

Multi-Route Multi-Vehicle Dial-a-Ride Problem: a Comparison Study

Philipp Triebold
 Bundeswehr University Munich
philipp.triebold@unibw.de

Rudy Milani
 Bundeswehr University Munich
rudy.milani@unibw.de

Stefan Pickl
 Bundeswehr University Munich
stefan.pickl@unibw.de

Abstract

The Multi-Vehicle Dial-a-Ride Problem (MVDARP) consists of routing a fixed number of capacitated vehicles from their initial positions to multiple pick-up locations and, subsequently, to the delivery depots, while minimising the overall costs of the transportation. In this paper, we propose an extended version of the MVDARP which considers multiple consecutive routes: the Multi-Route Multi-Vehicle Dial-a-Ride Problem (MRMVDARP). In this case, a scalarized multi-objective MIP formulation, addressing at the same time the costs of the routing and the duration of the job, is examined together with a Markov Decision Process (MDP) conceptualisation. Thus, the results found using an exact solver applied to the former problem are compared to the solutions of the MDP achieved by Q-learning. The computational experiments tested in small grid-world environments show that the performance of the Reinforcement Learning algorithm outperforms the exact solver in the larger maps tested. This preliminary evaluation indicates a promising research direction that could be explored in the future.

Keywords: Dial-a-Ride Problem, Multi-Route Multi-Vehicle Dial-a-Ride Problem, Mixed Integer Programming, Reinforcement Learning, Q-learning

1. Introduction

The classical dial-a-ride problem (DARP) is one of the fundamental optimisation problems in the logistics part of Operations Research (OR). One of the first papers (Guenther, 1970) discussing the DARP is concerned with tests conducted by the Ford Motor Company for public transport companies in small to medium-sized towns in the US in the late 1960s. One

of the more complex definitions introduced is the Many-to-Many problem, in which many passengers want to get to many individual locations. One of the first solutions to this complex Many-to-Many problem came in a 1980 paper which provided a solution with dynamic programming (Psaraftis, 1980). Another DARP definition is the Many-to-One problem, which is many passengers in different locations all want to get to the same location, which is similar to the direction of this paper. One of the first solutions to the Many-to-One DARP problem comes in a 1973 paper that discusses the demand for responsive transportation service for a university (Gibson and White, 1972).

The Multi-Vehicle Dial-a-ride problem (MVDARP) is an extension of the DARP. It extends the number of capacitated vehicles trying to fulfil the transportation requests from the passengers. One application for the Many-to-One problem is meal delivery with autonomous drones (Liu, 2019). In that paper the authors talk about solving an MIP model for efficiently dispatching and routing autonomous delivery drones. This can be aligned with the Many-to-One approach since many customers get food delivered from one restaurant. Alternatively, this application can be similar to another DARP definition the Many-to-Few, where instead of many customers requesting food from one restaurant, they request food from a few restaurants. Importantly, the number of restaurants would be much smaller than the number of customers.

A more classical approach is the transportation of passengers with reduced mobility in airports (Reinhardt et al., 2013). The authors describe the problem of having many passengers at an airport who need to get to their plane. Their greedy heuristic solution to their modification of the DARP can find good solutions within minutes enabling dynamic scheduling.

Extending the DARP is the approach of including the fixed route public transport into the DARP to formulate the Integrated-Dial-a-Ride problem (IDARP) (Häll et al., 2009). They extend the DARP in a way that one part of each trip may be carried out by a fixed route public transport system. A further extension to the IDARP came with constraining the use of the public transport system to its timetables, thus eliminating long wait times for the passengers at the transfer stops (Posada et al., 2017).

In this paper we introduce the Multi Route Multi Vehicle Dial-a-Ride problem (MRMVDARP) which is a further extension of the MVDARP, introducing the option for the vehicles to do multiple routes. An application for the MRMVDARP could be in precision agriculture when considering the problem of selective pesticide spraying with drones (Srivastava et al., 2020). The authors describe a Problem, where they are considering the battery and pesticide level in the route planning for an Unmanned Aerial Vehicle (UAV) spraying pesticide selectively on affected areas of a field. This problem could also be considered as an MRMVDARP, since it is likely, that the UAVs will neither have the battery nor pesticide capacity to spray an entire field.

For the MRMVDARP we propose a Mixed-Integer-Programming (MIP) formulation with a new objective function and a Markov-Decision-Process (MDP) formulation. These are compared against each other with computational experiments and provide, to the best of our knowledge, a new solution to this novel problem.

In this paper we will first discuss background information for reinforcement learning (RL), then formulate the MVDARP and the MRMVDARP, followed by the computational experiments and finally a conclusion and outlook.

2. Background

The theoretical background, required to fully comprehend the methodology applied in this paper, is based on the RL framework. Specifically, the scenario that will be studied concerns a discrete problem. Hence, the basic notions related to RL are introduced in the following section.

The general setting for RL consists of discrete time steps $t = 0, 1, \dots, T$ decision processes where an agent has to take action $a_t \in \mathbb{A}$ given a state $s_t \in \mathbb{S}$ provided by the environment (Sutton and Barto, 2018);

with \mathbb{A} and \mathbb{S} denoting respectively the set of the actions and the states. Consequent to the choice performed by the agent, the environment produces a new state $s_{t+1} \in \mathbb{S}$ and a numerical reward $r_{t+1} \in \mathbb{R}$. In detail, the next state is computed through a transition function $\mathbb{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow [0, 1]$ that assign the probability $\mathbb{T}(s_t, a_t, s_{t+1})$ of passing from state s_t to s_{t+1} by performing action a_t . In this way, also the probabilistic environments can be taken into account, where the deterministic problems can be identified by a singular 1 value for each combination of state-action pairs. Moreover, the transition function \mathbb{T} has to satisfy some requirements like the normalization condition and the Markov property, i.e., the transition to the next state only depends on the last state and last action and not on the precedent information. Similarly to the next state computation, the reward obtained is calculated using a reward function $r : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow \mathbb{R}$ that associates to the state, action, and next state the following gain received by the agent, i.e., $r(s_t, a_t, s_{t+1}) = r_{t+1}$. Finally, before introducing the objective function, it is necessary to present the discounting factor $\gamma \in [0, 1]$, which can scale the interest of the agent on rewards appearing later in the episode. In this way, it can be defined the return $G_t = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1}$, representing the total discounted reward obtained from time step t . The goal of the RL agent is to maximize the return. With these data, it can be defined the MDP as the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, r, \gamma)$, where each element has been already explained.

The task of the RL algorithms involves the creation of an optimal policy $\pi^* : \mathbb{S} \rightarrow \mathbb{A}$, which is a function that maps the states into actions and is capable of attaining the maximum of the return if followed. Nevertheless, it is possible to work on a different mathematical object that can summarise the information of the MDP: the state-action value function or Q-value function $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1} | s_t = s, a_t = a]$. This fundamental function evaluates the expected return achieved following policy π from state s and performing action a . Thus, exploiting the adoption of this particular function can lead to the definition of an optimal policy by looking at the action that can maximize the Q-value associated with the actual state. Thereby, multiple approaches consider approximating the state-action value function in order to subsequently derive the optimal policy.

An example of these methodologies is *Q-learning* (Watkins and Dayan, 1992). This off-policy model-free algorithm is based on learning an estimation of the Q-functions through a tabular representation. Rigorously, at the beginning of the training, a matrix defined as Q-table $Q(\cdot, \cdot) \in \mathbb{R}^{|\mathbb{S}| \times |\mathbb{A}|}$, is initialised at

zero. Then, after reading state s_t and operating action a_t , the Q-table is updated using the following rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in \mathbb{A}} Q(s_{t+1}, a) - Q(s_t, a_t) \right],$$

where $\alpha \in (0, 1]$ indicates the learning rate. The updating rule used is found from the Bellman's principle of optimality which characterize the value functions for optimal policies (Bellman, 1957).

3. Problem Formulation

In this section, the focus is centred on the mathematical definition of the problem tackled. In particular, the addressed mathematical program is derived from the classical dial-a-ride problem (DARP) through the inclusion of multiple paths performed by a fleet of vehicles. In this way, it is introduced the Multi Route Multi Vehicle DARP (MRMVDARP). Subsequently, an MDP formulation of this problem is developed to adopt an RL approach in finding the optimal solution.

3.1. MIP formulation

The mathematical formulation of the MVMPDARP takes inspiration from the classical DARP definition from (Cordeau and Laporte, 2007). Practically, the problem faced in this paper consists of picking up resources from multiple locations and bringing them to a singular depot position through the use of a fleet of vehicles. Moreover, the vehicles have limited capacities, leading to doing multiple consecutive routes to accomplish the task. To solve optimally this problem, the picked-up items have to be transported as quickly as possible to the depot and cover the overall shortest distance.

After this pragmatic description of the problem, the MIP model for the MVMPDARP can be introduced. Specifically for this mathematical program, the notation used is summarised in Table 1, Table 2, and Table 3 where respectively are listed and briefly explained parameters, the sets, and variables used. The MIP formulation is shown in the following:

Table 1. Description of the parameters used in this paper.

Parameter	Definition
n	Number of users
0	Index of the virtual node associated to the starting position
$n + 2$	Index of the virtual node associated to the depot position
$Q^{k,m}$	Maximum capacity of vehicle m on route k
$T^{k,m}$	Maximum time duration for route k with vehicle m
q_i	Load of node i
d_i^m	Non-negative service duration of node i for vehicle m ($d_0^m = d_{n+2}^m = 0, \forall m \in M$)
$[e_i, l_i]$	Time window for the service of node i with earliest time e_i and latest time l_i
c_{ij}^m	Cost of traversing arc (i, j) with vehicle m
t_{ij}^m	Time necessary for traversing arc (i, j) with vehicle m
L	Maximum time for a user to reach the depot
$\widetilde{M}_{ij}^{k,m}, \overline{M}_{ij}^{k,m}$	Constants for the big-M constraints
w_1, w_2	Weights for the scalarisation of the objective function

$$\min \left(w_1 \sum_{m \in M} \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{i,j}^{k,m} x_{i,j}^{k,m} + \right. \quad (1)$$

$$\left. w_2 \max_{m \in M} \sum_{k \in K} B_{n+2}^{k,m} \right)$$

$$\text{s.t.} \quad \sum_{m \in M} \sum_{k \in K} \sum_{j \in N} x_{i,j}^{k,m} = 1 \quad \forall i \in P \quad (2)$$

$$\sum_{j \in N \setminus \{n+2\}} x_{0,j}^{k,m} - x_{n+1,n+2}^{k,m} = 0 \quad (3)$$

$$\forall k \in K, m \in M$$

$$\sum_{j \in N \setminus \{0\}} x_{0,j}^{k,m} = 1 \quad \forall k \in K, m \in M \quad (4)$$

$$\sum_{j \in N} x_{j,i}^{k,m} - \sum_{j \in N} x_{i,j}^{k,m} = 0 \quad (5)$$

$$\forall i \in P \cup D, k \in K, m \in M$$

Table 2. Description of the sets used in this paper.

Set	Definition
P	Set of pick up nodes, i.e., $P = \{1, \dots, n\}$
D	Set of drop off nodes, i.e., $D = \{n + 1\}$
N	Complete set of all nodes, i.e., $N = P \cup D \cup \{0, n + 2\}$
K	Set of routes
M	Set of vehicles

Table 3. Description of the variables used in this paper.

Variable	Definition
$x_{ij}^{k,m}$	Binary variable checking whether vehicle m traverses arc (i, j) during route k
$B_i^{k,m}$	Leaving time from node i of vehicle m during route k
$Q_i^{k,m}$	Capacity of vehicle m during route k after leaving node i
$L_i^{k,m}$	Ride time of user i using vehicle m in route k

$$\sum_{i \in N \setminus \{n+2\}} x_{i,n+2}^{k,m} = 1 \quad \forall k \in K, m \in M \quad (6)$$

$$B_j^{k,m} \geq B_i^{k,m} + d_i^m + t_{i,j}^m - \overline{M}_{i,j}^{k,m} (1 - x_{i,j}^{k,m}) \quad \forall i \in N, j \in N, k \in K, m \in M \quad (7)$$

$$B_j^{k,m} \leq B_i^{k,m} + d_i^m + t_{i,j}^m + \overline{M}_{i,j}^{k,m} (1 - x_{i,j}^{k,m}) \quad \forall i \in N, j \in N, k \in K, m \in M \quad (8)$$

$$Q_j^{k,m} \geq Q_i^{k,m} + q_j^m - \widetilde{M}_{i,j}^{k,m} (1 - x_{i,j}^{k,m}) \quad \forall i \in N, j \in N, k \in K, m \in M \quad (9)$$

$$Q_j^{k,m} \leq Q_i^{k,m} + q_j^m + \widetilde{M}_{i,j}^{k,m} (1 - x_{i,j}^{k,m}) \quad \forall i \in N, j \in N, k \in K, m \in M \quad (10)$$

$$L_i^{k,m} = B_{n+1}^{k,m} - (B_i^{k,m} + d_i^m) \quad \forall i \in P, k \in K, m \in M \quad (11)$$

$$B_{n+2}^{k,m} - B_0^{k,m} \leq \mathbf{T}^{k,m} \quad \forall k \in K, m \in M \quad (12)$$

$$x_{i,j}^{k,m} \in \{0, 1\} \quad \forall i \in N, j \in N, k \in K, m \in M \quad (13)$$

$$e_i \leq B_i^{k,m} \leq l_i \quad \forall i \in N, k \in K, m \in M \quad (14)$$

$$\max\{0, q_i\} \leq Q_i^{k,m} \leq \min\{\mathbf{Q}^{k,m}, \mathbf{Q}^{k,m} + q_i\} \quad \forall i \in N, k \in K, m \in M \quad (15)$$

$$L_i^{k,m} \leq \mathbf{L} \quad \forall i \in P, k \in K, m \in M. \quad (16)$$

The objective function and all the constraints of the problem above are now thoroughly explicated. In Equation 1, the objective function to optimize is a combination of two components: the first one represents the costs of all movements for all vehicles, and the second one indicates the latest time of delivery for all the routes and vehicles. Thereby, it has to be minimised at the same time the costs of the movements and the time of completion of the task. Through the scalarization with the weights w_1 and w_2 , both the tasks can be modelled.

Moving to the description of the constraints, Equation 2 ensures that each pick-up location is visited only once, while Equation 3 forces all the vehicles that came to a specific pick-up location to deposit the item in the depot virtual node. The constraint in Equation 4 indicates the starting position at the virtual node 0 and eliminates the possibility of self-loops in the initial location. Equation 5 represents the classical conservation of the flow for each node. With Equation 6 it is imposed the arrival to the virtual depot node of all the routes and vehicles. Then, big-M constraints have been used in Equations 7–8 for the correct identification of the leaving time and in Equations 9–10 for the capacity updates. After the definition of the proper values of the leaving times, it is helpful to calculate the ride time of each user through Equation 11 and check that the maximum travel time is respected with Equation 12. Finally, the variables' boundaries are explicated in constraints Equations 13–16.

With this set of constraints, the general problem is well-posed. Nevertheless, it is possible to include a variety of well-known valid inequalities to improve the computational performances of exact solvers (Cordeau, 2006). However, first, it can also be constrained the number of total routes considered, i.e., the cardinality of the route set $|K|$. In fact, in the optimal solution, the number of total routes can be bounded as follows:

$$|K| \leq \left\lceil \frac{\sum_{i \in P} q_i}{|M| \cdot \min_{m \in M, k \in K} \mathbf{Q}^{k,m}} \right\rceil \quad (17)$$

where the smallest integer that is not less than the total demand is divided by the number of vehicles considered, multiplied by the minimum capacity available. Thereby, the maximum number of the routes that are required to optimally solve MVMPDARP corresponds to the right-hand side of Equation 17.

Moving to the proper valid inequalities, it is possible to eliminate all the self-loops, moving out from the virtual depot or going to the virtual start node, and going back to the same pick-up node through the following set

of constraints:

$$\sum_{j \in N} x_{j,j}^{k,m} = 0 \quad \forall k \in K, m \in M \quad (18)$$

$$\sum_{i \in N} x_{n+2,i}^{k,m} = 0 \quad \forall k \in K, m \in M \quad (19)$$

$$\sum_{i \in N} x_{i,0}^{k,m} = 0 \quad \forall k \in K, m \in M \quad (20)$$

$$\sum_{k \in K} \sum_{m \in M} x_{i,j}^{k,m} + x_{j,i}^{k,m} \leq 1 \quad \forall i \in P, j \in P. \quad (21)$$

All of these sub-tour elimination constraints are reducing the feasible region but not eliminating any optimal solution.

Moreover, it is useful to fix to zeros the movements towards the deposit node from the non-pick-up locations and the trips going from a pick-up position to the virtual node with the following constraints:

$$\sum_{i \in N \setminus P} x_{i,j}^{k,m} = 0 \quad \forall j \in D, k \in K, m \in M \quad (22)$$

$$\sum_{i \in N \setminus (D \cup \{0\})} x_{i,n+2}^{k,m} = 0 \quad \forall k \in K, m \in M. \quad (23)$$

Indeed, with the aforementioned set of constraints, the number of variables can be lowered and the problem can be simplified (Cordeau, 2006).

A different way of increasing the speed for finding an optimal solution is to reduce the symmetries present in the problem. To this end, it is possible to reduce the number of optimal solutions considered by forcing all the vehicles to start their respective paths at time 0 with the following constraint:

$$B_0^{k,m} = 0 \quad \forall k \in K, m \in M. \quad (24)$$

Although this approach leads to restricting the set of optimal solutions, it reduces the search space significantly by imposing a specific condition on the optimal solution.

The final MIP employed for the computational analysis consists of the objective function in Equation 1 and all the constraints in Equations 2–24.

3.2. MDP Formulation

After introducing the MIP formulation in the previous section, now we focus on the formulation of the MVMPDARP as an MDP. Specifically for this aim, a custom $w \times h$ grid-world environment representing a factory has been created, where w is the width and

h is the height of the map. In this context, multiple tiles denoting the stations are identified as the pick-up locations, while the starting position represents also the depot. Hence, the total number of pick-up locations is $|P| = w \cdot h - 1$ since, for hypothesis, the starting position does not present any item to move.

Given the presence of a fleet of vehicles, it has been adopted a centralized representation for the states to simplify the learning process. With this gimmick, just an agent can move simultaneously multiple vehicles knowing the full information of the overall problem. Thus, the states are denoted, at each time step t , by the position $p_{m,t}$ and capacity $c_{m,t}$ of each vehicle m , and the remaining items that have to be picked up M_t , i.e., $s_t = (p_{1,t}, c_{1,t}, \dots, p_{|M|,t}, c_{|P|,t}, M_t)$.

Also the action representation couples together all the operations of each singular vehicle, leading to an action vector of m components describing the action taken by each worker. Formally, the action set is composed of vectors $a_t = (a_{1,t}, \dots, a_{|M|,t})$, where $a_{m,t}$ denotes the action selected for the vehicle m at time step t . The possible atomic actions that can be performed are movement to neighbouring nodes or picking or dropping off the items. All the transitions of the environment are deterministic, and follow the usual physical limitations of the map considered. Therefore, when a worker tries to move outside of the boundaries of the environment no changes in the next state instantiation will appear. In the same manner, if the agent tries to pick up an item when the maximum capacity $Q^{k,m}$ of vehicle m is reached, then the next state features are not changed. Nevertheless, this unnecessary operation will lead to losing time and earning a penalty reward, thus, invalidating any possibility of reaching optimality.

The agent receives a reward of -1 at each time step. Therefore, it is encouraged to complete the task promptly, and making wrong decisions will affect the final accumulated reward. To learn properly the mission of picking up items and dropping them off in the depot, a reward of 10 is associated with the correct accomplishments of these two works. Moreover, a penalty of -5 is provided when the pick-up and drop-off actions are done in erroneous situations, e.g., taking an item from an empty tile or depositing it in a pick-up location.

With this information, the MDP structure for the MVMPDARP is delineated. Thus, in the following section, the comparisons between the two formulations in a set of computational experiments are discussed.

Table 4. Declaration of the Instances considered in the Computational Experiments

Instance	$ M $	$(Q^1, \dots, Q^{ M })$	w	h
m2_c2_2-2	2	(2, 2)	2	2
m2_c2_2-3	2	(2, 2)	2	3
m2_c2_2-4	2	(2, 2)	2	4
m2_c2_3-2	2	(2, 2)	3	2
m2_c2_3-3	2	(2, 2)	3	3
m2_c2_3-4	2	(2, 2)	3	4
m2_c2_4-2	2	(2, 2)	4	2
m2_c2_4-3	2	(2, 2)	4	3
m2_c2_4-4	2	(2, 2)	4	4

4. Computational Experiments

To evaluate the models on different map sizes, experiments were conducted on a Server running Ubuntu 22.04 with two AMD EPYC 7763 processors and 1024GB RAM. The MIP model was solved using IBM CPLEX 22.1.1, working with the default configuration apart from running on 128 Threads of the 256 available due to a massive performance penalty when splitting the load over multiple processors. The training environment for the RL utilised Python 3.10.12, along with the numpy 1.26.4 and gymnasium 0.29.1 modules. The Python script was running strictly single-threaded.

The set of instances evaluated in the test phase is reported in Table 4. To enhance the readability, an ID showing the parameter settings is linked at each instance. In detail, the minimum dimensions of the rectangular maps considered span from a minimum of 2 to a maximum of 4. For the sake of simplicity, the maximum number of vehicles involved in the computational experiments has been fixed to 2 as well and the corresponding maximum capacity is set to 2 for each vehicle.

For the RL agent, the Q-table is initialised with zeros and has a shape of $(2^{w \times h}, |M| \times [w \times h], |M| \times [Q + 2], 6, |M|)$, where $|M|$ is the total number of robots, w is the width of the map, h is the height of the map, and Q is the maximum capacity of the robots. For instance, for a 3×3 map with two robots and a capacity of 2, the Q-table shape is (512, 9, 9, 4, 4, 6, 2).

The training was conducted over 1,500,000 episodes with a learning rate $\alpha = 0.1$ and a discount factor $\gamma = 0.99$. The exploration rate started at $\epsilon_{start} = 1.0$ and decayed to $\epsilon_{end} = 0.1$ with a decay rate of $\epsilon_{decay} = 400,000$. The exploration rate ϵ for each episode i was calculated using equation 25.

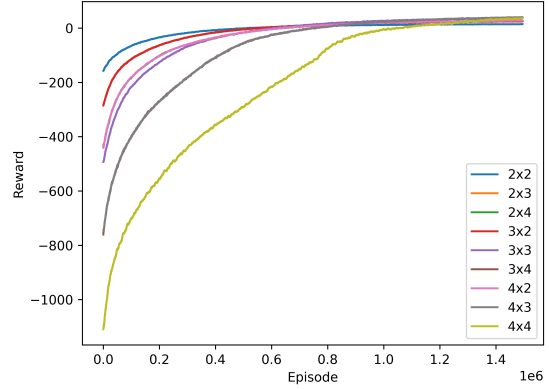


Figure 1. Reward compared to the episodes(Smoothed over 5000 Episodes)

$$\epsilon = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) * e^{-\frac{i}{\epsilon_{decay}}} \quad (25)$$

4.1. Results

The results of the MIP and the MDP experiments based on the map ID are summarised in Table 5. The exact MIP solutions are effective for smaller problem sizes but become impractical as the problem scale increases. Specifically, all map sizes up to 3×4 were solvable within one hour, while the 4×4 experiment was terminated after one hour (highlighted in red). At this point, memory usage reached approximately 114GB, significantly exceeding that of the smaller maps. The Validation column of the MDP is calculated by subtracting the reward of the RL agent from the maximum amount of points available (number of fields times dropoff reward), resulting in a result that is comparable to the minimum number of steps calculated by the MIP.

The results of the MDP experiments are promising, demonstrating the ability to find optimal solutions for all test cases except the largest map, for which near-optimal solutions were obtained. The gaps for the RL are zero, although the validation result exceeds the minimum steps because the RL agent considers picking up and dropping off cargo as individual time steps. Combined with the constraint that the robots cannot remain idle, this sometimes necessitates additional steps for one robot while the other completes its tasks. This is exemplified in the *m2_c2_4-4* experiment, where robot 1 undertakes a longer route, picking up from 8 fields. Ideally, this would require 28 steps, but in this

Table 5. Comparison of results from the MIP and MDP experiments

Instance	Steps	MIP			MDP			Gap [%]
		Optimal	Time [s]	Memory [MB]	Validation	Time [s]	Memory [MB]	
m2_c2_2-2	6	10	0.21	22.27	10	642.5	0.1875	0.00
m2_c2_2-3	12	18	1.03	37	14	1095	1.6875	0.00
m2_c2_2-4	20	30	9.53	63.58	22	1642.4	12	0.00
m2_c2_3-2	12	18	2.24	36.09	14	1102.3	1.6875	0.00
m2_c2_3-3	22	34	3.51	76.47	24	2012	30.375	0.00
m2_c2_3-4	36	54	101.76	553.67	38	30892.1	432	0.00
m2_c2_4-2	20	34	3.89	64.17	22	1638.9	12	0.00
m2_c2_4-3	36	54	108.63	674.19	38	30895.3	432	0.00
m2_c2_4-4	56	84	3610.48	114642.13	60	48735.1	12288	3.57

Table 6. Results of the reinforcement learner

Total positive return	Best validation	Gap	Sparsity
30	20	0	13.70%
50	36	0	20.27%
70	48	0	19.72%
50	36	0	19.83%
80	56	0	18.19%
110	72	0	6.20%
70	48	0	19.93%
110	72	0	6.50%
150	90	2	2.23%

experiment, the robot made an erroneous move into a wall, requiring the other robot to compensate in addition to the two extra steps it has to do since it only picks up 7 fields.

The learning time for the RL agent even for small maps is quite long, but this could be reduced since most maps reach a steady point after much less than the 1.5 million episodes (see Figure 1)

Memory statistics, presented in Table 5, indicate the theoretical maximum memory requirements of the Q-Table in megabytes (MB). However, the data reveals a high degree of sparsity, with values as low as 2.2% for the *m2_c2_4-4* experiment (see Table 6). This high sparsity suggests that utilising a library capable of handling sparse tensors could significantly reduce memory usage, potentially to as low as 274 MB. This approach would enable the Q-Table’s benefits to scale effectively with increasing map sizes, making it feasible to compute maps larger than 4×4 , which is not possible with exact solvers.

5. Conclusion and Future Works

In this paper, we have extended the classical definition of the DARP formulating the novel MRMVDARP. The proposed MRMVDARP extends

and modifies the classical DARP by consolidating all drop-off locations into a single point and allowing a vehicle to move multiple times after reaching the depot.

While a sparse Q-Table could potentially handle larger maps in tabular Q-learning, it remains a suboptimal solution. A promising direction in reinforcement learning involves methods such as deep Q-networks (DQN), which abstract the Q-Table into a neural network, potentially offering a more general solution. Currently, agents are trained on specific configurations of map size, capacity, and the number of robots. We assume that with larger maps and constant capacities, recognisable patterns will emerge that can be learned by DQN agents. Another avenue within reinforcement learning is multi-agent reinforcement learning. This could be realised as either a competitive or collaborative problem, with each having potential benefits. Multi-agent reinforcement learning seems like an obvious choice, considering the amount of independence that the robots have already.

Exact methods struggle a lot to solving larger problems (Ho et al., 2018), which is why often a heuristic is applied. Adapting such heuristics to the model we’ve proposed represents another promising solution.

References

- Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, 679–684.
- Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations research*, 54(3), 573–586.
- Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of operations research*, 153, 29–46.
- Gibson, J., & White, R. A. (1972). *Dial-a-bus for a university: Demand response service in a*

- many-to-one environment* (tech. rep.). SAE Technical Paper.
- Guenther, K. W. (1970). The mansfield dial-a-ride experiment.
- Häll, C. H., Andersson, H., Lundgren, J. T., & Värbrand, P. (2009). The integrated dial-a-ride problem. *Public Transport, 1*, 39–54.
- Ho, S. C., Szeto, W. Y., Kuo, Y. H., Leung, J. M., Petering, M., & Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological, 111*, 395–421. <https://doi.org/10.1016/j.trb.2018.02.001>
- Liu, Y. (2019). An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Computers & Operations Research, 111*, 1–20. <https://doi.org/https://doi.org/10.1016/j.cor.2019.05.024>
- Posada, M., Andersson, H., & Häll, C. H. (2017). The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport, 9*, 217–241.
- Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science, 14*(2), 130–154.
- Reinhardt, L. B., Clausen, T., & Pisinger, D. (2013). Synchronized dial-a-ride transportation of disabled passengers at airports. *European Journal of Operational Research, 225*(1), 106–117. <https://doi.org/https://doi.org/10.1016/j.ejor.2012.09.008>
- Srivastava, K., Pandey, P. C., & Sharma, J. K. (2020). An approach for route optimization in applications of precision agriculture using uavs. *Drones, 4*(3). <https://doi.org/10.3390/drones4030058>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning, 8*, 279–292.