Context-based Learning of Adaptive Vehicle Dispatching Strategies

Gerd Heiserich[1] and **Ludger Overmeyer**[2]

Abstract. The reinforcement learning framework has been used for finding and improving solutions to complex scheduling and resource allocation problems. However, in many real-world applications, state or action sets become very large due to the curse of dimensionality. This leads to approaches involving an approximation of the value function rather than a complete representation in a table. This contribution shows an application of reinforcement learning for a scenario in internal logistics involving multiple transportation vehicles operating in a common workspace. We show a solution for learning an adaptive vehicle dispatching strategy, which is suitable for a wide range of environmental conditions. This solution uses a context-based state representation that is scalable and allows a tabular representation of the value function even for large problem instances. The performance of the learned behaviour is evaluated by using a simulation model and is compared to common heuristic dispatching strategies.

1 INTRODUCTION

The ongoing trend towards mass customisation with an increasing number of product variants leads to the objective of agile and transformable systems in both production and transportation technology [17]. Quickly changing requirements have an impact on internal and external transportation processes, because production sites and machinery are generally served using just-in-time and just-in-sequence concepts, which hardly allow any temporal tolerances. For this reason, future logistic systems have to offer short reaction times and must react autonomously to unplanned scenarios.

In a supply network, numerous unanticipated events can occur, which potentially affect the entire system. Examples include failure of production equipment, changing amounts of demanded goods or uncertain transportation times due to traffic conditions. Finding optimal schedules for a complex structure with multiple participants such as a supply network is most often intractable due to the high number of degrees of freedom. Some authors even claim that a totally integrated supply chain management as a paradigm is generally not a viable approach [3].

Scheduling problems in general are known to be NP-hard. For this reason, the algorithms used for large real-world applications do not try to find a global optimum, but rather look for feasible solutions. Recently, methods from artificial intelligence have successfully been used for solving complex planning and scheduling problems in industrial applications. Examples include the stochastic lot scheduling problem [18] and job sequencing for a given setup of production machinery [4].

Several research groups have specifically examined problems arising in transportation and logistics. The aim is to create transportation systems which are able to cope with variable environmental conditions without the need for reconfiguration or reprogramming. This leads to two major directions of research: On one hand, mechanical and control issues are covered. The aim is to construct flexible and recombinable components capable of performing a variety of tasks. Examples for this direction of research are flexible conveyor modules as shown in [13], where the modules can be arbitrarily recombined and are able to autonomously set up the required configuration settings. An innovative mechanical concept using small-scaled modules for the transportation of packaged goods is introduced in [16]. A control architecture for transportation systems, which is based on hierarchical finite state machines with transition functions on different abstraction levels, is described in [15].

The second direction of research concentrates on the operational view of transportation systems in terms of a more abstract description of behaviour. Discussed issues include decision making and routing for vehicles in stochastic environments. This is closely related to the stochastic vehicle routing problem (SVRP), where a continuous appearance or cancellation of assignments is possible (see for example [5], [6]). In [20] a method for automatic adaptation of a decision-making algorithm for servicing stochastic customer requests is described. [19] shows a distributed routing algorithm for a fleet of vehicles which have to process a certain workload in a logistic net.

All the work mentioned above deals with external transportation processes with typical travel times in the range of several hours. In contrast, this article considers an internal transportation scenario, which is found at warehouses or dispatching points of shipping agencies. In these cases, transportation times between points of delivery are a few minutes at most. Moreover, dispatching points often have to deal with uncertain information about arrival times of inbound trucks. For both reasons, continuous re-planning on an operational level is not a feasible approach. The objective of this work is to use a learning approach to establish an adaptive dispatching strategy that makes online decisions based on the current state of the system; "adaptive" meaning that the system is able to cope with stochastic assignments at changing input rates.

[1] shows the portions of time of each partial process in a production and delivery chain. According to this, waiting times represent as much as 75 % of the total time and thus have by far the largest influence. This implies that there is a considerable potential for optimisation of transportation operation.

In the following section, the fundamental ideas of reinforcement learning are briefly reviewed. Section 3 gives a detailed explanation

[1] Institute of Transport and Automation Technology (ITA), Leibniz University of Hannover, Germany, email: gerd.heiserich@ita.uni-hannover.de
[2] Institute of Transport and Automation Technology (ITA), Leibniz University of Hannover, Germany, email: ludger.overmeyer@ita.uni-hannover.de

of the considered application scenario. After that, the applied learning algorithm for finding a dispatching strategy is discussed. The main contribution of this article is the development of a general state representation for the application, which makes the approach scalable to large problem instances. Finally, simulation results for the learned dispatching strategy in comparison to heuristic strategies are presented.

2 REINFORCEMENT LEARNING

The reinforcement learning framework deals with a class of problems in which a system - often called an agent - is operating in an environment while aiming at achieving a desired goal. At this point, we will just introduce the basic formula symbols and definitions that will be used throughout this article. Excellent introductions to theoretical foundations, methods and applications can be found in [7] and [21].

Reinforcement learning problems are modelled as Markov Decision Processes (MDP). This means the agent can reside in one state from a set of possible states. In each of these states, it can chose from a number of possible actions, which take it to another state. Actions are associated with rewards which contain information about their utility for the agent. The state transitions possess the Markov property: In a known system state, all the transition properties to following states depend only on the current state, and not on any previous states or chosen actions. Formally, an MDP is described by a tuple $(S,\ A,\ p,\ r)$ with

- S : set of system states
- A : set of actions
- $p\ :\ S \times A \times S \rightarrow [0, 1]$: transition probabilities
- $r\ :\ S \times A \rightarrow \mathbb{R}$: reward function

If the agent is in a state $s \epsilon S$ and executes an action $a \epsilon A$, it will proceed to state s' with the probability $p(s, a, s') = P(s' \mid s, a)$ and will receive the reward $r(s, a)$.

The aim is to find a policy $\pi\ :\ S \rightarrow A$ which is a function that maps a system state to an action. If a policy π^* maximizes the total reward over a long period of time, it is called optimal. There can be more than one policy to meet this condition. For evaluation of policies, the majority of approaches estimate a value function to describe the value of states or state-action pairs. This value function is denoted by

$$Q:\ S \times A \rightarrow \mathbb{R}$$

and returns the expected value of the future reward for taking an action $a\ \epsilon\ A$ in a state $s\ \epsilon\ S$ and following the policy π thereafter. For small state and action sets, the value function can be represented as a complete table with one entry for each state-action pair. For many real applications, states and actions are expressed as n-dimensional vectors of some features. This has the consequence that state and action sets grow exponentially in the number of components n. This effect is often called the curse of dimensionality and is a considerable challenge when solving real-world problems.

A possible approach to find the value function $Q(s, a)$ is to estimate it from experience. In this case, the learning system keeps an average value of all the received rewards for each state-action pair. This average will converge to the true value if each state-action pair is visited often enough. Estimation methods involving averaging over samples are referred to as Monte Carlo methods. Other approaches are based on learning new estimates based on existing estimates.

There is also a considerable amount of research concerning transferring these methods to the multi-agent case, i.e. to learn a behaviour for a group of several cooperating learning systems (see [8], [9]). If all agents can observe their own and all other agents' actions, the problem can be reduced to the single-agent case. In our work, we will explicitly allow communication between subsystems. This leads to a global improvement of the behaviour without the need for a central planning strategy.

3 APPLICATION SCENARIO

3.1 Principle of Cross-docking

As an application scenario for the investigations presented here, a cross-docking distribution centre is considered. Cross-docking denotes the handling of transport units at a dispatching point with little or no storage of goods. The units are instead delivered pre-sorted and labelled by the suppliers and then recombined and directly forwarded after arrival. Consequently, all processes of transfer and removal of storage or stock rotation can be omitted. This leads to a reduction of operating costs in comparison to a traditional warehouse. The operating principle is illustrated in figure 1.

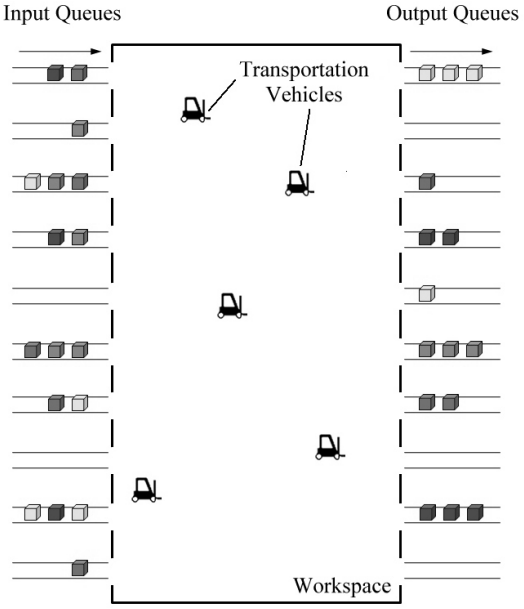

Figure 1. Cross-docking application scenario

Cross-docking is a distribution strategy which combines the advantages of supply via intermediate storage facilities on the one hand and direct supply on the other hand. Operators of cross-docking facilities are shipping agencies and large retail store chains [11]. Research has focused on several aspects such as the optimal position of a cross-docking facility in a supply network, the geometry of the workspace and scheduling and door assignment of inbound and outbound trucks [10].

During operation, transport units are being received at the input queues (pickup points) and have to be assigned to corresponding output queues (delivery points). The input and output queues correspond to loading ramps or doors where trucks dock from the outside to be

loaded and unloaded. Transportation within the distribution centre is typically performed by forklift trucks due to their capability of handling a large variety of goods.

In practical applications, the inbound and outbound doors are each assigned to a fixed destination to be served by the corresponding trucks. This assignment is usually persistent over a longer period of time such as several months. An adequate door assignment can help reducing the average distances for deliveries within the workspace. However, optimizing the assignments in short intervals of time is unfeasible for two major reasons: First, the employees can work more efficiently when being used to the assignments. Second, continuous re-planning would require precise and complete data about amount, type and destination of the transport units to be transported in the future. Such data is generally not available in practice.

Arrival times of trucks are only approximately known in advance in terms of time windows. Frequently, trucks will not be able to adhere to their itinerary, which leads to short-term changes of planned sequences and door allocations. These effects depend on external influences and are random processes from the cross-docking system's point of view. For this reason, the arrival of internal transportation requests will be considered an external factor that cannot be controlled. The task of the cross-docking system as a whole is then to react to these short-term changes and to find a favourable internal workflow in spite of unknown environmental conditions.

The application scenario is distinguished by quickly changing requirements: the arrival rate of new transportation requests is very unsteady during a given observation period. There are typical peak times during a day, a week or a year, but these are only significant in terms of statistic mean values and are not well suited for forecasting single events on the operational level.

For modelling the scenario, the transportation is assumed to take place in a rectangular workspace with N_p pickup points on one side and N_d delivery points on the opposite side, both equally spaced. Each pickup point corresponds to an input queue which may be occupied by a number of transport units. Within the workspace, N_v vehicles can freely move, i.e. there are no fixed lanes to drive on. Transport units are labelled by their destination, which is always exactly one delivery point, i.e. there is no choice from a set of possible delivery points. Furthermore, each vehicle can transport one transport unit at a time, so there are continuously alternating loaded and unloaded trips.

The presented application is related to dynamic vehicle routing or pickup and delivery problems on the one hand and to scheduling problems on the other hand, but shows some specific differences with respect to all of these: Unlike the cross-docking scenario, vehicle routing problems assume that service trips start and end at a central common depot [5]. One-to-one pickup and delivery problems assume transportation assignments from a single pickup point to a single delivery point, but are typically located in urban environments with large distances. In this case, re-optimisation of an existing plan using exact or heuristic methods is feasible for the given time constraints. In contrast, the cross-docking scenario has much shorter transportation times and a much higher arrival rate of new transportation requests. Furthermore, it is much more challenging to maintain a consistent representation of the global system state in real-time than for slowly changing environments.

Scheduling and resource allocation problems generally consider temporal constraints between activities, whereas in vehicle routing and cross-docking problems the visits at pickup points are independent of each other [2]. Moreover, vehicle routing problems aim at minimizing the total traveled distance. In scheduling applications the processing times are known, and the objective is to optimize an average waiting time or the overall throughput. These criteria are also relevant for cross-docking.

To sum this up, vehicle routing problems focus primarily on spatial aspects, whereas scheduling problems focus on temporal relationships. Cross-docking is closer related to vehicle routing than to scheduling with regard to additional constraints. On the other hand, cross-docking is closer related to scheduling with regard to the relevant optimisation criteria.

3.2 Heuristic Dispatching Strategies

The problem of assigning transportation vehicles to requests in manufacturing environments and internal logistics is often solved by using simple heuristic rules [14]. These rules are usually designed to fulfill a single optimisation criterion and are optimal only for special cases. This statement will be illustrated for the cross-docking vehicle dispatching task by means of 3 heuristic dispatching strategies. These same strategies will also be used for evaluation and benchmarking later in this contribution.

Shortest Distance After completion of a transportation assignment at a delivery point, an unloaded trip will be carried out to the nearest pickup point with a vehicle demand greater than zero. This heuristic minimizes the total length of unloaded trips, but requires waiting transport units being kept at each pickup point. This leads to an increase of waiting times before pickup. This strategy maximizes the throughput on the condition that all input queues contain at least one transport unit, which cannot be guaranteed in the general case.

Highest Demand After completion of an assignment at a delivery point, an unloaded trip will be carried out to the pickup point with the highest vehicle demand, even if this pickup point is at a farther distance than other pickup points. This heuristic minimizes the length of the input queues, but leads to longer distances for the unloaded trips and to a decrease of throughput due to lower capacity utilisation.

Remaining processing time New transportation requests are immediately assigned to the vehicle with the minimum sum of remaining processing times, i.e. to the vehicle which can serve the request first. A disadvantage of this strategy is that an established plan might not be optimal any more at the time of execution, because new transportation requests can have arrived in the meantime. To overcome this problem, continuous re-planning would have to be introduced, which was already pointed out to be unfeasible in this scenario. The strategy also requires an estimation of all the remaining processing times, i.e. precise information about the position of all vehicles in the workspace. This information is usually not available in real systems today.

A consequence of these considerations is that the optimisation criteria *capacity utilisation* and *length of input queues* are not compatible with each other. The *average length of input queues* and the *average waiting time* are equivalent due to Little's Law [12].

An approach to optimize the behaviour beyond these heuristic appoaches results from the unsteady arrival of new transportation requests, which can only be described by an average rate over a given interval of time. In fact, even this rate is not a constant, but typically depends on the time of day, the day of the week and the calendar month. In practice, the necessary number of vehicles is calculated using statistic mean values of input rate, transportation cycle times and

availability of equipment. For a given period of time such as a day or a shift, there is a certain amount of staff available according to prior manpower planning. A short-term adjustment of the available technical and human resources is hardly possible. As a consequence, there will always be phases of lower workload (or overcapacities respectively), during which the system could for instance process longer input queues at the cost of a sub-optimal capacity utilisation. This gives a frame for optimisation of vehicle assignment on the operational level. The question is then how to develop a vehicle dispatching strategy, which can efficiently handle progressions of incoming requests that are not precisely known in advance.

4 LEARNING A STRATEGY

4.1 Episodic Structure of the Task

The task is modelled as a decision process with states and actions according to section 2. From a transportation vehicle's point of view, the operation process consists of cyclic episodes of alternating loaded and unloaded trips. Each episode starts with picking up the first waiting transport unit from an input queue. At this point, there are no alternatives to chose from, since the destination point is defined by the transport unit. The unit is then transferred to a destination point. After delivery, a new destination (i.e. pickup point) has to be chosen for the following unloaded trip. After arriving at the chosen pickup point and pickup of a transport unit, the next episode begins. Hence, actions are chosen depending on the current system state each time a loaded trip was completed . The usefulness of this decision turns out when arriving at the chosen pickup point.

We define negative rewards for each time step of an unloaded trip and positive rewards for picking up transport units at a pickup point. These positive rewards are weighted by the length of the corresponding queue, in order to make a reduction of long queues more attractive. The value function is updated at the end of each episode with the accumulated reward. A special property of the application considered here is that it is an asynchronous decision process: Transportation vehicles are normally busy for longer periods of time during which they do not have to make any decisions. Decision events take place at randomly distributed intervals over time.

In phases of low workloads, vehicles might idle, because all waiting assignments are being served already. In this case it seems to be more appropriate to chose an arbitrary pickup point as a waiting position than to stay at the delivery point where the last transport was completed, in order to be able to serve a new transportation assignment more quickly. In a real system, the number of vehicles N_v is typically smaller than the number of pickup points N_p, so even if all vehicles were waiting at pickup points, the next assignment might still appear at a position with no waiting vehicle. For this reason, changing between waiting positions must also be coordinated between the vehicles. This behaviour can be understood as another dispatching strategy, for which a value function is required. As it turns out, the same learned value function can be successfully used for both dispatching strategies.

4.2 Learning Algorithm

The aim of the method which is presented here is to automatically find a set of decision rules for vehicle dispatching by using a simulated environment. In this environment, properties such as the geometry of the workspace, the number of vehicles, and the number of pickup and delivery points can be chosen by the user. The result of the simulation is a behaviour which can be transferred to a real-world system.

The idea of the learning algorithm is to estimate the value $Q(s, a)$ of all state-action pairs by averaging the obtained rewards. This is known as Monte Carlo Sampling [21]. Monte Carlo methods have the advantage that a learning system can learn a behaviour based on experience by interaction with the environment; no model of the environment with explicitly known probability distributions for state transitions is required.

For the following investigations, we are interested in the system behaviour under the condition of unknown arrival times of transportation requests. For this reason, we assume no *a priori* knowledge, but rather model the arrival of new requests as a Poisson process with the input rate λ. This corresponds to the average number of new transportation requests per time step. This random process is used both for training and for evaluation.

In order to guarantee convergence of the algorithm, each state-action pair must be visited sufficiently often. For this reason, occasionally actions have to be chosen that are not considered optimal. This is known as the exploration-exploitation dilemma. For our investigations, actions are chosen by ε-greedy action selection, i.e.

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & with\ probability\ 1-\varepsilon \\ random\ a \epsilon A & with\ probability\ \varepsilon \end{cases}$$

We chose $\varepsilon = 0.1$ for learning. For evaluation of the learned behaviour, ε was set to zero to prevent further exploration.

4.3 State and Action Representation

The value function is to be represented as a table, in which the mean values of the observed rewards for all visited state-action pairs are stored along with the number of visits. This table has a maximum of $L(Q) = |S| \cdot |A|$ entries. In the given application scenario, there are as many possible actions as there are pickup points, i.e. $|A| = N_p$. The actions can be easily represented by an index number with respect to a sorted list of pickup points.

In contrast, the state description must contain information about both the occupancy of input queues and about the state of all vehicles. Therefore, a straightforward approach for a state representation is the tuple

$$s = (v_1,\ v_2,\ ..., v_{N_v},\ q_1,\ q_2,\ ..., q_{N_p})$$

with v_i the state of the vehicle i and q_j the occupancy of the input queue j. The occupancy q_j is an integer number from the set $S_p = \{0, ..., \hat{q}\}$ where $\hat{q}$ denotes the maximum buffer capacity. The vehicle state is given by the current position, orientation and loading state. The position and orientation are continuous variables, leading to an infinite number of possible states. This is a considerable difficulty for handling the decision process. To overcome this problem, the set of vehicle states can be transformed by discretisation to a finite set S_v. Then the total state set has the finite cardinality

$$|S| = |S_v|^{N_v} \cdot |S_p|^{N_p}$$

This grows exponentially with both the number of vehicles and the number of pickup points and is therefore only feasible for very small problem instances.

For this reason, the objective of this work is to develop a scalable state representation by reducing the dimensionality. To achieve this, we take advantage of two symmetric properties of the problem. The first of these properties is that all vehicles have the same features and can be interchanged. The second property is that from a vehicle's

point of view, the characteristics of a pickup point are its vehicle demand and its distance from the vehicle's current position. Therefore, the pickup points are also interchangeable with respect to the value function, i.e. two pickup points with the same demand and distance should have the same estimated value.

According to this, any vehicle i that has to choose an action can as well use the following parametrisation of the value function:

$$Q(s,a) = Q(demand(a),\ distance(v_i,a),\ \mathbf{v}_{rem}(s),\ \mathbf{q}_{rem}(s))$$

where the vehicle states of all remaining vehicles $i' \neq i$ are concentrated in the vector $\mathbf{v}_{rem} \in S_v^{N_v-1}$. The properties of all remaining pickup points $j \neq a$ are concentrated in a vector $\mathbf{q}_{rem} \in S_p^{N_p-1}$. We call the vectors $\mathbf{v}_{rem}$ and $\mathbf{q}_{rem}$ the "context" of a decision.

So far, this is only an alternative parametrisation and does not yet lead to a reduction of the number of dimensions. The idea is to find a function

$$f_c\ :\ S_v^{N_v-1} \times S_p^{N_p-1} \rightarrow \mathbb{R}$$

where $f_c(\mathbf{v}_{rem}(s),\ \mathbf{q}_{rem}(s)) = g_c(s)$ is an indicator or a characteristic number, representing an abstraction of the actual context. This function is a projection of the context part of the state vector s to a subspace and provides the desired reduction of dimensionality. Therefore, the value function can be expressed as

$$Q(s,a) = Q(demand(a),\ distance(v_i,a),\ g_c(s))$$

From the theoretical considerations shown in section 3.2 and from experiences with the described heuristics it follows that the vehicles' behaviour should depend on the current degree of workload. A possible measure for this is the **total occupancy** of the input queues. This represents the amount of work to do within a certain time horizon. Another measure is the loading state of the vehicles. This can be represented by the number of currently **busy vehicles**, which will not be available for another transportation assignment until some future point in time. Both measures can also be combined to the **total demand**, which is the difference of the total occupancy and the number of vehicles currently executing an unloaded trip to a pickup point.

In all three cases, the context function g_c maps the entire context to a single integer number. If for instance the number of busy vehicles is used as the context, g_c will yield values from the set $\{0,\ ...,\ N_v - 1\}$ because the deciding vehicle is never busy. From the parametrisation given above follows that the state set with respect to g_c has the cardinality

$$|S_{g_c}| = (\hat{q} + N_v + 1) \cdot N_p \cdot N_v$$

which has a linear growth with the number of pickup points and a quadratic growth with the number of vehicles. We assume symmetry of pickup and destination points, so there are N_p possible distances from a destination point to a pickup point. For non-symmetric cases the growth is still linear, however with a different second factor.

The major questions arising are if this is a valid simplification and which approach to chose for representing the degree of workload. These questions were investigated using a simulation model. The results are shown in the next section.

5 SIMULATION RESULTS

Policies were learned according to the method described in section 4 during a learning phase of 10^7 simulation steps at an input rate of $\lambda = 0.06$. This corresponds to an expected value of 600000 transportation operations being carried out during this period of time. As an evaluation criterion we use the average waiting time of new assignments from their appearance in an input queue until pickup by a transportation vehicle. Minimizing the average waiting time also minimizes the passage time through the dispatching point. This is favourable for serving the adjacent external transportation processes and also for minimizing the duration of stay of inbound and outbound trucks.

The throughput is less suitable for benchmarking, because for the steady state, the throughput will be equal to the input rate λ if the cross-docking system is properly dimensioned. An input rate higher than the transportation capacity will lead to a capacity overload. In this case, all input queues will be filled to a maximum, and it is already known that the *shortest distance* heuristic is best suitable for this case.

Table 1 shows a comparison of the three approaches to represent the context for an example instance of the problem with $N_p = N_d = 10$ and $N_v = 5$. The number of busy vehicles clearly yields the best results for both the average waiting time at the end of the learning phase and for the steady state. Surprisingly, this is also the approach with the shortest decision table.

Table 1. Results for different context functions

Context	$\bar{t}_{w,0}$	$L(Q)$	$\bar{t}_w$
total occupancy	54.5	3083	49.5
busy vehicles	**49.8**	700	**43.9**
total demand	57.2	4257	47.2

$\bar{t}_{w,0}$: average waiting time after learning phase
$L(Q)$: length of value table
$\bar{t}_w$: average waiting time (steady state)

The number of resulting table entries after the learning phase is shown in table 2 for different problem instances. Investigations were successfully carried out up to a size of $N_p = N_d = 30$ (denominated "30 x 30" in the table) with 10 vehicles. The growth of $L(Q)$ is very moderate, so even for large problem instances, a tabular representation of the value function is well tractable.

Table 2. Length of value table $L(Q)$ for different problem instances

	Number of vehicles N_v			
N_p x N_d	4	6	8	10
10 x 10	588	857	1308	1789
15 x 15	764	1116	1589	1974
20 x 20	901	1387	1973	2573
25 x 25	970	1582	2222	2919
30 x 30	986	1592	2270	2945

As our final result, figure 2 shows a comparison of the average waiting times for the 3 heuristic strategies and the strategy obtained by reinforcement learning. Each point represents the resulting average waiting time after 200000 simulations steps, averaged over 20 simulation runs. The diagram shows that the *shortest distance* heuristic achieves a better performance than the *remaining processing time* heuristic for medium and high input rates, whereas for low input rates it is vice versa. The learned strategy is superior to the heuristics for all input rates except for a very small range near the maximum workload. The *highest demand* heuristic is only favourable for very low input rates or for problem instances with a very small number of

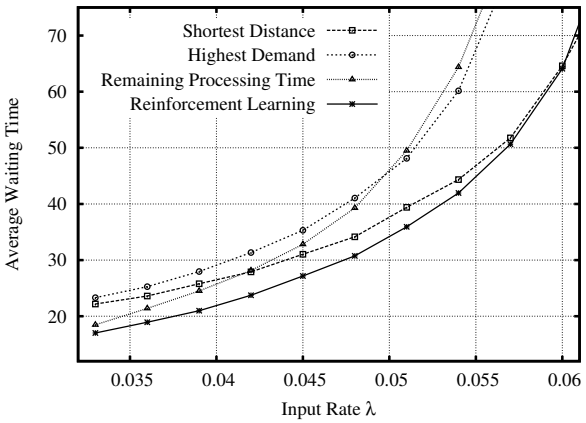

Figure 2. Performance of dispatching strategies for different input rates

pickup points (not shown in the figure). On average over the examined range of input rates, the learned strategy leads to a reduction of the average waiting time of 8.3 % with respect to the corresponding best heuristic strategies.

6 CONCLUSION

We have shown how reinforcement learning can be applied to find efficient dispatching strategies for a number of transportation vehicles in a cross-docking application scenario. The method is able to derive a behaviour from simulated experience and does not require an explicit representation of state transition probabilities of the environment. For a system configuration, whose properties can be specified by the user, the simulation model will generate a corresponding decision table.

We addressed the exponential growth of the state set by introducing a projection of the context description to a subspace. This method reduces the dimensionality of the state set. Using this context-based state representation, which estimates the general value of state-action pairs, our approach scales very well to problem instances of a realistic size. The performance of the adaptive strategy obtained by reinforcement learning was shown to yield a better performance than commonly used heuristic dispatching strategies on random conditions over a wide range of input rates.

This is certainly a rather specific approach which is tailored to the considered application domain. In particular, we took advantage of both the symmetry of the workspace and of the interchangeability of the individual transportation systems. However, the idea of using context descriptions might be extended to more general problems. An objective of our future research is to include the process of finding an appropriate state representation in the learning algorithm in order to automatically generate an appropriate context description for a given problem domain.

ACKNOWLEDGEMENTS

This contribution was prepared within the research project “Cognitive Logistic Networks (CogniLog)” and supported by the German federal state of Lower Saxony with funds of the European Regional Development Fund (ERDF).

References

[1] Dieter Arnold and Kai Furmans, *Materialfluss in Logistiksystemen*, Springer Berlin Heidelberg, 5th edn., 2007.

[2] J. Christopher Beck, Patrick Prosser, and Evgeny Selensky, ‘Vehicle routing and job shop scheduling: What’s the difference?’, in *Proc. 13th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 267–276, Trento (Italy), (2003).

[3] Wolf-Rüdiger Bretzke, ‘Supply chain management: notes on the capability and the limitations of a modern logistic paradigm’, *Logistics Research*, **1**(2), 71–82, (2009).

[4] Balázs Csanád Csáji, László Monostori, and Botond Kádár, ‘Reinforcement learning in a distributed market-based production control system’, *Advanced Engineering Informatics*, **20**, 279–288, (2006).

[5] Michel Gendreau, François Guertin, Jean-Yves Potvin, and René Séguin, ‘Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries’, *Transportation Research Part C*, **14**(3), 157–174, (2006).

[6] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin, ‘Vehicle dispatching with time-dependent travel times’, *European Journal of Operational Research*, **144**, 379–396, (2003).

[7] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, ‘Reinforcement learning: A survey’, *Journal of Artificial Intelligence Research*, **4**, 237–285, (1996).

[8] Martin Lauer and Martin Riedmiller, ‘An algorithm for distributed reinforcement learning in cooperative multi-agent aystems’, in *Proc. 17th Int. Conference on Machine Learning*, pp. 535–542, Stanford (CA), (2000).

[9] Martin Lauer and Martin Riedmiller, ‘Reinforcement learning for stochastic cooperative multi-agent-systems’, in *Proc. 3rd Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pp. 1516–1517, (2004).

[10] Shayla Ley and Sherif Elfayoumi, ‘Cross dock scheduling using genetic algorithms’, in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, pp. 416–420, Jacksonville (FL), (2007).

[11] Y. Li, A. Lim, and B. Rodrigues, ‘Crossdocking - JIT scheduling with time windows’, *Journal of the Operational Research Society*, **55**(12), 1342–1351, (2004).

[12] John D. C. Little, ‘A proof for the queuing formula: L= λ w’, *Operations Research*, **9**(3), 383–387, (1961).

[13] Stephan H. Mayer, *Development of a completely decentralized control system for modular continuous conveyors*, PhD thesis, University of Karlsruhe (TH), Faculty of Mechanical Engineering, 2009.

[14] David Naso and Biagio Turchiano, ‘Multicriteria meta-heuristics for agv dispatching control based on computational intelligence’, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, **35**(2), 208–226, (2005).

[15] Ludger Overmeyer, Gerd Heiserich, Sascha Falkenberg, and Andreas Jungk, ‘Automatische Konfiguration und Optimierung von Materialflusssystemen durch kognitive Logistikmodule’, in *VDI-Berichte Band 2066, 18. Deutscher Materialfluss-Kongress*, pp. 197–208, Munich (Germany), (2009).

[16] Ludger Overmeyer, Kai Ventz, and Sascha Falkenberg, ‘Interfaced multidirectional small-scaled modules for intralogistic operation’, *WGTL Logistics Journal*, (2009).

[17] H. Van Dyke Parunak, ‘Applications of distributed artificial intelligence in industry’, in *Foundations of Distributed Artificial Intelligence*, eds., G.M.P. O’Hare and N.R. Jennings, chapter 4, 139–164, Wiley-Interscience, (1996).

[18] Carlos D. Paternina-Arboleda and Tapas K. Das, ‘A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem’, *Simulation Modelling Practice and Theory*, **13**, 389–406, (2005).

[19] Henning Rekersbrink, Thomas Makuschewitz, and Bernd Scholz-Reiter, ‘A distributed routing concept for vehicle routing problems’, *Logistics Research*, **1**(1), 45–52, (2009).

[20] Jörn Schönberger and Herbert Kopfer, ‘On decision model adaptation in online optimization of a transport system’, in *Management logistischer Netzwerke - Entscheidungsunterstützung, Informationssysteme und OR-Tools*, eds., Hans-Otto Günther, Dirk C. Mattfeld, and Leena Suhl, 361–381, Physica-Verlag, (2007).

[21] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge (MA), 1998.