24 and A25). This is especially seen in the breakdown of K1 and K2 in Figures S46 and S52 which Figure S46 interestingly also show a good performance of TBIH-1 for K1. This shows the relevance of the matheuristic approach for complex stochastic problems. In most of the individual networks, the matheuristic approach seems to also perform better compared to the other proposed approaches for K1 and K2 (Figures S46 and S52). However, as it can be seen in Figures A12–A17 with the exception of Figures A14 and A15, the application of PDS-RA with proposed heuristics remains competitive with low gaps of difference. This is also supported by the statistical numerical evidence that show lower significance differences

recorded throughout all simulated data involving K1 and K2 when compared with data obtained by the matheuristic rollout (Table A1).

Of those, a vast majority of significance difference is seen in Figures A14, A15, A18 and A19 which corroborates findings in terms of computation time for K3 and K4. Judging from the trends, the practicality of the matheuristic rollout as benchmarked, is shown to be poor at least for the given hardware used for the experimentation. This is despite the good performance shown for K1 and K2, albeit with no significance difference.

On the other hand, the TBIH-1 shows clear advantage as shown from Figures A20–A27 in terms of the BSF count. This is perhaps expected considering the stochastic problem which may favour the exploratory approach more than the exploitation part, as is performed in TBIH-1 with random selection for the SP part of the algorithm. The comparable performance of PDS-RA with TBIH-1 compared to the matheuristic rollout is also shown in most of the road networks, respectively. Furthermore, TBIH-1 is seen at times neck to neck with the benchmark when looking into the performance in each individual network, such as in Figures S26, S29, S33, S34, and S36 among others. It is also noteworthy to see that the algorithm also performs rather well for the network D4N30S10, with the exception of key measurement K2. Moreover, the dominance of the TBIH-1 is increasingly more noticeable for larger networks as best seen in Figures S41, S47, S53, and S58. Meanwhile, both the TBIH-2 and TBIH-4 also perform well in the overall BSF count (as seen in Figures A20–A27) when compared to that of TBIH-3 and TBIH-5 which is based on DCW. This highlights the advantages of the DSIH which centred on the concept of inserting and placing promising nodes in ways that optimise the operation.

DCW, on the other hand, tends to ignore the inner part of the nodes and favour the outer nodes in an attempt to reduce parallel connections to the origin node as CW has always been intended for. This is evident by the performance of TBIH-3 which is the lowest followed by TBIH-2 when looking at BSF counts for both the individual network and overall networks (Figures S41–S62). Except for K3 in Figure S55, the TBIH-3 only scores a BSF count of one for all other key measurements (Figures S43, S49 and S60). This translates in a low BSF count obtained in Figures A20–A27 where the TBIH-3 is seen multiple times with BSF counts as low as 0% and 2.5% while topping at most only an 8% as seen in Figure A26. TBIH-5 shows an improved performance when compared to TBIH-3 (for K1, K2, K3, and K4 in Figures S45, S51, and S57) and TBIH-2 (except for K3 and K4) with the addition of a lookahead mechanism for selecting more promising nodes in the route. This demonstrates the strength of exploitation in the heuristics to improve selection. The TBIH-2, however, is better in terms of K3 and K4 (Figures S54 and S59), displaying the trade off for embedding such features.

Similarly the TBIH-2's performance is improved in TBIH-4 by means of an exploitation mechanism that requires a lookahead in selecting more promising nodes and filtering out those that are not. Unlike the TBIH-3 and TBIH-5, however, the gap in the computation speed between the TBIH-2 and TBIH-4 is not obvious. This shows that the TBIH-5 might be more costly to implement compared to TBIH-4 which improves on the TBIH-2 with less trade-off as seen in Figures S54, S56, S59 and S61. As such, the TBIH-4 could be considered an all-rounder with a balanced performance next to TBIH-1.

It should be noted that the PDS–RA is performed per vehicle when making a collective decision for all vehicles. Furthermore, both the number of Monte Carlo simulations and the length of the lookahead horizon shown in Table 4 could be considered low when compared to other similar work. However, the new perspective of the computing decision, as proposed in Equation (24), demands some compromise be made, especially with limited computational power available for this research. Furthermore, the method applied in this research is necessary to break the usual practice of clustering the emergency hot-spots per vehicle and then computing the routing decision afterwards. Additionally, the research for stochastic road capacity problems with additional consideration for damaged roads is very limited among reinforcement-learning-oriented research. Due to the stochastic road capacity, the resulting key measurements are highly varied as shown in the variance and

covariance measurement of each of the simulated samples collected (Tables S2–S81). Ideally, a good amount of Monte Carlo simulations of the rollout and a longer horizon for the lookahead would be best to account for such stochastic problems. A trade-off still needs to be made where the limitation of computation is a concern. If anything, this research proves that the proposed methods could be applied to a machine with limited capability to simulate, visualise, and compute decisions as a DSS for an emergency medical supplies delivery humanitarian operation.

However, more study into this research is warranted. With capable machines, the number of lookahead horizons and the number of Monte Carlo simulations should be increased. With such an increase in parameters, perhaps the TP of the algorithm could be discarded; allowing the agent a pure learning opportunity when making decisions. In the experiments here, this could not be achieved; hence the TP is needed. Furthermore, with enough Monte Carlo simulations, the highly stochastic problem concerning routing can be properly addressed. A longer horizon of the lookahead ensures better decisions in a long-term perspective.

The investigation included a one-factor experiment performed by varying the fixed number of vehicles per road networks, which is a limitation. It should be note,d however that various road networks were tested consisting of varying numbers of depots, emergency shelters, and connecting nodes. Furthermore, given the entry level machine that is utilized, this experiment (involving 2900 simulations and 11,600 measurement readings) took more than one year to complete. Given a more capable machine, factorial experiments should be performed to investigate the performance of the proposed heuristics againist the matheuristic benchmark. For example, through a factorial experiment, the existing network could be expanded into more challenging networks. In D4N30S10 for instance, it would also be interesting to see how the delivery operation with such a number of depots and an increase of connecting nodes fares with a smaller number of emergency shelters as more options for routing become available. Will the agent with the proposed solution method be able to navigate intelligently among these many options? Hence, more studies should be performed with expanded networks where the combination of ratios between depots, connecting nodes, and emergency shelters are varied. For this experiment, the vehicles are placed randomly at depots initially. This is performed to account for the degree of unpreparedness, where coordination should be planned with random accounts of assets. Hence, even though the key measurements are assessed through 10 average readings for each test instance, the initial situation for each simulation run is varied. There are two ways that this study could be expanded further: (1) to increase the number of simulations per test instance to obtain more than 10 readings for a better average reading, and (2) to apply a fixed assignment of vehicles per depot for all simulations. The latter approach, however would not account for a more realistic scenario of emergency medical supplies delivery operations. Finally, in this study, the placement of depots, connecting nodes, and emergency shelters are made such that the findings obtained from the lessons learned in the 2015 Nepal earthquake are addressed. Instead of utilising a simulated network, a more concrete simulation could be performed by applying real networks and incorporating details of the depots, connecting nodes, emergency shelters, vehicles, road damages, and road capacities during that actual disaster event. It is noted that such data is usually of a sensitive nature. However, developing a simulated network allows for flexibility when completing planning exercise and experiments.

7. Conclusions

As part of the DSS for humanitarian emergency medical supplies delivery operations, the 2015 Nepal earthquake is referred to in developing the MDDVRPSRC MDP model. The presented model focuses on the difficulty in navigating through stochastic road capacity within the compromised road network due to the ongoing tremors from the earthquake. The model also incorporates multi-depots, multi-trips, and split delivery operations. Here the conventional approach of "cluster first, route second" largely applied among related

research cannot be applied. Instead, to solve the problem, a lookahead approach of ADP is adopted, where the PDS-RA is applied. As part of the PDS-RA mechanism, five base constructive heuristics are proposed to construct the decision rule on the go dynamically and iteratively. Unlike conventional applications of the PDS-RA in VRP, this research adopted the proposed method in the work of [15] for a consecutive application of the PDS-RA for each vehicle that arrives at every decision point. The resulting individual assignment of vehicles computed collectively forms an MDP decision at every decision point.

The five proposed base heuristics are based on a decision-making strategy that consists of obvious decisions (TP) and non-obvious decisions (SP) to reduce the burden of computation. In the TBIH-1, the SP applied pure random selection for selecting a vehicle's next destination. Alternatively, the principle of constructive heuristics used in SIH(I1) and CW, (coined as DSIH and DCW, respectively) are adopted in the TBIH-2 and the TBIH-3. A lookahead exploitation mechanism is adapted to both the DCW and the DSIH, giving birth to DLASIH and DLACW which is applied in the proposed TBIH-4 and TBIH-5, respectively. These five proposed base heuristics are compared with the matheuristic proposed in the authors' previous work, [15]. Moreover, test instances were developed and made available in the repository [87]. The results presented in the supplementary file validate the model where expected behaviour is observed from the simulated operations based on four key measurements: K1, K2, K3, and K4. Furthermore, the performance of the PDS-RA applied with the proposed five base heuristics shows comparable performance for K1 and K2 with no significant difference recorded. Meanwhile, all the proposed heuristics showed superior performance for K3 and K4 when compared to the matheuristic. The results also highlight the power of exploration associated to pure random selection in the TBIH-1 in addressing a highly stochastic problem such as the MDDVRPSRC. Furthermore, the advantages of exploitation are shown in TBIH-4 and TBIH-5 when compared with the performance of TBIH-2 and TBIH-3, respectively. For problems such as the MDDVRPSRC, it would appear that the DSIH (TBIH-2) and DLASIH (TBIH-4) perform better than their counterparts: DCW (TBIH-3) and DLACW (TBIH-5).

Supplementary Materials: The following are available at https://www.mdpi.com/article/10.3390/math10152699/s1.

Author Contributions: Conceptualisation, W.K.A., L.S.L. and S.P.; methodology, W.K.A., L.S.L. and S.P.; software, W.K.A. and L.S.L.; validation, W.K.A. and L.S.L.; formal analysis, W.K.A., L.S.L. and H.-V.S.; investigation, W.K.A. and L.S.L.; resources, W.K.A.; data curation, W.K.A. and L.S.L.; writing—original draft preparation, W.K.A., L.S.L. and H.-V.S.; writing—review and editing, W.K.A., L.S.L. and H.-V.S.; visualisation, W.K.A.; supervision, L.S.L. and S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The primary simulated data is presented in the supplementary file. The test instances from which the simulated data is generated is found in [87].

Acknowledgments: This research was supported by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme with reference code FRGS/1/2019/STG06/UPM/02/1.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADP	Approximate Dynamic Programming
MDDVRPSRC	Multi-Depot Dynamic Vehicle Routing Problem with Stochastic Road Capacity
CSM	Supply Chain Management
MDP	Markov Decision Processes
DSS	Decision Support System
LSPs	Logistics Service Providers
ML	Machine Learning
RL	Reinforcement Learning
MIP	Mixed Integer Programming
PRE	Pre Decision State
PDS	Post Decision State
RA	Rollout Algorithm
PDS–RA	Post Decision State Rollout Algorithm
SILP	Stochastic Linear Integer Programming
SIH	Sequential Insertion Heuristic
DSIH	Dynamic Sequential Insertion Heuristic
DLASIH	Dynamic Lookahead Sequential Insertion Heuristic
CW	Clarke and Wright
DCW	Dynamic Clarke and Wright
DLACW	Dynamic Lookahead Clarke and Wright

Appendix A. Simulated Road Networks and Analysis Results

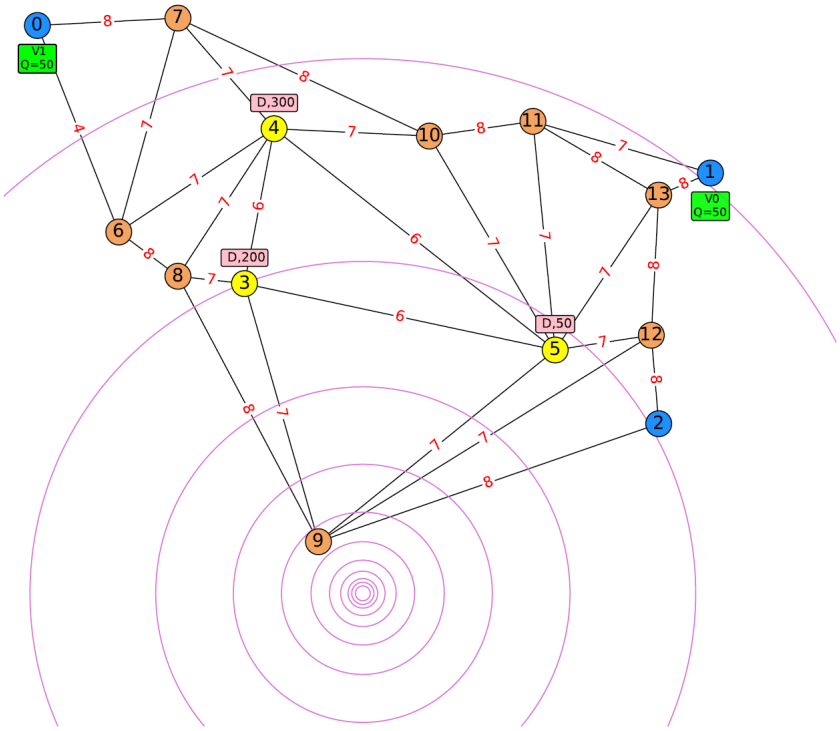


Figure A1. Road network for instance D3N8S3 [15].

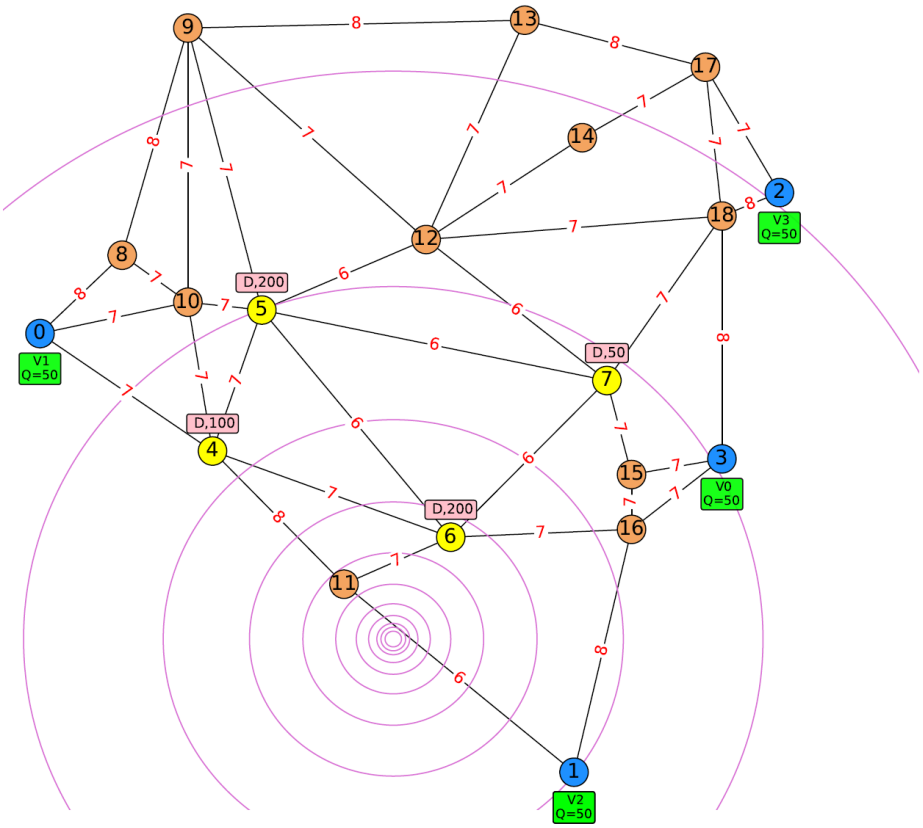


Figure A2. Road network for instance D4N11S4 [15].

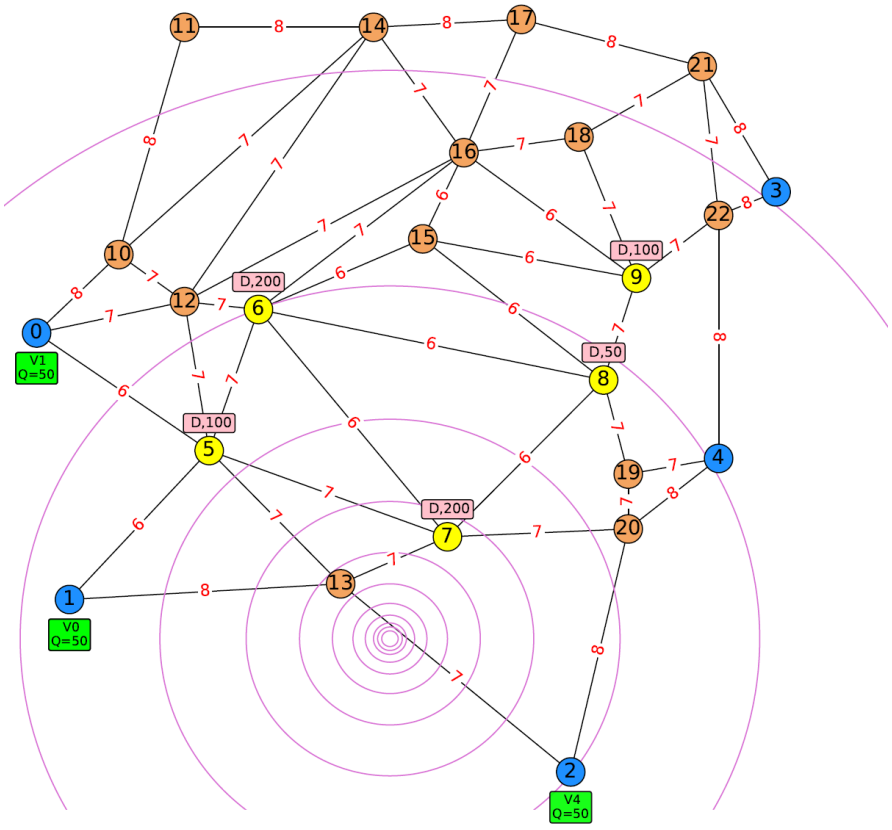


Figure A3. Road network for instance D5N13S5 [15].

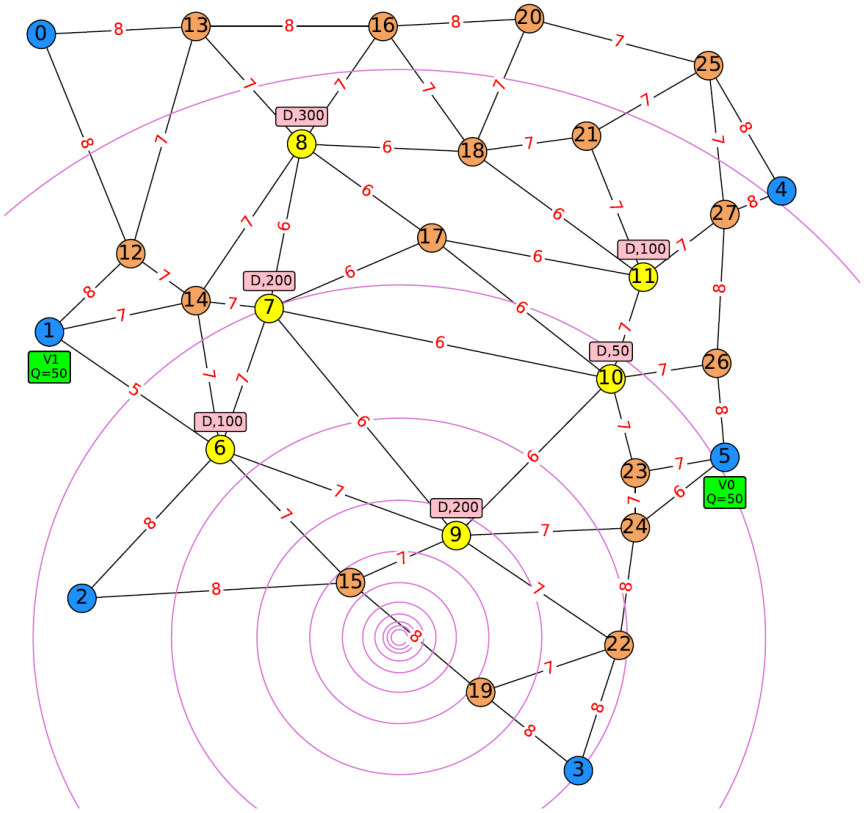


Figure A4. Road network for instance D6N16S6.

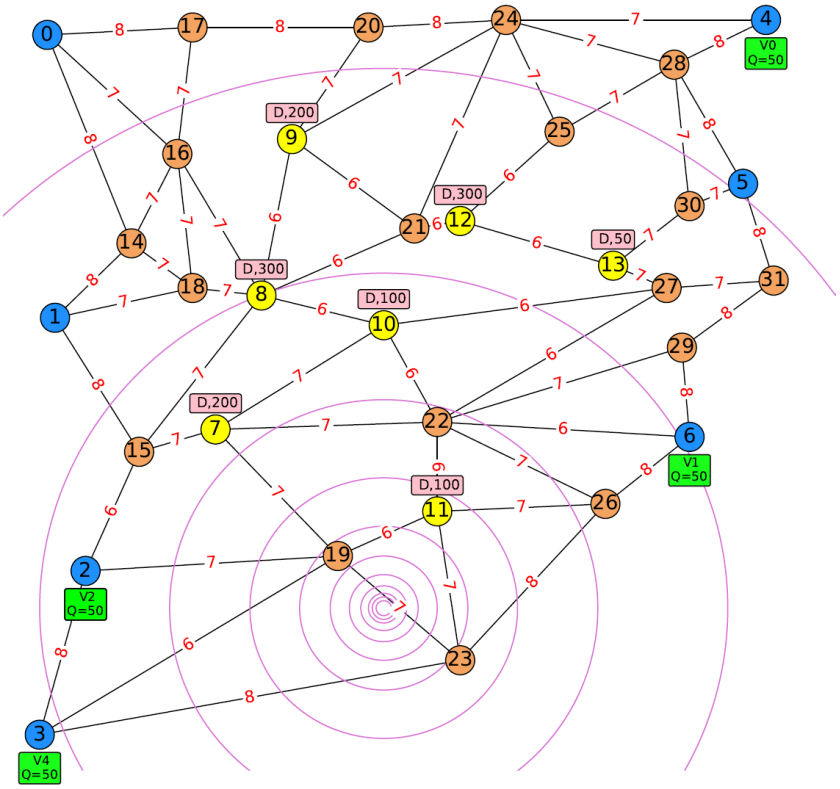


Figure A5. Road network for instance D7N18S7.

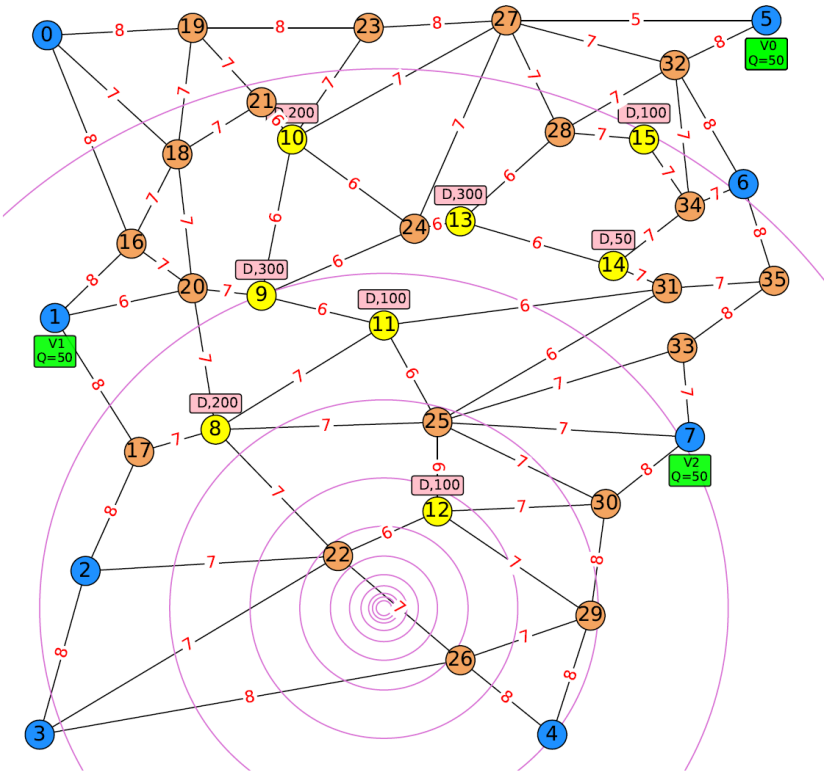


Figure A6. Road network for instance D8N20S8.

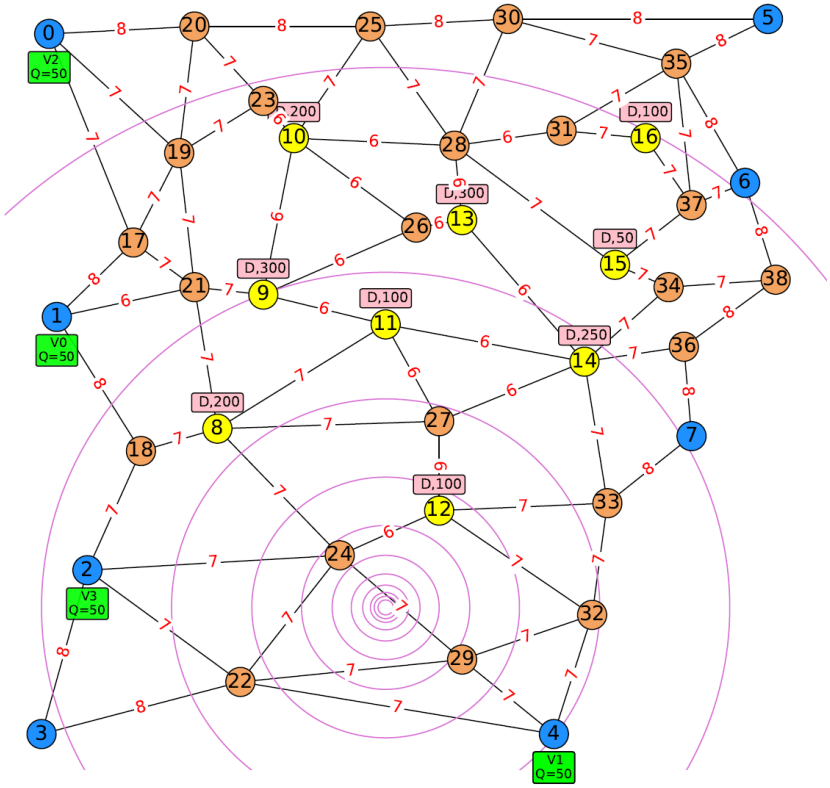


Figure A7. Road network for instance D8N22S9.

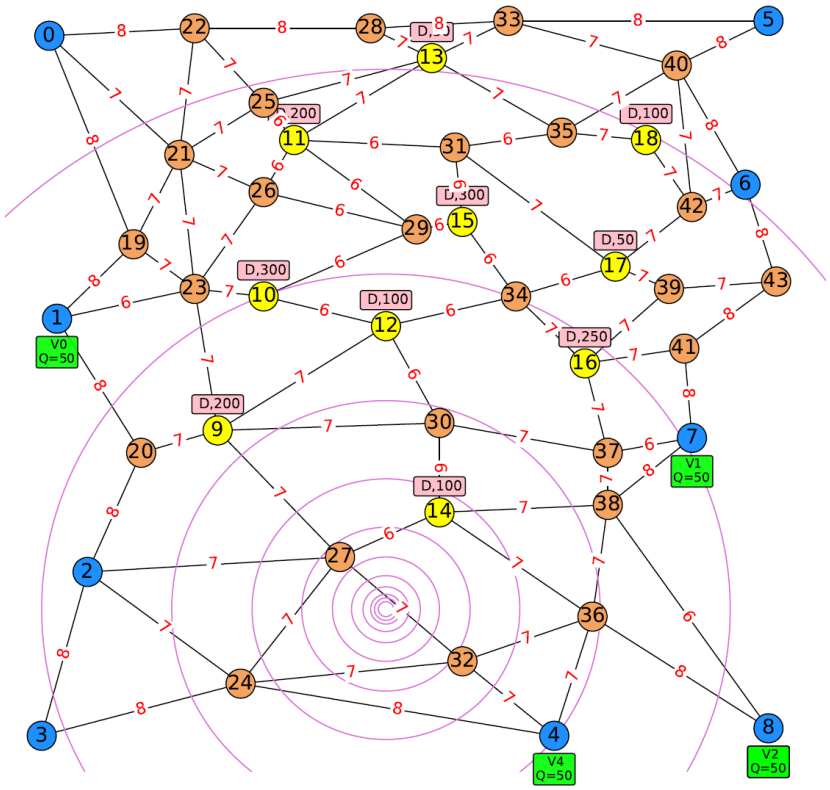


Figure A8. Road network for instance D9N25S10.

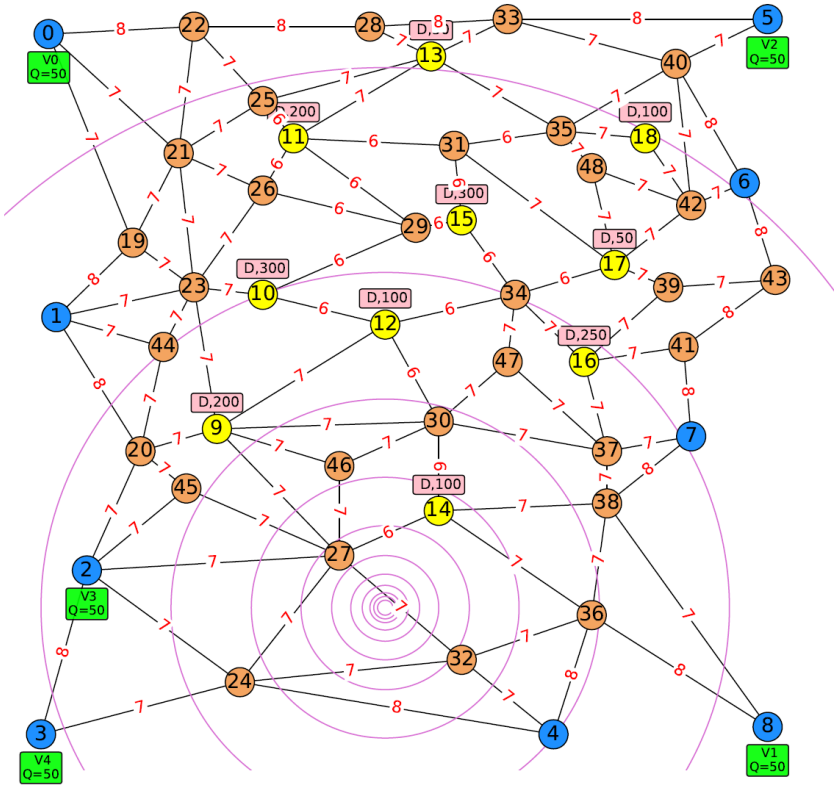


Figure A9. Road network for instance D9N30S10.

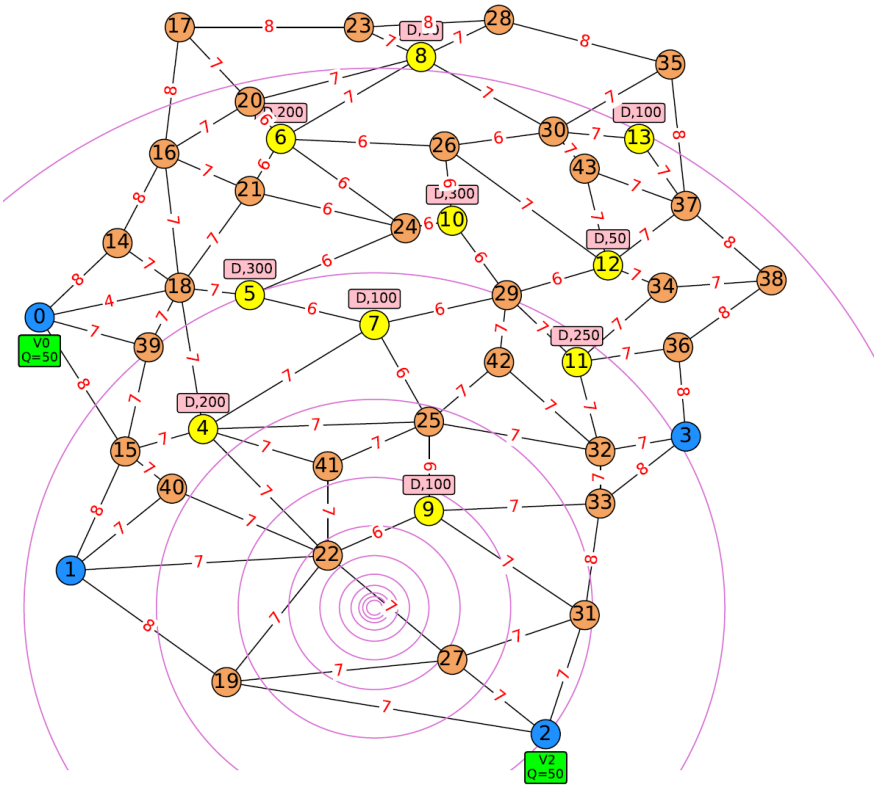


Figure A10. Road network for instance D4N30S10.

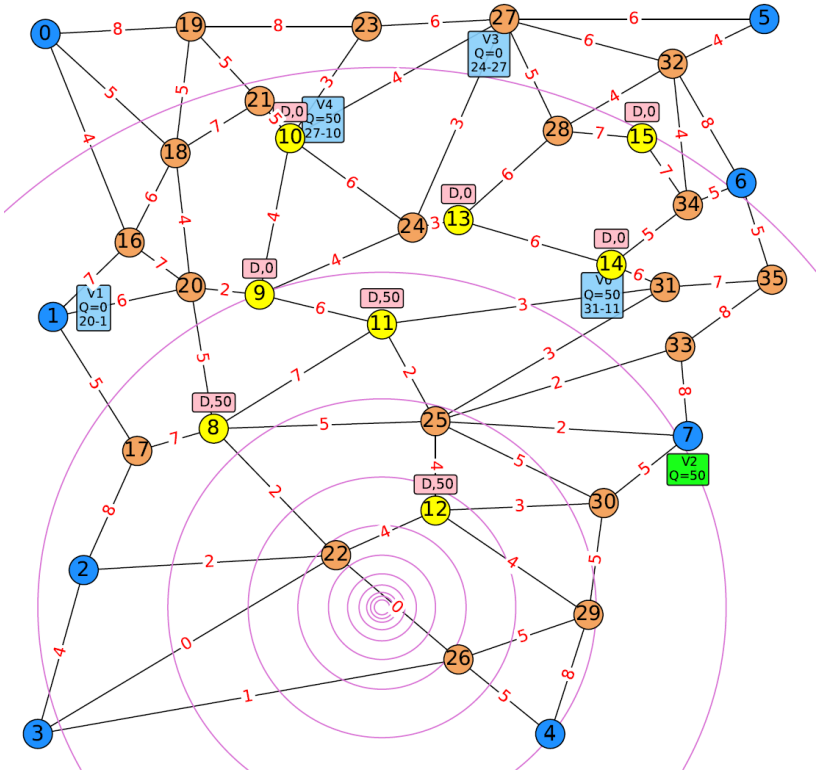


Figure A11. Example of medical supply delivery in progress for the network D8N20S20.

Table A1. Descriptive overall view on simulated data collected.

Total Test Instances: (10 Networks × 5 vehicle settings)	50
Proposed Base Heuristics & Benchmark: (5 + 1)	6
Total Test Instances: Base Heuristic & Benchmark (with 10 Omitted Matheuristic Benchmark: ((50 × 6) − 10))	290
Total Simulation Run	2900
Total Key Measurements	4
Total Set of 10 Samples Readings: 290 × 4 (for Four Key Measurements)	1160
Total Sample Readings	11,600
Total Normality Analysis (Shapiro-Wilk Test) Applied (for Each Key Measurement)	290
Total 10 Normal Sample Reading (Total Travelled Distance)	39 (13.44%)
Total 10 Normal Sample Reading (Total Travelled Time)	42 (14.48%)
Total 10 Normal Sample Reading (Total Computation Time)	148 (51.03%)
Total 10 Normal Sample Reading (Average Decision Computation Time)	166 (57.24%)
Total Comparative Analysis (Wilcoxon Signed-Ranks Test) Applied (for Each Key Measurement: ((50 × 5) − (10 × 5)))	200
Total Significant Difference (Total Travelled Distance)	68 (34%)
Total Significant Difference (Total Travelled Time)	69 (34.5%)
Total Significant Difference (Total Computation Time)	191 (95.5%)
Total Significant Difference (Average Decision Computation Time)	194 (97%)

Table A2. PDS_RA performance with TBIH-1 application as base heuristic.

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4		✓	✓			✓	✓	✓			✓	
D3N8S3_8		✓					✓	✓			✓	
D3N8S3_15	✓							✓			✓	
D3N8S3_30								✓			✓	
D3N8S3_50		✓	✓				✓			✓		
	1	3	2	0	0	1	3	4	0	1	4	0
D4N11S4_4							✓	✓			✓	
D4N11S4_8				✓				✓		✓	✓	✓
D4N11S4_15	✓	✓		✓	✓			✓		✓	✓	✓
D4N11S4_30		✓			✓		✓	✓		✓	✓	
D4N11S4_50	✓						✓			✓	✓	
	2	2	0	2	2	0	3	4	0	4	5	2

Table A2. *Cont.*

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D5N13S5_4							✓	✓	✓	✓	✓	
D5N13S5_8							✓	✓		✓	✓	
D5N13S5_15			✓				✓	✓	✓	✓	✓	
D5N13S5_30								✓			✓	✓
D5N13S5_50	✓			✓			✓	✓		✓	✓	
	1	0	1	1	0	0	4	5	2	4	5	1
D6N16S6_4							✓	✓	✓	✓	✓	
D6N16S6_8		✓					✓	✓		✓	✓	✓
D6N16S6_15								✓	✓	✓	✓	✓
D6N16S6_30			✓			✓		✓	✓		✓	✓
D6N16S6_50		✓			✓			✓			✓	
	0	2	1	0	1	1	2	5	3	3	5	3
D7N18S7_4	✓	✓		✓			✓	✓		✓	✓	✓
D7N18S7_8			✓				✓	✓	✓	✓	✓	✓
D7N18S7_15							✓	✓		✓	✓	
D7N18S7_30	✓	✓			✓			✓		✓	✓	✓
D7N18S7_50				✓				✓	✓		✓	
	2	2	1	2	1	0	3	5	2	4	5	3
D8N20S8_4			✓			✓	✓	✓	✓	✓	✓	✓
D8N20S8_8	✓		✓				✓	✓	✓	✓	✓	✓
D8N20S8_15							✓	✓		✓	✓	
D8N20S8_30	✓								✓			✓
D8N20S8_50				✓								✓
	2	0	2	1	0	1	3	3	3	3	3	4
D8N22S9_4		✓			✓		✓	✓		✓	✓	
D8N22S9_8								✓	✓		✓	✓
D8N22S9_15			✓				✓	✓	✓	✓	✓	✓
D8N22S9_30			✓	✓		✓			✓	✓		✓
D8N22S9_50			✓			✓			✓			
	0	1	3	1	1	2	2	3	4	3	3	3
D9N25S10_4		✓			✓		✓	✓	✓	✓	✓	✓
D9N25S10_8								✓		✓	✓	✓
D9N25S10_15							✓	✓		✓	✓	
D9N25S10_30			✓				✓			✓		
D9N25S10_50			✓									
	0	1	2	0	1	0	3	3	1	4	3	2
D9N30S10_4		✓			✓		✓	✓	✓	✓	✓	✓
D9N30S10_8	✓						✓	✓		✓	✓	
D9N30S10_15			✓			✓	✓	✓	✓	✓	✓	
D9N30S10_30			✓			✓	✓			✓		
D9N30S10_50				✓								
	1	1	2	1	1	2	4	3	2	4	3	1

Table A2. *Cont.*

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D4N30S10_4		✓			✓		✓	✓		✓	✓	
D4N30S10_8		✓			✓			✓			✓	
D4N30S10_15		✓			✓			✓			✓	✓
D4N30S10_30			✓				✓		✓			
D4N30S10_50			✓									
	0	3	2	0	3	0	2	3	1	1	3	1
Total	9	15	16	8	10	7	29	38	18	31	39	20
(%)	18.00	30.00	32.00	16.00	20.00	14.00	58.00	76.00	36.00	62.00	78.00	40.00

Table A3. PDS_RA performance with TBIH-2 application as base heuristic.

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4								✓	✓		✓	✓
D3N8S3_8								✓			✓	✓
D3N8S3_15	✓				✓		✓	✓	✓	✓	✓	✓
D3N8S3_30					✓			✓			✓	
D3N8S3_50							✓					
	1	0	0	0	2	0	2	4	2	1	4	3
D4N11S4_4							✓	✓	✓	✓	✓	✓
D4N11S4_8				✓				✓			✓	
D4N11S4_15			✓				✓	✓			✓	
D4N11S4_30		✓		✓			✓	✓		✓	✓	
D4N11S4_50	✓							✓			✓	
	1	1	1	2	0	0	3	5	1	2	5	1
D5N13S5_4		✓			✓		✓	✓		✓	✓	✓
D5N13S5_8	✓			✓			✓	✓	✓	✓	✓	✓
D5N13S5_15								✓		✓	✓	✓
D5N13S5_30								✓		✓	✓	
D5N13S5_50		✓		✓				✓		✓	✓	✓
	1	2	0	2	1	0	2	5	1	5	5	4
D6N16S6_4							✓	✓		✓	✓	
D6N16S6_8							✓	✓	✓	✓	✓	
D6N16S6_15		✓			✓		✓	✓		✓	✓	
D6N16S6_30								✓			✓	
D6N16S6_50		✓			✓			✓			✓	
	0	2	0	0	2	0	3	5	1	3	5	0
D7N18S7_4							✓	✓		✓	✓	
D7N18S7_8							✓	✓		✓	✓	
D7N18S7_15							✓	✓	✓	✓	✓	✓
D7N18S7_30		✓			✓			✓		✓	✓	
D7N18S7_50		✓			✓			✓			✓	✓
	0	2	0	0	2	0	3	5	1	4	5	2

Table A3. *Cont.*

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D8N20S8_4					✓		✓	✓		✓	✓	
D8N20S8_8						✓	✓	✓		✓	✓	
D8N20S8_15						✓	✓	✓		✓	✓	
D8N20S8_30										✓		
D8N20S8_50												
	0	0	0	0	1	2	3	3	0	4	3	0
D8N22S9_4							✓	✓		✓	✓	
D8N22S9_8							✓	✓		✓	✓	
D8N22S9_15							✓	✓		✓	✓	
D8N22S9_30												
D8N22S9_50												✓
	0	0	0	0	0	0	3	3	0	3	3	1
D9N25S10_4							✓	✓		✓	✓	
D9N25S10_8								✓			✓	
D9N25S10_15	✓						✓	✓		✓	✓	
D9N25S10_30						✓	✓			✓		
D9N25S10_50												
	1	0	0	0	0	1	3	3	0	3	3	0
D9N30S10_4		✓					✓	✓		✓	✓	
D9N30S10_8		✓			✓		✓	✓		✓	✓	
D9N30S10_15							✓	✓		✓	✓	✓
D9N30S10_30							✓			✓		
D9N30S10_50			✓									
	0	2	1	0	1	0	4	3	0	4	3	1
D4N30S10_4		✓			✓		✓	✓		✓	✓	
D4N30S10_8					✓		✓	✓		✓	✓	
D4N30S10_15					✓			✓	✓	✓	✓	
D4N30S10_30												✓
D4N30S10_50										✓		
	0	1	0	0	3	0	2	3	1	4	3	1
Total	4	10	2	4	12	3	28	39	7	33	39	13
(%)	8.00	20.00	4.00	8.00	24.00	6.00	56.00	78.00	14.00	66.00	78.00	26.00

Table A4. PDS_RA performance with TBIH-3 application as base heuristic.

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4								✓			✓	
D3N8S3_8	✓							✓			✓	
D3N8S3_15							✓	✓			✓	
D3N8S3_30				✓	✓							
D3N8S3_50					✓		✓			✓		
	1	0	0	1	2	0	2	3	0	1	3	0

Table A4. *Cont.*

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D4N11S4_4							✓	✓			✓	
D4N11S4_8		✓			✓			✓		✓	✓	
D4N11S4_15		✓			✓			✓			✓	
D4N11S4_30		✓					✓	✓		✓	✓	
D4N11S4_50	✓	✓		✓	✓						✓	
	1	4	0	1	3	0	2	4	0	2	5	0
D5N13S5_4		✓			✓		✓	✓		✓	✓	
D5N13S5_8		✓			✓		✓	✓			✓	
D5N13S5_15					✓			✓		✓	✓	
D5N13S5_30	✓			✓				✓		✓	✓	
D5N13S5_50		✓		✓	✓			✓			✓	
	1	3	0	2	4	0	2	5	0	3	5	0
D6N16S6_4		✓			✓		✓	✓		✓	✓	
D6N16S6_8		✓			✓		✓	✓		✓	✓	
D6N16S6_15		✓			✓		✓	✓		✓	✓	
D6N16S6_30	✓			✓				✓		✓	✓	
D6N16S6_50							✓	✓			✓	
	1	3	0	1	3	0	4	5	0	4	5	0
D7N18S7_4							✓	✓		✓	✓	
D7N18S7_8							✓	✓		✓	✓	
D7N18S7_15							✓	✓		✓	✓	
D7N18S7_30		✓		✓	✓			✓	✓		✓	
D7N18S7_50		✓			✓			✓			✓	
	0	2	0	1	2	0	3	5	1	3	5	0
D8N20S8_4								✓			✓	
D8N20S8_8	✓			✓			✓	✓		✓	✓	
D8N20S8_15							✓	✓		✓	✓	
D8N20S8_30	✓		✓			✓						
D8N20S8_50			✓			✓			✓			
	2	0	2	1	0	2	2	3	1	2	3	0
D8N22S9_4		✓			✓		✓	✓		✓	✓	
D8N22S9_8							✓	✓		✓	✓	
D8N22S9_15							✓	✓		✓	✓	
D8N22S9_30												
D8N22S9_50												
	0	1	0	0	1	0	3	3	0	3	3	0
D9N25S10_4							✓	✓		✓	✓	
D9N25S10_8					✓			✓			✓	
D9N25S10_15							✓	✓		✓	✓	
D9N25S10_30									✓			
D9N25S10_50												
	0	0	0	0	1	0	2	3	1	2	3	0

Table A4. *Cont.*

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D9N30S10_4		✓			✓		✓	✓		✓	✓	
D9N30S10_8							✓	✓		✓	✓	
D9N30S10_15							✓	✓		✓	✓	
D9N30S10_30										✓		
D9N30S10_50												
	0	1	0	0	1	0	3	3	0	4	3	0
D4N30S10_4		✓			✓			✓			✓	✓
D4N30S10_8		✓			✓			✓		✓	✓	
D4N30S10_15		✓		✓	✓			✓			✓	
D4N30S10_30												
D4N30S10_50									✓			
	0	3	0	1	3	0	0	3	1	1	3	1
Total	6	17	2	8	20	2	23	37	4	25	38	1
(%)	12.00	34.00	4.00	16.00	40.00	4.00	46.00	74.00	8.00	50.00	76.00	2.00

Table A5. PDS_RA performance with TBIH-4 application as base heuristic.

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4	✓			✓				✓		✓	✓	
D3N8S3_8								✓			✓	
D3N8S3_15	✓							✓		✓	✓	
D3N8S3_30			✓	✓		✓	✓	✓	✓	✓	✓	✓
D3N8S3_50								✓	✓		✓	✓
	2	0	1	2	0	1	1	5	2	3	5	2
D4N11S4_4			✓		✓	✓	✓	✓		✓	✓	
D4N11S4_8							✓	✓	✓	✓	✓	
D4N11S4_15								✓	✓		✓	
D4N11S4_30		✓		✓		✓	✓	✓	✓	✓	✓	✓
D4N11S4_50		✓		✓				✓	✓		✓	✓
	0	2	1	2	1	2	3	5	4	3	5	2
D5N13S5_4	✓						✓	✓		✓	✓	
D5N13S5_8						✓	✓	✓		✓	✓	
D5N13S5_15	✓					✓	✓	✓			✓	
D5N13S5_30						✓		✓	✓		✓	
D5N13S5_50		✓		✓		✓	✓	✓	✓	✓	✓	
	2	1	0	1	0	4	4	5	2	3	5	0
D6N16S6_4								✓		✓	✓	✓
D6N16S6_8		✓					✓	✓		✓	✓	
D6N16S6_15		✓			✓		✓	✓		✓	✓	
D6N16S6_30								✓			✓	
D6N16S6_50		✓			✓			✓	✓		✓	✓
	0	3	0	0	2	0	2	5	1	3	5	2

Table A5. *Cont.*

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D7N18S7_4		✓			✓		✓	✓		✓	✓	
D7N18S7_8		✓			✓		✓	✓		✓	✓	
D7N18S7_15			✓			✓	✓	✓		✓	✓	
D7N18S7_30		✓			✓			✓		✓	✓	
D7N18S7_50		✓						✓			✓	
	0	4	1	0	3	1	3	5	0	4	5	0
D8N20S8_4							✓	✓		✓	✓	
D8N20S8_8							✓	✓		✓	✓	
D8N20S8_15	✓			✓			✓	✓		✓	✓	
D8N20S8_30												
D8N20S8_50												
	1	0	0	1	0	0	3	3	0	3	3	0
D8N22S9_4		✓			✓		✓	✓		✓	✓	
D8N22S9_8							✓	✓		✓	✓	
D8N22S9_15							✓	✓		✓	✓	
D8N22S9_30												
D8N22S9_50												
	0	1	0	0	1	0	3	3	0	3	3	0
D9N25S10_4							✓	✓		✓	✓	
D9N25S10_8								✓	✓		✓	
D9N25S10_15							✓	✓		✓	✓	
D9N25S10_30							✓			✓		✓
D9N25S10_50				✓		✓				✓		
	0	0	0	1	0	1	3	3	1	4	3	1
D9N30S10_4							✓	✓		✓	✓	
D9N30S10_8							✓	✓	✓	✓	✓	✓
D9N30S10_15				✓			✓	✓		✓	✓	
D9N30S10_30									✓			✓
D9N30S10_50				✓		✓			✓			✓
	0	0	0	2	0	1	3	3	3	3	3	3
D4N30S10_4	✓	✓		✓	✓		✓	✓		✓	✓	
D4N30S10_8					✓		✓	✓		✓	✓	
D4N30S10_15				✓	✓		✓	✓		✓	✓	
D4N30S10_30						✓						
D4N30S10_50						✓	✓			✓		
	1	1	0	2	3	2	4	3	0	4	3	0
Total	6	12	3	11	10	12	29	40	13	33	40	10
(%)	12.00	24.00	6.00	22.00	20.00	24.00	58.00	80.00	26.00	66.00	80.00	20.00

Table A6. PDS_RA performance with TBIH-5 application as base heuristic.

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4								✓		✓	✓	
D3N8S3_8		✓	✓	✓	✓	✓		✓	✓		✓	
D3N8S3_15								✓			✓	
D3N8S3_30		✓			✓		✓			✓		
D3N8S3_50	✓	✓					✓					
	1	3	1	1	2	1	2	3	1	2	3	0
D4N11S4_4							✓	✓			✓	
D4N11S4_8							✓	✓		✓	✓	
D4N11S4_15	✓				✓			✓			✓	
D4N11S4_30		✓					✓	✓		✓	✓	
D4N11S4_50								✓			✓	
	1	1	0	0	1	0	3	5	0	2	5	0
D5N13S5_4		✓			✓		✓	✓		✓	✓	
D5N13S5_8					✓		✓	✓		✓	✓	
D5N13S5_15								✓		✓	✓	
D5N13S5_30							✓	✓		✓	✓	
D5N13S5_50				✓	✓		✓			✓	✓	
	0	1	0	1	3	0	4	4	0	5	5	0
D6N16S6_4		✓			✓		✓	✓		✓	✓	
D6N16S6_8		✓			✓		✓	✓		✓	✓	
D6N16S6_15	✓	✓		✓	✓		✓	✓		✓	✓	
D6N16S6_30	✓							✓			✓	
D6N16S6_50		✓			✓			✓			✓	
	2	4	0	1	4	0	3	5	0	3	5	0
D7N18S7_4						✓	✓	✓	✓	✓	✓	
D7N18S7_8							✓	✓		✓	✓	
D7N18S7_15							✓	✓		✓	✓	
D7N18S7_30		✓			✓		✓	✓		✓	✓	
D7N18S7_50	✓	✓			✓			✓			✓	
	1	2	0	0	2	1	4	5	1	4	5	0
D8N20S8_4							✓	✓		✓	✓	
D8N20S8_8								✓		✓	✓	
D8N20S8_15			✓					✓	✓		✓	✓
D8N20S8_30												
D8N20S8_50												
	0	0	1	0	0	0	1	3	1	2	3	1
D8N22S9_4							✓	✓	✓	✓	✓	✓
D8N22S9_8			✓			✓	✓	✓		✓	✓	
D8N22S9_15							✓	✓		✓	✓	
D8N22S9_30							✓			✓		
D8N22S9_50												
	0	0	1	0	0	1	4	3	1	4	3	1

Table A6. *Cont.*

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D9N25S10_4							✓	✓		✓	✓	
D9N25S10_8								✓			✓	
D9N25S10_15							✓	✓	✓	✓	✓	✓
D9N25S10_30	✓											
D9N25S10_50	✓						✓		✓			✓
	2	0	0	0	0	0	3	3	2	2	3	2
D9N30S10_4		✓			✓		✓	✓		✓	✓	
D9N30S10_8		✓			✓		✓	✓		✓	✓	
D9N30S10_15							✓	✓		✓	✓	
D9N30S10_30	✓						✓			✓		
D9N30S10_50												
	1	2	0	0	2	0	4	3	0	4	3	0
D4N30S10_4		✓			✓		✓	✓	✓	✓	✓	
D4N30S10_8					✓			✓	✓		✓	✓
D4N30S10_15			✓		✓			✓			✓	
D4N30S10_30				✓								
D4N30S10_50												✓
	0	1	1	1	3	0	1	3	2	1	3	2
Total	8	14	4	4	17	3	29	37	8	29	38	6
(%)	16.00	28.00	8.00	8.00	34.00	6.00	58.00	74.00	16.00	58.00	76.00	12.00

Table A7. PDS_RA performance with CPLEX application as base heuristic.

	Total Distance Travelled			Total Travelled Time			Total Computation Time			Average Decision Computation		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D3N8S3_4												
D3N8S3_8										✓		
D3N8S3_15			✓	✓		✓						
D3N8S3_30												
D3N8S3_50						✓						
	0	-	1	1	-	2	0	-	0	1	-	0
D4N11S4_4												
D4N11S4_8			✓			✓				✓		
D4N11S4_15	✓			✓		✓						
D4N11S4_30			✓	✓								
D4N11S4_50	✓		✓	✓		✓	✓			✓		
	2	-	3	3	-	3	1	-	0	2	-	0
D5N13S5_4			✓			✓						
D5N13S5_8			✓									
D5N13S5_15												
D5N13S5_30	✓		✓							✓		
D5N13S5_50			✓	✓			✓			✓		
	1	-	4	1	-	1	1	-	0	2	-	0

Table A7. *Cont.*

	Total Distance Travelled			**Total Travelled Time**			**Total Computation Time**			**Average Decision Computation**		
Test Instance	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**	**N**	**Sig.**	**BSF**
D6N16S6_4			✓			✓	✓			✓		
D6N16S6_8	✓		✓			✓				✓		
D6N16S6_15			✓			✓	✓					
D6N16S6_30										✓		
D6N16S6_50	✓		✓	✓		✓	✓			✓		
	2	-	4	1	-	4	3	-	0	4	-	0
D7N18S7_4			✓									
D7N18S7_8						✓	✓					
D7N18S7_15				✓			✓			✓		
D7N18S7_30			✓			✓						
D7N18S7_50			✓			✓				✓		
	0	-	3	1	-	3	2	-	0	2	-	0
D8N20S8_4							✓			✓		
D8N20S8_8												
D8N20S8_15							✓					
D8N20S8_30												
D8N20S8_50												
	0	-	0	0	-	0	2	-	0	1	-	0
D8N22S9_4			✓			✓	✓			✓		
D8N22S9_8												
D8N22S9_15	✓					✓						
D8N22S9_30												
D8N22S9_50												
	1	-	1	0	-	2	1	-	0	1	-	0
D9N25S10_4			✓			✓						
D9N25S10_8			✓			✓				✓		
D9N25S10_15			✓			✓						
D9N25S10_30												
D9N25S10_50												
	0	-	3	0	-	3	0	-	0	1	-	0
D9N30S10_4			✓			✓						
D9N30S10_8			✓			✓						
D9N30S10_15												
D9N30S10_30												
D9N30S10_50												
	0	-	2	0	-	2	0	-	0	0	-	0
D4N30S10_4			✓			✓				✓		
D4N30S10_8			✓			✓						
D4N30S10_15						✓						
D4N30S10_30												
D4N30S10_50												
	0	-	2	0	-	3	0	-	0	1	-	0
Total	6	-	23	7	-	23	10	-	0	15	-	0
(%)	12.00	-	46.00	14.00	-	46.00	20.00	-	0.00	30.00	-	0.00
(% V2)	15.00	-	57.50	17.50	-	57.50	25.00	-	0.00	37.50	-	0.00

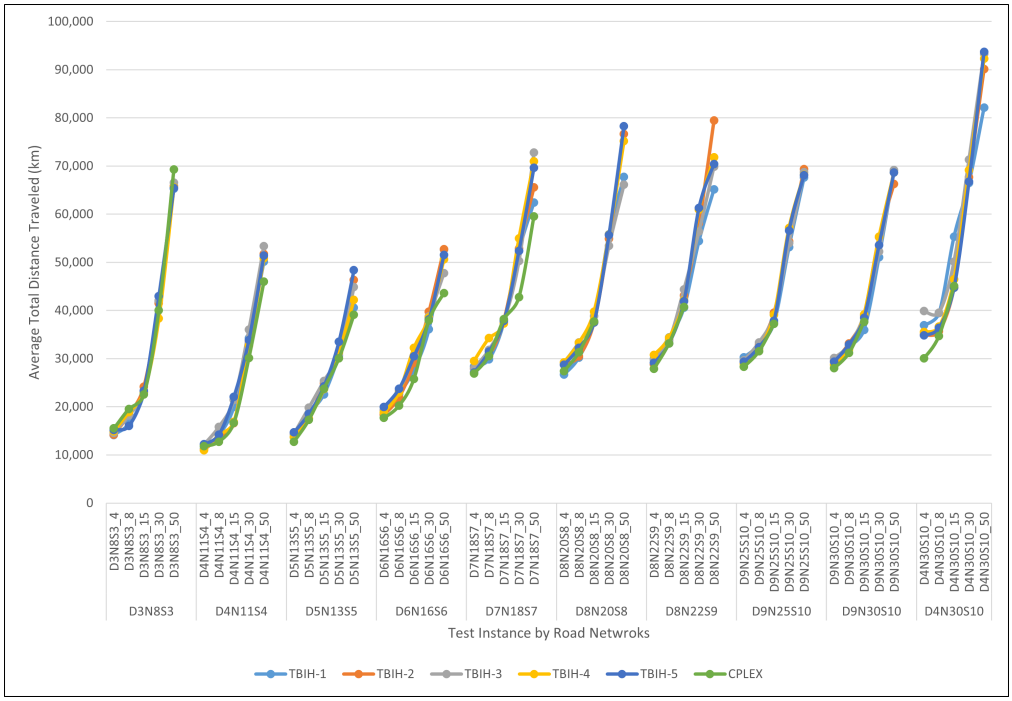


Figure A12. Algorithms performance compared to matheuristic rollout applied in PDS-RA: average total distance travelled (km) based on test instances.

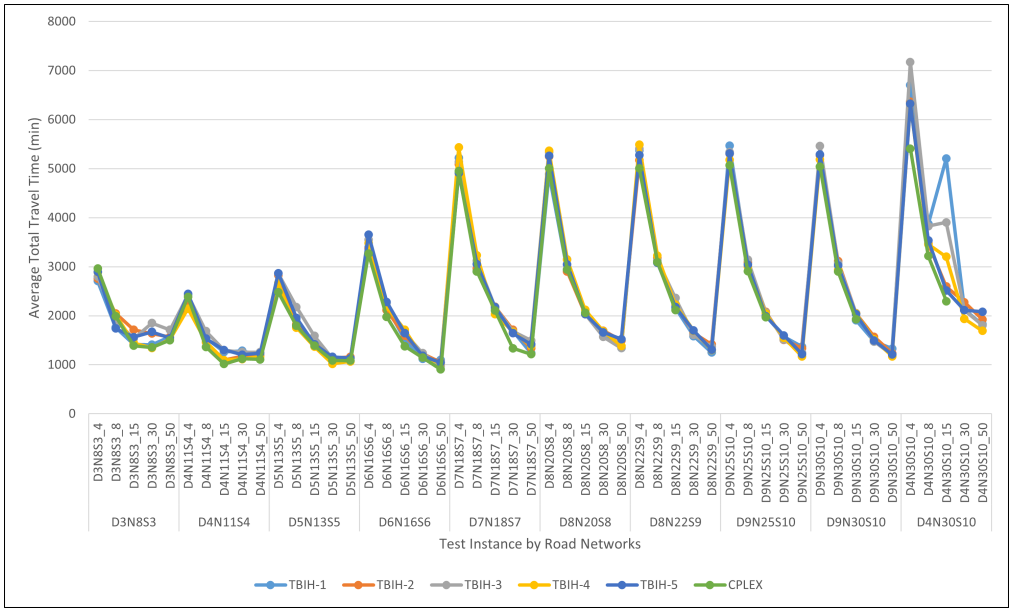


Figure A13. Algorithms performance compared to matheuristic rollout applied in PDS-RA: average total travel time (min) based on test instances.

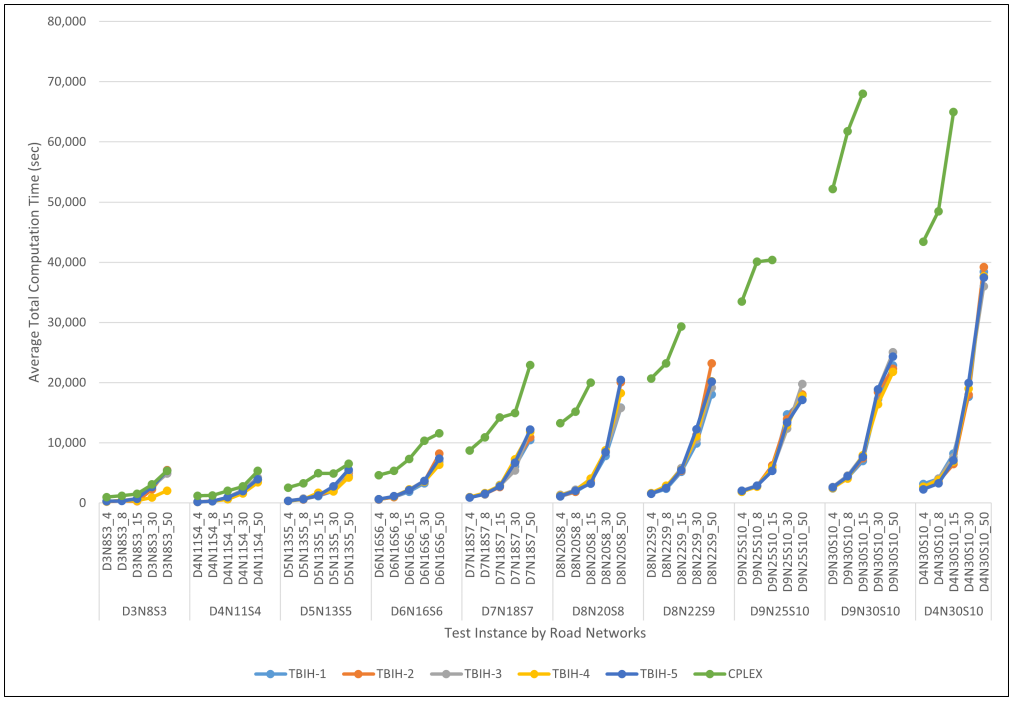


Figure A14. Algorithms performance compared to matheuristic rollout applied in PDS-RA: average total computation time (sec) based on test instances.

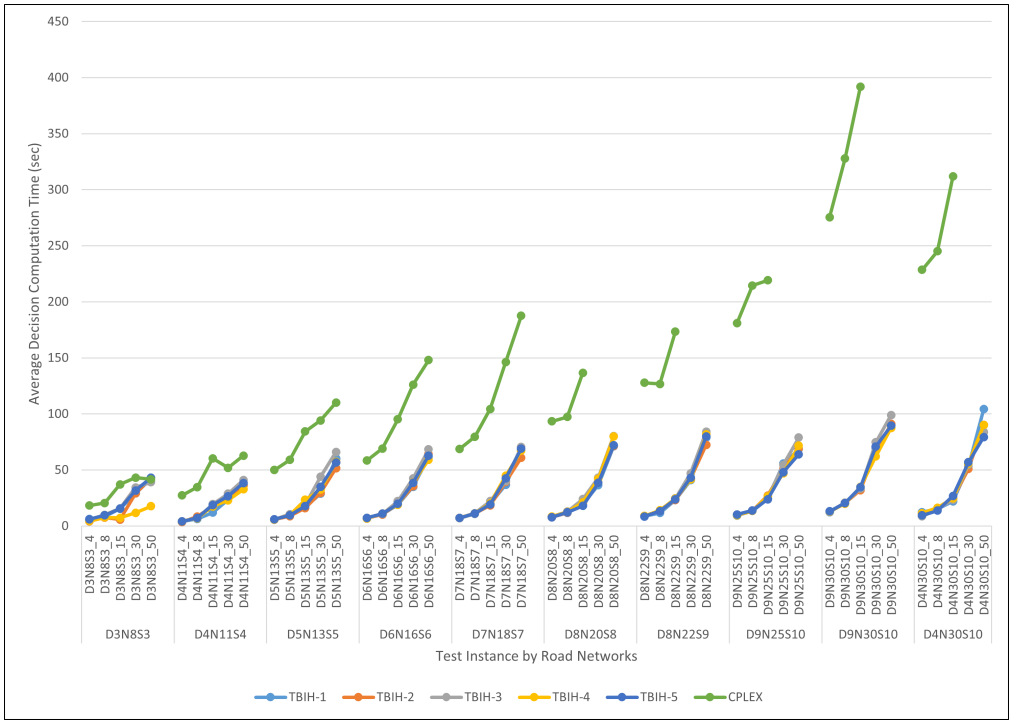


Figure A15. Algorithms performance compared to matheuristic rollout applied in PDS-RA: average decision computation time (sec) based on test instances.

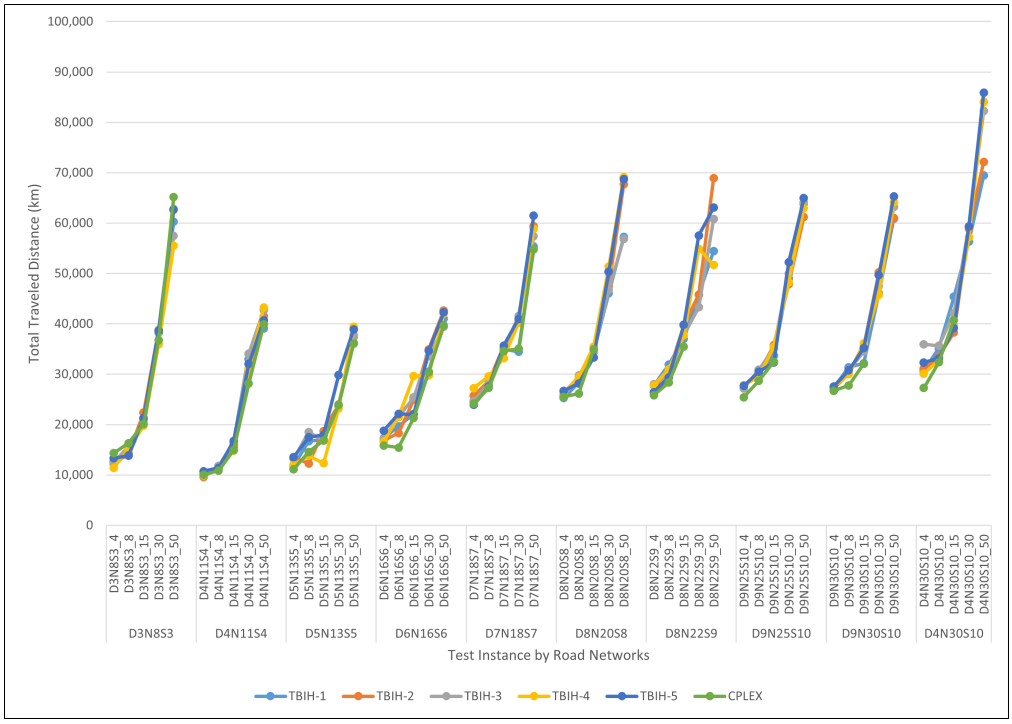


Figure A16. Algorithms performance compared to matheuristic rollout applied in PDS-RA: best total distance travelled (km) measured based on test instances.

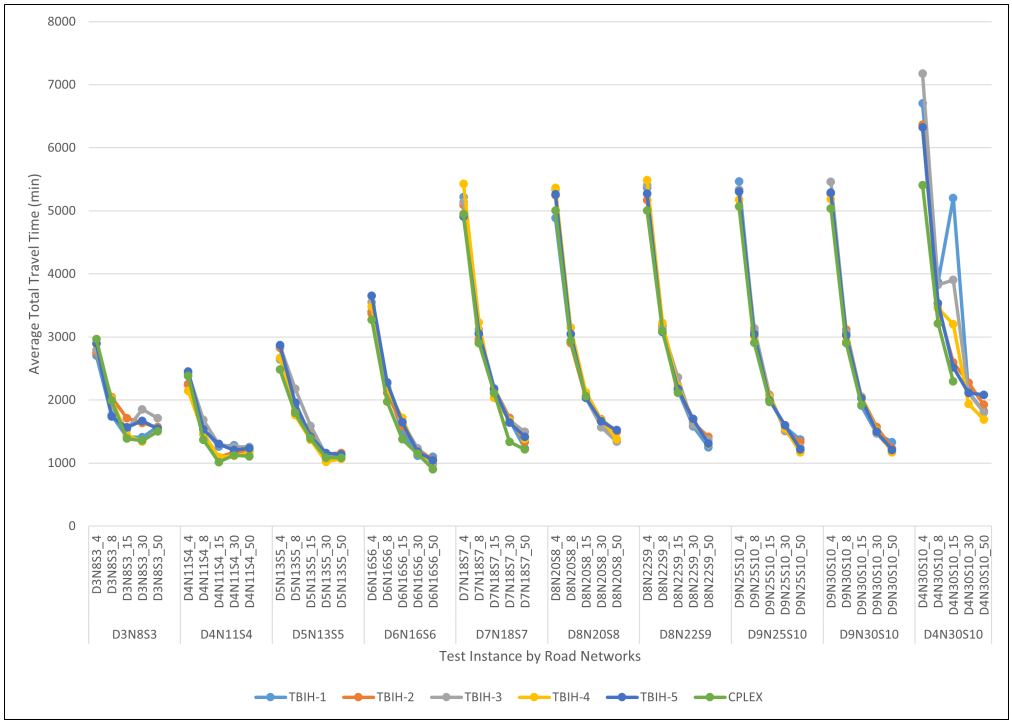


Figure A17. Algorithms performance compared to matheuristic rollout applied in PDS-RA: best total travel time (min) measured based on test instances.

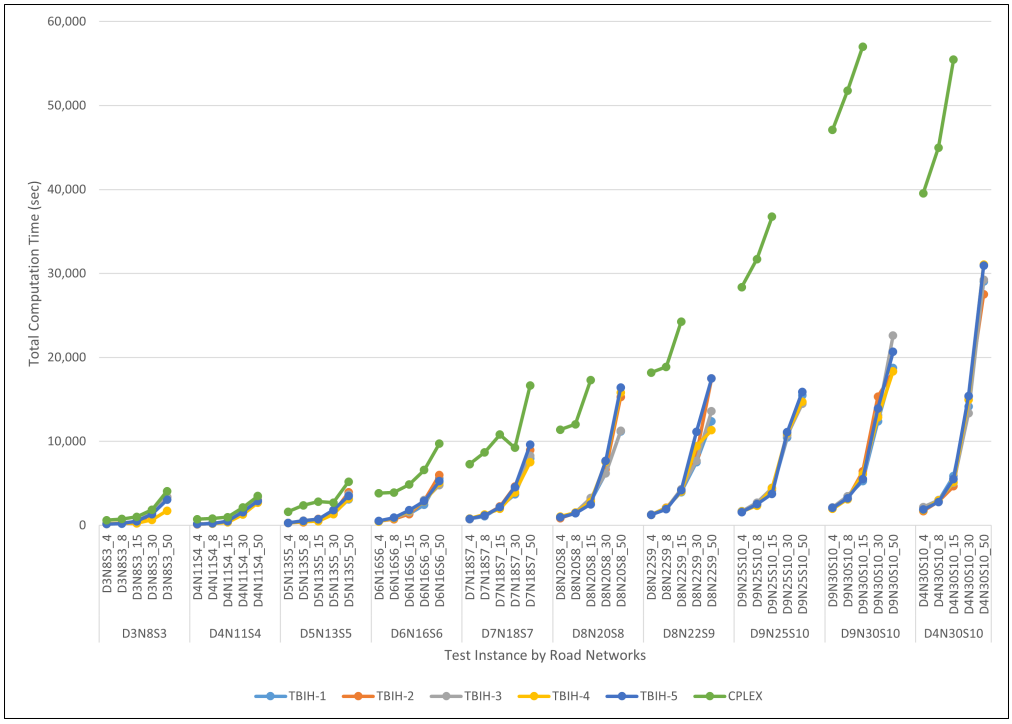


Figure A18. Algorithms performance compared to matheuristic rollout applied in PDS-RA: best total computation time (sec) measured based on test instances.

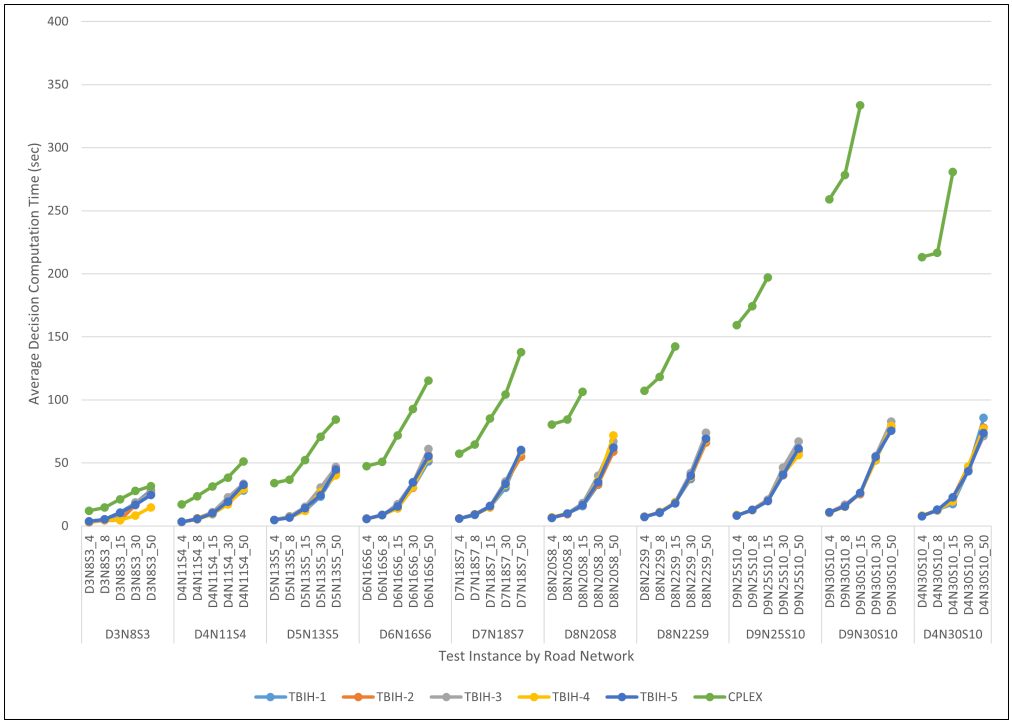


Figure A19. Algorithms performance compared to matheuristic rollout applied in PDS-RA: best average decision computation time (sec) measured based on test instances.

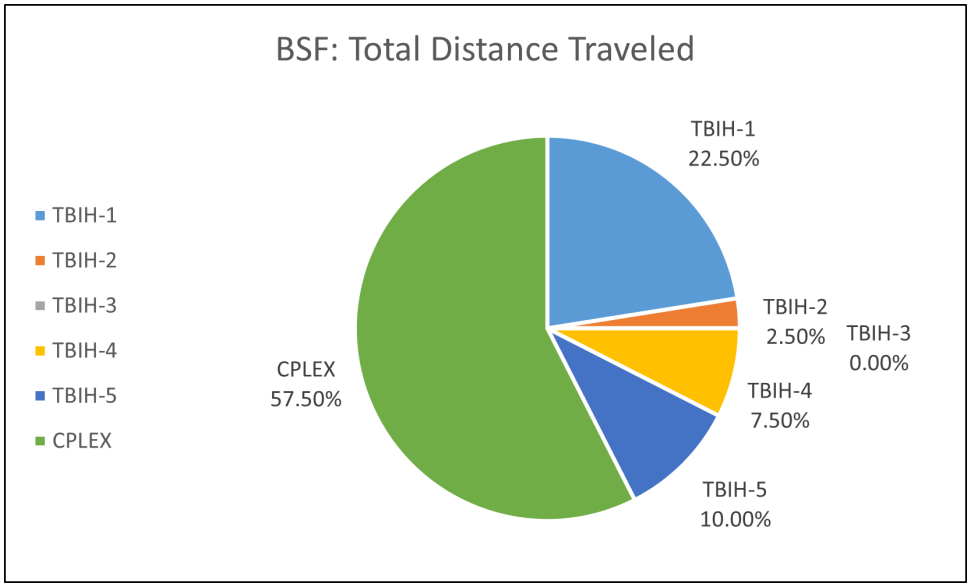


Figure A20. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total travelled distance over all test instances applied (omitting 10 non-benchmarked measurements).

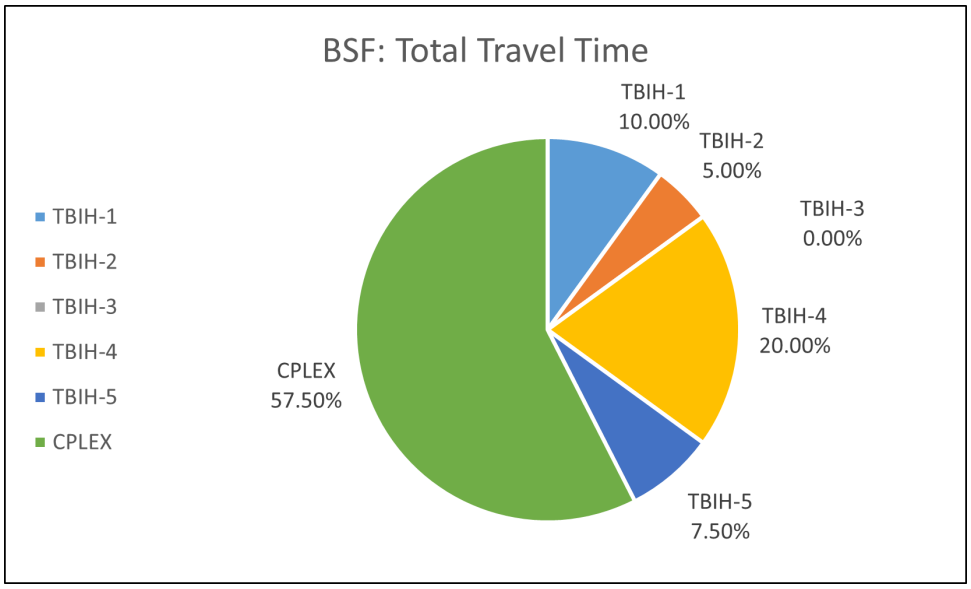


Figure A21. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total travel time over all test instances applied (omitting 10 non-benchmarked measurements).

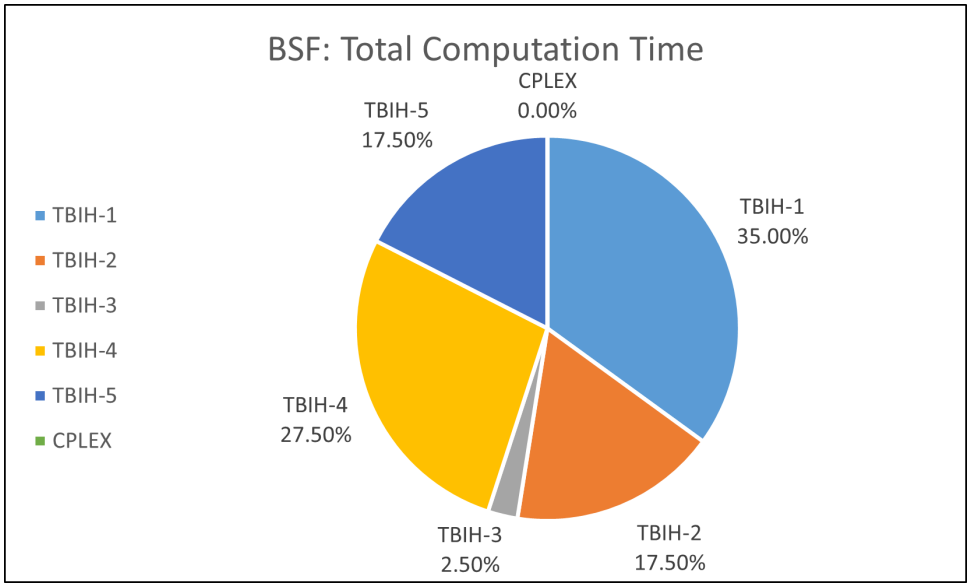


Figure A22. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total computation time over all test instances applied (omitting 10 non-benchmarked aasurements).

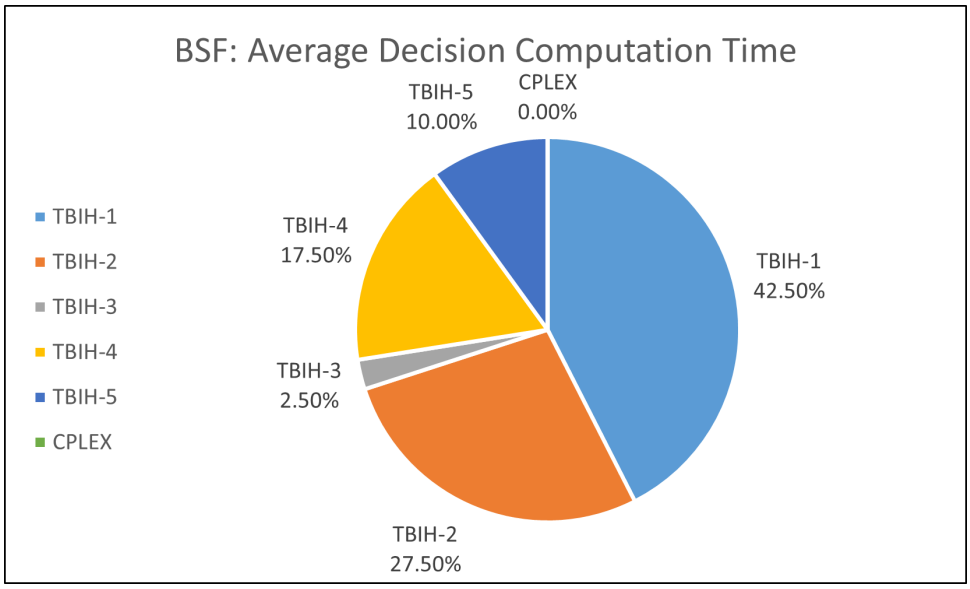


Figure A23. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for average decision computation time over all test instances applied (omitting 10 non-benchmarked measurements).

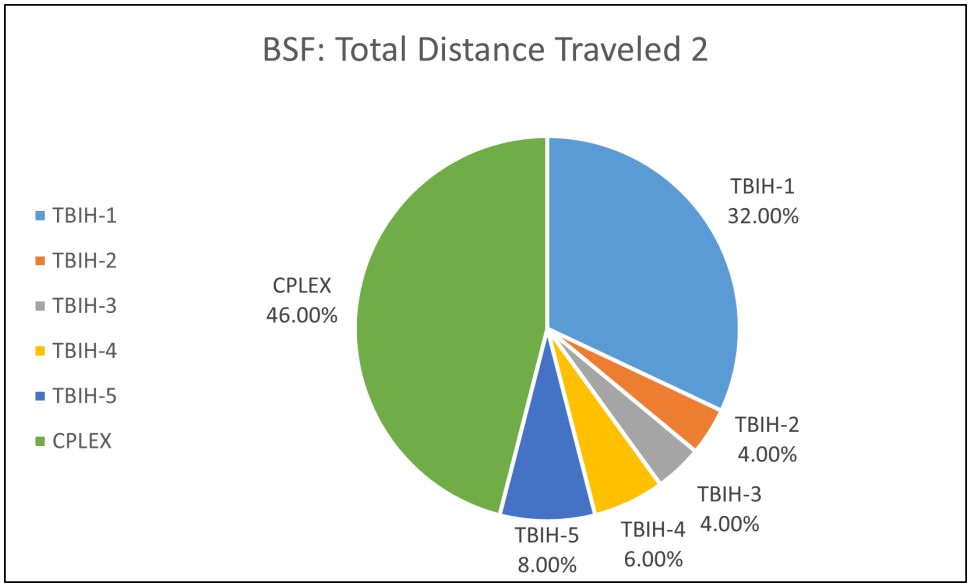


Figure A24. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total travelled distance over all test instances applied (including 10 non-benchmarked measurements).

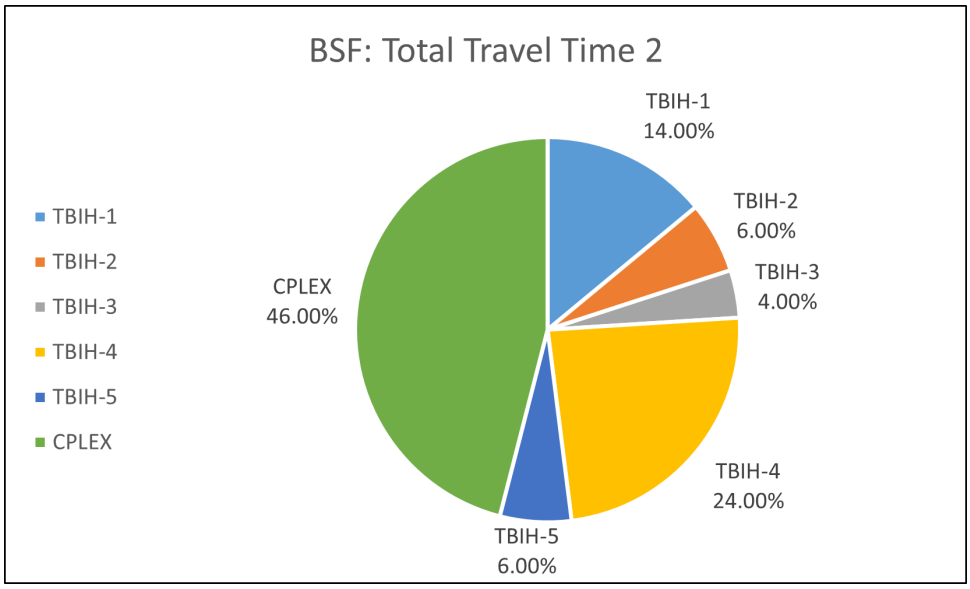


Figure A25. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total travel time over all test instances applied (including 10 non-benchmarked measurements).

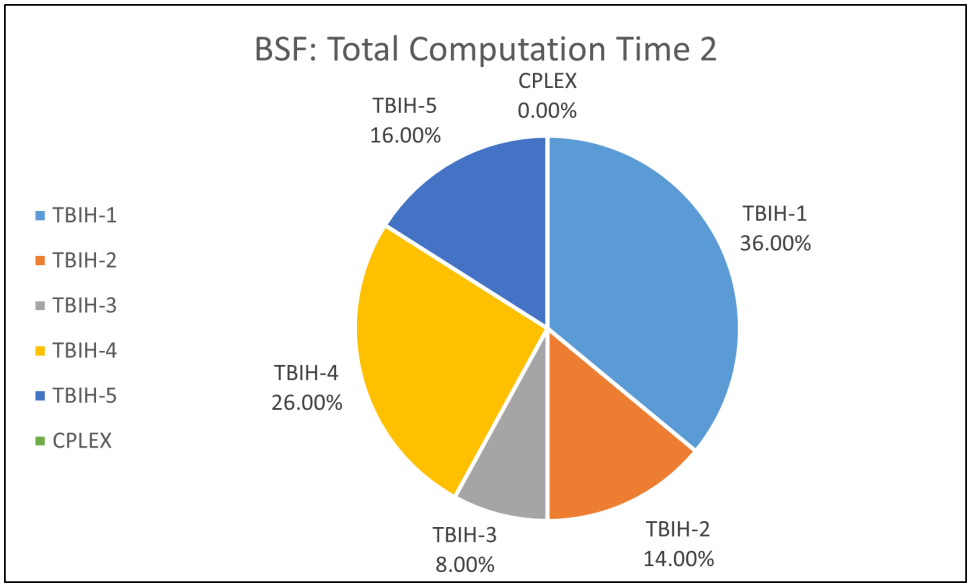


Figure A26. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for total computation time over all test instances applied (including 10 non-benchmarked measurements).

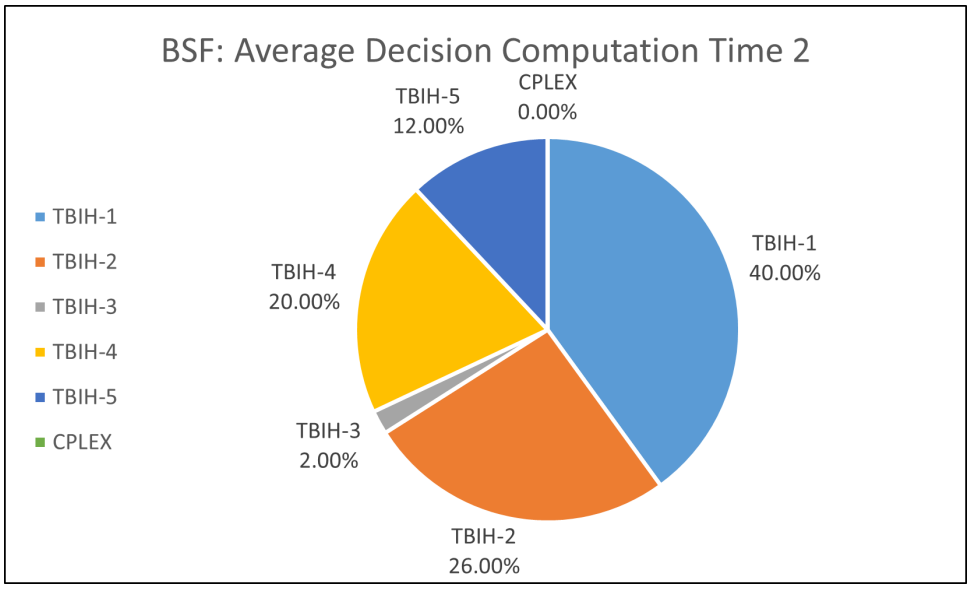


Figure A27. Algorithms performance compared to matheuristic rollout applied in PDS-RA: total BSF measured for average decision computation time over all test instances applied (including 10 non-benchmarked measurements).

References

1. Chauhan, A.; Chopra, B.K. Deployment of medical relief teams of the Indian army in the aftermath of the Nepal earthquake: Lessons learned. *Disaster Med. Public Health Prep.* **2017**, *11*, 394–398. [CrossRef] [PubMed]
2. Sharma, D.C. Nepal earthquake exposes gaps in disaster preparedness. *Lancet* **2015**, *385*, 1819–1820. [CrossRef]
3. Radianti, J.; Hiltz, S.R.; Labaka, L. An overview of public concerns during the recovery period after a major earthquake: Nepal twitter analysis. In Proceedings of the 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA, 5–8 January 2016; pp. 136–145.
4. Baharmand, H.; Comes, T.; Lauras, M. Managing in-country transportation risks in humanitarian supply chains by logistics service providers: Insights from the 2015 Nepal earthquake. *Int. J. Disaster Risk Reduct.* **2017**, *24*, 549–559. [CrossRef]
5. Tian, Y.; Owen, L.A.; Xu, C.; Ma, S.; Li, K.; Xu, X.; Figueiredo, P.M.; Kang, W.; Guo, P.; Wang, S.; et al. Landslide development within 3 years after the 2015 M w 7.8 Gorkha earthquake, Nepal. *Landslides* **2020**, *17*, 1251–1267. [CrossRef]

6. Xie, Q.; Gaohu, L.; Chen, H.; Xu, C.; Feng, B. Seismic damage to road networks subjected to earthquakes in Nepal, 2015. *Earthq. Eng. Eng. Vib.* **2017**, *16*, 649–670. [CrossRef]
7. Heckmann, I.; Comes, T.; Nickel, S. A critical review on supply chain risk–Definition, measure and modeling. *Omega* **2015**, *52*, 119–132. [CrossRef]
8. Neupane, S.P. Immediate lessons from the Nepal earthquake. *Lancet* **2015**, *385*, 2041–2042. [CrossRef]
9. Archetti, C.; Savelsbergh, M.W.; Speranza, M.G. Worst-case analysis for split delivery vehicle routing problems. *Transp. Sci.* **2006**, *40*, 226–234. [CrossRef]
10. Bellman, R.; Lee, E. History and development of dynamic programming. *IEEE Control Syst. Mag.* **1984**, *4*, 24–28. [CrossRef]
11. Van Roy, B.; Bertsekas, D.P.; Lee, Y.; Tsitsiklis, J.N. A neuro-dynamic programming approach to retailer inventory management. In Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, CA, USA, 12 December 1997; Volume 4, pp. 4052–4057.
12. Bellman, R. The theory of dynamic programming. *Bull. Am. Math. Soc.* **1954**, *60*, 503–515. [CrossRef]
13. Goodson, J.C. Solution Methodologies for Vehicle Routing Problems with Stochastic Demand. Ph.D. Thesis, University of Iowa, Iowa City, IA, USA, 2010.
14. Ulmer, M.W.; Goodson, J.C.; Mattfeld, D.C.; Thomas, B.W. *Route-Based Markov Decision Processes for Dynamic Vehicle Routing Problems*; Technical Report; Braunschweig, Geremany, 2017. Available online: https://web.winforms.phil.tu-bs.de/paper/ulmer/Ulmer_model.pdf (accessed on 7 July 2021).
15. Anuar, W.K.; Lee, L.S.; Seow, H.V.; Pickl, S. A multi-depot vehicle routing problem with stochastic road capacity and reduced two-stage stochastic integer linear programming models for rollout algorithm. *Mathematics* **2021**, *9*, 1572. [CrossRef]
16. Anuar, W.K.; Moll, M.; Lee, L.; Pickl, S.; Seow, H. Vehicle routing optimization for humanitarian logistics in disaster recovery: A survey. In Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, NV, USA, 28 July–1 August 2019; pp. 161–167. Available online: https://www.proquest.com/openview/24d13cfa7f7ed47c7948a895a66d8a1a/1?pq-origsite=gscholar&cbl=1976342 (accessed on 7 July 2021).
17. Anuar, W.K.; Lee, L.S.; Pickl, S.; Seow, H.V. Vehicle routing optimisation in humanitarian operations: A survey on modelling and optimisation approaches. *Appl. Sci.* **2021**, *11*, 667. [CrossRef]
18. Yan, Y.; Chow, A.H.; Ho, C.P.; Kuo, Y.H.; Wu, Q.; Ying, C. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *162*, 102712. [CrossRef]
19. Wang, Q.; Tang, C. Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowl.-Based Syst.* **2021**, *233*, 107526. [CrossRef]
20. Rios, B.H.O.; Xavier, E.C.; Miyazawa, F.K.; Amorim, P.; Curcio, E.; Santos, M.J. Recent dynamic vehicle routing problems: A survey. *Comput. Ind. Eng.* **2021**, *160*, 107604.
21. Chang, K.H.; Hsiung, T.Y.; Chang, T.Y. Multi-Commodity distribution under uncertainty in disaster response phase: Model, solution method, and an empirical study. *Eur. J. Oper. Res.* **2022**, *303*, 857-876. [CrossRef]
22. Nodoust, S.; Pishvaee, M.S.; Seyedhosseini, S.M. Vehicle routing problem for humanitarian relief distribution under hybrid uncertainty. *Kybernetes* **2021**. [CrossRef]
23. Balcik, B.; Iravani, S.; Smilowitz, K. Multi-vehicle sequential resource allocation for a nonprofit distribution system. *IIE Trans.* **2014**, *46*, 1279–1297. [CrossRef]
24. Abazari, S.R.; Aghsami, A.; Rabbani, M. Prepositioning and distributing relief items in humanitarian logistics with uncertain parameters. *Socio-Econ. Plan. Sci.* **2021**, *74*, 100933. [CrossRef]
25. Mondal, A.; Roy, S.K. Multi-objective sustainable opened-and closed-loop supply chain under mixed uncertainty during COVID-19 pandemic situation. *Comput. Ind. Eng.* **2021**, *159*, 107453. [CrossRef]
26. Mohammadi, S.; Darestani, S.A.; Vahdani, B.; Alinezhad, A. 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.* **2020**, *148*, 106734. [CrossRef]
27. Zhong, S.; Cheng, R.; Jiang, Y.; Wang, Z.; Larsen, A.; Nielsen, O.A. Risk-averse optimization of disaster relief facility location and vehicle routing under stochastic demand. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *141*, 102015. [CrossRef]
28. Bruni, M.; Khodaparasti, S.; Beraldi, P. The selective minimum latency problem under travel time variability: An application to post-disaster assessment operations. *Omega* **2020**, *92*, 102154. [CrossRef]
29. Nadi, A.; Edrisi, A. Adaptive multi-agent relief assessment and emergency response. *Int. J. Disaster Risk Reduct.* **2017**, *24*, 12–23. [CrossRef]
30. Sidrane, C.; Kochenderfer, M.J. Closed-loop planning for disaster evacuation with stochastic arrivals. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 2544–2549.
31. Bi, C.; Pan, G.; Yang, L.; Lin, C.C.; Hou, M.; Huang, Y. Evacuation route recommendation using auto-encoder and markov decision process. *Appl. Soft Comput.* **2019**, *84*, 105741. [CrossRef]
32. Çelik, M.; Ergun, Ö.; Keskinocak, P. The post-disaster debris clearance problem under incomplete information. *Oper. Res.* **2015**, *63*, 65–85. [CrossRef]
33. Mills, A.F.; Argon, N.T.; Ziya, S. Dynamic distribution of patients to medical facilities in the aftermath of a disaster. *Oper. Res.* **2018**, *66*, 716–732. [CrossRef]

34. Secomandi, N. *Exact and Heuristic Dynamic Programming Algorithms for the Vehicle Routing Problem with Stochastic Demands*; University of Houston: Houston, TX, USA, 1999.
35. Secomandi, N. A rollout policy for the vehicle routing problem with stochastic demands. *Oper. Res.* **2001**, *49*, 796–802. [CrossRef]
36. Novoa, C.M. *Static and Dynamic Approaches for solving the Vehicle Routing Problem with Stochastic Demands*; Lehigh University: Bethlehem, PA, USA, 2005.
37. Novoa, C.; Storer, R. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *Eur. J. Oper. Res.* **2009**, *196*, 509–515. [CrossRef]
38. Thomas, B.W.; White Iii, C.C. Anticipatory route selection. *Transp. Sci.* **2004**, *38*, 473–487. [CrossRef]
39. Fan, J.; Wang, X.; Ning, H. A multiple vehicles routing problem algorithm with stochastic demand. In Proceedings of the 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 21–23 June 2006; Volume 1, pp. 1688–1692.
40. Goodson, J.C.; Ohlmann, J.W.; Thomas, B.W. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Oper. Res.* **2013**, *61*, 138–154. [CrossRef]
41. Goodson, J.C.; Thomas, B.W.; Ohlmann, J.W. Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transp. Sci.* **2016**, *50*, 591–607. [CrossRef]
42. Ulmer, M.W.; Mattfeld, D.C.; Hennig, M.; Goodson, J.C. A rollout algorithm for vehicle routing with stochastic customer requests. In *Logistics Management*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 217–227.
43. Ulmer, M.W.; Goodson, J.C.; Mattfeld, D.C.; Thomas, B.W. On modeling stochastic dynamic vehicle routing problems. *EURO J. Transp. Logist.* **2020**, *9*, 100008. [CrossRef]
44. Ulmer, M.W.; Soeffker, N.; Mattfeld, D.C. Value function approximation for dynamic multi-period vehicle routing. *Eur. J. Oper. Res.* **2018**, *269*, 883–899. [CrossRef]
45. Ulmer, M.W.; Goodson, J.C.; Mattfeld, D.C.; Hennig, M. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transp. Sci.* **2019**, *53*, 185–202. [CrossRef]
46. Ulmer, M.W. Horizontal combinations of online and offline approximate dynamic programming for stochastic dynamic vehicle routing. *Cent. Eur. J. Oper. Res.* **2020**, *28*, 279–308. [CrossRef]
47. Yu, X.; Gao, S.; Hu, X.; Park, H. A Markov decision process approach to vacant taxi routing with e-hailing. *Transp. Res. Part B Methodol.* **2019**, *121*, 114–134. [CrossRef]
48. Bertsekas, D.P.; Tsitsiklis, J.N.; Wu, C. Rollout algorithms for combinatorial optimization. *J. Heuristics* **1997**, *3*, 245–262. [CrossRef]
49. Secomandi, N. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Comput. Oper. Res.* **2000**, *27*, 1201–1225. [CrossRef]
50. Dror, M.; Laporte, G.; Trudeau, P. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transp. Sci.* **1989**, *23*, 166–176. [CrossRef]
51. Dror, M. Modeling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution. *Eur. J. Oper. Res.* **1993**, *64*, 432–441. [CrossRef]
52. Bertsekas, D.P. *Dynamic Programming and Optimal Control*, 3rd ed.; Athena Scientific: Belmont, MA, USA, 2011; Volume 2.
53. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-Dynamic Programming*; Athena Scientific: Nashua, NH, USA, 1996.
54. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998; Volume 22447.
55. Shannon, C.E. XXII. Programming a computer for playing chess. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1950**, *41*, 256–275. [CrossRef]
56. Samuel, A.L. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.* **1959**, *3*, 210–229. [CrossRef]
57. Samuel, A.L. Some studies in machine learning using the game of checkers. II—Recent progress. *IBM J. Res. Dev.* **1967**, *11*, 601–617. [CrossRef]
58. Barto, A.G.; Sutton, R.S.; Anderson, C.W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **1983**, *5*, 834–846. [CrossRef]
59. Sutton, R.S. Learning to predict by the methods of temporal differences. *Mach. Learn.* **1988**, *3*, 9–44. [CrossRef]
60. Watkins, C.J.C.H. *Learning from Delayed Rewards*. 1989. Available online: https://d1wqtxts1xzle7.cloudfront.net/50360235/Learning_from_delayed_rewards_20161116-28282-v2pwvq-with-cover-page-v2.pdf?Expires=1659006720&Signature=XMv610R4pgdMEva3Jg8e8SqjYPOgg~BcjROgGKK4dak2z5aUwWMbxqGanaYDj9GuKMWKjTsTAGRQilNeQEOOcHtP~52zthGvsGXmKoa60~jJA3qW6AKYyC1UsDQQX5K~NUZqgaSmRekMdhhrTY8SsZ2gFXj24-Me93ZIBL1GwKXqY~BYVKva1mfLKWagtRo4xOO4qOD3bltUG5r2jz2CxMwODZLB5NR8xQi3wWdddVRfr2GrThK08nvUwJD4QV~5jaydvc9YLAuLl3tmUAWlbPj20a0ioTkA3VneMHRMDHItoIfa88KKZC8SPhxtVK7r-iCfiUemnJfFDYzxrS~E~Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (accessed on 7 July 2021).
61. Tesauro, G. Practical issues in temporal difference learning. *Mach. Learn.* **1992**, *8*, 257–277. [CrossRef]
62. Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* **1994**, *6*, 215–219. [CrossRef]
63. Tesauro, G. Temporal difference learning and TD-Gammon. *Commun. ACM* **1995**, *38*, 58–68. [CrossRef]
64. Ghiani, G.; Manni, E.; Quaranta, A.; Triki, C. Anticipatory algorithms for same-day courier dispatching. *Transp. Res. Part E Logist. Transp. Rev.* **2009**, *45*, 96–106. [CrossRef]
65. Voccia, S.A.; Campbell, A.M.; Thomas, B.W. The same-day delivery problem for online purchases. *Transp. Sci.* **2019**, *53*, 167–184. [CrossRef]

66. Secomandi, N. Analysis of a rollout approach to sequencing problems with stochastic routing applications. *J. Heuristics* **2003**, *9*, 321–352. [CrossRef]
67. Powell, W.B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*; John Wiley & Sons: Hoboken, NJ, USA, 2007; Volume 703.
68. Bertazzi, L.; Secomandi, N. Faster rollout search for the vehicle routing problem with stochastic demands and restocking. *Eur. J. Oper. Res.* **2018**, *270*, 487–497. [CrossRef]
69. Goodson, J.C.; Thomas, B.W.; Ohlmann, J.W. A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *Eur. J. Oper. Res.* **2017**, *258*, 216–229. [CrossRef]
70. Zhao, Y.; Chen, X.; Jia, Q.S.; Guan, X.; Zhang, S.; Jiang, Y. Long-term scheduling for cascaded hydro energy systems with annual water consumption and release constraints. *IEEE Trans. Autom. Sci. Eng.* **2010**, *7*, 969–976. [CrossRef]
71. Bertazzi, L.; Bosco, A.; Guerriero, F.; Lagana, D. A stochastic inventory routing problem with stock-out. *Transp. Res. Part C Emerg. Technol.* **2013**, *27*, 89–107. [CrossRef]
72. Bertazzi, L.; Bosco, A.; Laganà, D. Managing stochastic demand in an inventory routing problem with transportation procurement. *Omega* **2015**, *56*, 112–121. [CrossRef]
73. Moin, N.H.; Halim, H.Z.A. Solving inventory routing problem with stochastic demand. In *AIP Conference Proceedings*; AIP Publishing LLC: Melville, NY, USA, 2018; Volume 1974, p. 020104. Available online: https://aip.scitation.org/doi/abs/10.1063/1.5041635 (accessed on 22 July 2022).
74. Secomandi, N.; Margot, F. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Oper. Res.* **2009**, *57*, 214–230. [CrossRef]
75. Ulmer, M.W. *Approximate Dynamic Programming for Dynamic Vehicle Routing*, 1st ed.; Operations Research/Computer Science Interfaces Series; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; Volume 61.
76. Mole, R.; Jameson, S. A sequential route-building algorithm employing a generalised savings criterion. *J. Oper. Res. Soc.* **1976**, *27*, 503–511. [CrossRef]
77. Solomon, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **1987**, *35*, 254–265. [CrossRef]
78. Clarke, G.; Wright, J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **1964**, *12*, 568–581. [CrossRef]
79. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
80. Chauhan, C.; Gupta, R.; Pathak, K. Survey of methods of solving tsp along with its implementation using dynamic programming approach. *Int. J. Comput. Appl.* **2012**, *52*, 12–18. [CrossRef]
81. Rand, G. 50 years of the savings method for vehicle routing problems. *Oper. Res. Manag. Sci. Today* **2014**, *41*, 14–15.
82. Nowak, M.A. *The Pickup and Delivery Problem with Split Loads*; Georgia Institute of Technology: Atlanta, GA, USA, 2005.
83. Paessens, H. The savings algorithm for the vehicle routing problem. *Eur. J. Oper. Res.* **1988**, *34*, 336–344. [CrossRef]
84. Larson, R.C.; Odini, A.R. Urban Operations Research. Massachusetts. 1999. Available online: http://web.mit.edu/urban_or_book/www/book/chapter6/6.4.12.html (accessed 7 July 2021).
85. Savelsbergh, M. A parallel insertion heuristic for vehicle routing with side constraints. *Stat. Neerl.* **1990**, *44*, 139–148. [CrossRef]
86. Anuar, W.K.; Lee, L.S.; Pickl, S. Benchmark dataset for multi depot vehicle routing problem with road capacity and damage road consideration for humanitarian operation in critical supply delivery. *Data Brief* **2022**, *41*, 107901. [CrossRef]
87. Anuar, W.K.; Lee, L.S. MDDVRPSRCV1_Test_Instance. Dataset in Mendeley Repository. 2021. Available online: https://www.sciencedirect.com/science/article/pii/S2352340922001135 (accessed on 7 July 2021).
88. Shapiro, S.S.; Wilk, M.B. An analysis of variance test for normality (complete samples). *Biometrika* **1965**, *52*, 591–611. [CrossRef]
89. Zaiontz, C. Real Statistics Using Excel. 2015. Available online: https://www.real-statistics.com (accessed on 11 May 2021).