Rethinking transport
27–30 April 2020

TRA 2020 HELSINKI

Proceedings of 8th Transport Research Arena TRA 2020, April 27-30, 2020, Helsinki, Finland

Reinforced logistics: Automated tour planning with reinforcement learning

Marco Kemmerling[a]*, Marc Haßler[a], Sophie Hallstedt[a], Alexia Fenollar Solvay[a]

[a]*Institute of Information Management in Mechanical Engineering, RWTH Aachen University, Dennewartstrasse 27, Aachen 52068, Germany*

Abstract

In times of ever-growing and flexible demand for freight transports, the ability to quickly create efficient tour plans becomes increasingly important. Currently, this process is often carried out manually and requires trained personnel with years of experience. Due to the increasing number of goods to be shipped, as well as short reaction and planning times, this task becomes more and more complex and time-consuming.

In this paper, we present a novel approach for automated tour planning based on reinforcement learning methods. This enables us to learn on the basis of historical tour data, react flexibly to changes in transport conditions, and create tours similar to the ones created by human planners. Our focus is on short-range business-to-business transports, where the delivery areas are not as densely populated with customers as within the consumer transport sector.

Keywords: tour planning; reinforcement learning; machine learning; vehicle routing; logistics; freight transport;

* Corresponding author. Tel.: +49 241 80 91183;
E-mail address: marco.kemmerling@ima-ifu.rwth-aachen.de

1. Introduction

Road freight transport accounted for over three quarters of all freight traffic in the European Union in 2016 (Eurostat, 2018), thus being an important contributor to as well as an enabler of the modern economy. The ideas behind the terms Industry 4.0 and Internet of Things push the logistics and transport sector to be more flexible, connected and automated. Additionally, increasingly fast-paced times where goods need to be delivered on short notice leave very little time to plan their delivery.

The capability to plan deliveries quickly by creating a set of time and cost efficient tours becomes ever more essential. For transport companies, this has a strong savings potential since customer wishes can be accommodated more quickly and subsets of orders can be delegated to subcontracting transport services earlier. Not only does this capability benefit individual companies, but also society as a whole, since the efficiency of tours corresponds with fuel expenditure to a large degree and thus also with emitted pollution.

Many different problems of this nature have been formalized and can be solved using mathematical optimization methods (e.g. (Laporte, 1992), (Zirour, Liong, Ismail, & Omar, 2008)). However, since formalized problem definitions often do not capture all the specifics of reality, the solutions generated by optimization methods often need to be modified and fine-tuned by human planners. Due to this and the required calculation time of traditional methods, the process of creating tours is often still carried out entirely manually by human experts.

To reduce the time needed to create tours, we do not attempt to eliminate humans from the process entirely, but focus on the creation of *initial* tours so that these can be further adapted by experts. Thus, the goal is to provide transportation tours, which are easy to work with for the human planners, i.e. are similar to expert tours and require only few modifications to be applied in practice. Since it is not clear how the goal of creating tours, which mimic expert tours, can be stated as a precise mathematical objective, traditional optimization methods are not well suited to this particular problem. Instead, we turn our attention to the field of machine learning.

While there have been some initial attempts at utilizing machine learning for combinatorial optimization (e.g. (Bengio, Lodi, & Prouvost, 2018), (Nazari M. , Oroojlooy, Snyder, & Takác, 2018)), they remain well within the realm of foundational research, without direct applicability to real-world problems. Although tailor-made machine learning methods could one day yield promising results for combinatorial optimization problems, we investigate if already existing techniques can be transferred and utilized to solve a problem of this kind. Specifically, we apply methods from reinforcement learning, a sub-field of machine learning where autonomous agents learn behaviors by receiving rewards and/or punishments while interacting with an environment.

Our research in this paper is guided by the following set of questions: Can a reinforcement learning agent learn to create transport tours similar to those created by human experts? How should the reward function be designed to enable the agent to achieve this overall goal in a time, resource and distance efficient way? Is it possible to do so receiving rewards for individual steps in the process of creating a solution, rather than one reward that judges the quality of the overall solution? How can the quality of the overall solution be measured?

We answer these questions in the remainder of this paper, which is structured as follows. In section 2, we describe the tour planning problem examined here in detail. Section 3 describes the state of the art concerning problems of a similar kind as well as a general introduction to reinforcement learning. In section 4, the solution approach is presented, including the system modeling and a tailored reinforcement learning implementation. Subsequently, section 5 shows the results achieved using the described methodology, both in terms of quality and speed. This is followed by a discussion as well as some considerations regarding improvements of the employed methods in section 6. At the end, a brief conclusion is given.

2. Problem Statement

The task examined here is the creation of initial tours in a certain geographical area for a given day. This area is generally small enough so that each order can be delivered on a single day.

Specifically, given a set of orders, which need to be delivered to various locations, the goal is to create a set of tours such that every order is assigned to exactly one tour. Each order has the following properties: a weight measured in kilograms, a space requirement measured in the amount of pallets required, as well as a delivery location and a pickup location. The pickup location is usually a central warehouse from which the majority of the orders are dispatched, although this can differ in some cases.

A tour is subject to a set of constraints consisting of (1) a weight limit w_{max} given in metric tons (2) a space limit s_{max} measured in the maximum number of pallets, and (3) a maximum duration t_{max} given in hours. While variations of the exact constraints are conceivable, for the proof-of-concept described here we assume the following values based on a typical vehicle used in practice:

- $\mathrm{w}_{\mathrm{max}} = 7.5$
- $s_{max} = 11$
- $t_{max} = 9$

We further assume that the loading of goods into the vehicle is performed before the beginning of the tour and consequently does not contribute to the tour duration, while the unloading of goods does contribute to the duration of the tour and takes a fixed amount of 15 minutes.

Apart from the general idea that orders need to be assigned to tours, an explicit objective function, which needs to be minimized or maximized, is not given. Instead, the goal pursued in this article is to create tours that mimic those created by a human tour planner. The challenge is therefore to create a reward function that enables the agent to follow the implicit goal of minimizing the number of tours and the traveled distance.

3. State of the Art

3.1. The Vehicle Routing Problem

The vehicle routing problem (VRP) is concerned with the delivery of goods from a depot to a set of customer locations such that the overall distance driven is minimized (Clarke & Wright, 1964). In its most basic form, the VRP is already NP-hard, i.e. we cannot hope to find optimal solutions to large problem instances efficiently (Lenstra & Kan, 1981). Many variations on the basic VRP exist, varying in their objective functions (Lee & Ueng, 1999) (Giannikos, 1998), their constraints (Kumar & Panneerselvam, 2012) and the considered time horizons (Francis, Smilowitz, & Tzur, 2008).

Due to the above-mentioned NP-hardness of even the basic VRP formulation, exact solution methods tend to be prohibitively time-consuming in practice. Instead, heuristics and meta-heuristics can be used to compute solutions which may not be optimal, but are usually good enough for practical concerns (Laporte, 1992).

How viable these methods are depends on the specific needs of each individual business. Although there are use-cases where a solution only needs to be computed once and is from then on set in stone, in many cases changing circumstances require the solution to be modified or created from scratch many times in a short period of time. Even then, resulting solutions are frequently not perfectly suited for deployment in practice because of edge cases the solution method is not aware of. Handling these edge cases often requires knowledge, which cannot easily be formalized, but is important for the successful execution of the delivery process. In these cases, human experts, equipped with the aforementioned implicit knowledge, are required to ensure that all edge cases are covered. As a result, in many cases, the tour planning process is carried out by human experts either entirely, or with the help of common vehicle routing software.

3.2. Reinforcement Learning

Reinforcement Learning is an area of machine learning in which an agent can interact with an environment and observe its state with the goal to learn how to accomplish a certain task (Sutton & Barto, 2018). At all times, the agent is in some state $s \in S$, which it perceives as an observation o, and can perform some action $a \in A$, where both the state space S and the action space A may be either discrete or continuous. Upon performing an action, the agent transitions to a new state s' and receives some reward r. In which state s' the agent arrives in, when performing action a in state s, may or may not be defined deterministically. The aim of the agent is to learn some policy π, i.e. a mapping from states to actions, in order to maximize not its immediate, but rather its long-term cumulative reward.

Once the environment has been sufficiently explored and a good policy has been found, it is prudent to stick to the learned policy rather strictly to maximize rewards and avoid an unexpected behavior. In the early stages of training, however, a good policy has not been found yet and thus rather than strictly following the current policy, some mechanism to encourage exploration is needed. How much exploration and how much exploitation of the current policy should be performed at which stages of training is often called the *exploration-exploitation trade-off.* A popular approach to the exploration-exploitation problem is the ϵ-greedy method (Watkins, 1989), where, at each step, a random action is chosen with a probability of ϵ while the best action according to the current policy is chosen with probability $1 - \epsilon$.

One of the more popular variants of reinforcement learning is Q-learning (Watkins, 1989), where an action-value function $Q(s, a)$ is estimated, i.e. a function describing the long-term expected reward when performing action a in state s. In the basic Q-learning algorithm, this action-value function is usually represented by a simple table. At each time-step, the action a with the highest Q-value in the current state s is chosen, a reward r is collected, and the Q-table is updated as follows:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where α is the step-size, which describes how strongly the value estimates are affected by new information, and γ is a discount factor, describing to which degree estimated rewards from future actions are taken into account.

In recent times, an approach called Deep Q-learning has gained popularity. Here, the Q-values are computed by a (deep) neural network. Additionally, Deep Q-learning approaches are usually characterized by the usage of experience replay buffers (Lin, 1993) as well as target networks (Mnih, et al., 2015), which are both employed to stabilize the learning process. Experience replay describes the act of keeping a buffer of a certain size filled with past experiences. During training, this buffer is periodically sampled from such that the agent is not only exposed to its current situation in the environment, but also to past experiences, thus decorrelating successive inputs to the agent. This prevents the agent from overadapting to its current situation while losing the power to generalize to different situations. The second addition, a target network is a second network which is used to compute the Q-value of the best future action in equation 1. This second network is a copy of the first network but its parameters are updated less frequently, which means that the agent has a stable target to converge to.

4. Methods

We approach the problem described in the section 2 by training a reinforcement learning agent to iteratively assign orders to tours. Note that the agent only groups orders to a tour, but does not determine in which sequence the orders in each tour are delivered. The latter is instead computed heuristically by first applying a greedy algorithm and further optimizing the resulting solution using the 2-OPT heuristic (Croes, 1958) to minimize the tour's overall distance. This computed result is used within the reward function (c.f. section 4.2) of the agent.

4.1. Environment

Given the general introduction to reinforcement learning in the preceding section, we now describe how we model the problem at hand in order to solve it in a reinforcement learning context.

As our approach mimics a human approach, it requires the agent to build tours in a step-wise manner. At each step, the agent is presented with observations from a single order and a single tour (illustrated in Fig. 1). Initially, the tour is empty and the order is randomly selected from the list of all unassigned orders O_u.
It can then choose between four possible actions: (1) skip the current order, (2) assign the current order to the current tour, (3) finish the current tour, and (4) terminate the episode. In case of actions (1) and (2), the agent is presented with a new unassigned order. When performing action (3), a new tour is created and the agent is presented with one of the remaining orders. Finally, action (4) terminates the entire tour planning process.

The agent aims to maximize its long-term reward by choosing between the different actions based on a set of observations pertaining to its state and especially to the current order and tour. Observations describing the current tour include how much space is left in the vehicle, how much additional weight the vehicle can carry, and how much longer the duration of the tour can be without violating constraints.

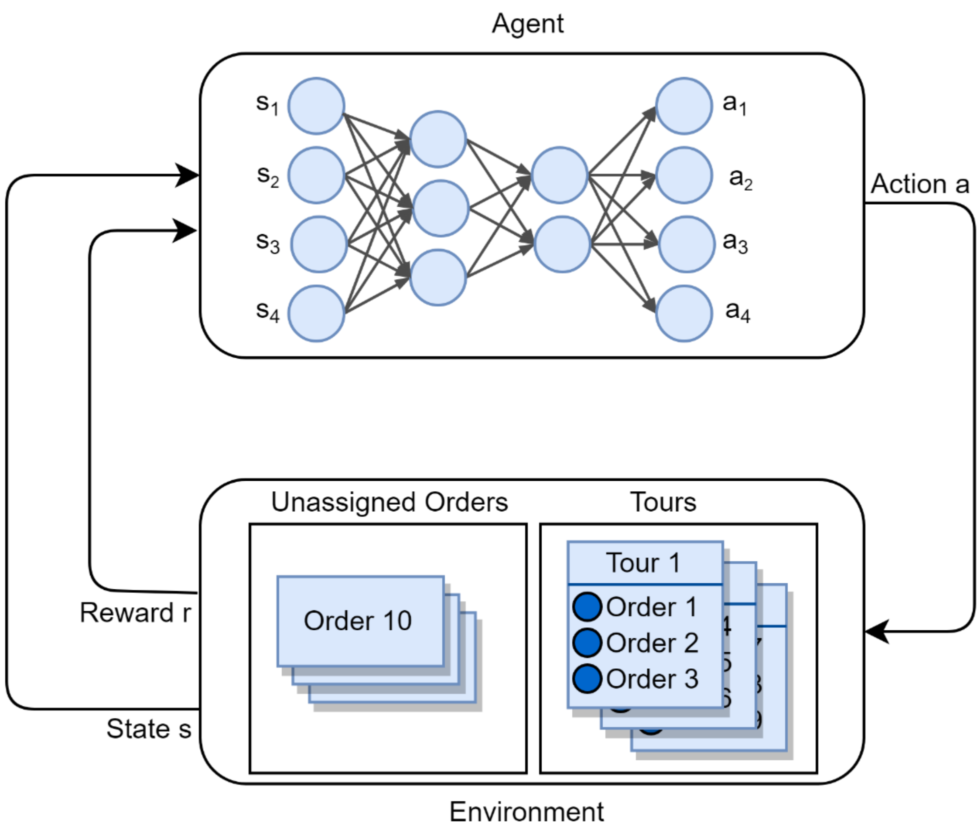

Fig. 1: Diagram of the agent and the environment it interacts in. Note that this only serves illustration purposes, exact numbers of e.g. state elements or neural network nodes differ in the actual implementation

An observation concerning both the current tour and the current order is the size of the angle α, which is the angle in which the combination of the current tour and the current order operate in geographically. More specifically, given a set of lines L, where each $l \in L$ is defined by two points of which one is the depot and one the location of an order already in the current tour, α is defined as the maximum angle between any $l \in L$ and the line defined by the locations of the depot and the current order.

Other observations describing the current order include the required weight and space, as well as the change in tour duration which would result when adding this order to the current tour. In addition, the agent receives the weight of the lightest and heaviest orders, as well as the space requirements of the largest and smallest unassigned orders. Finally, the agent can observe the remaining number of unassigned orders as well as the current number of inactive steps, i.e. the number of consecutive skipping actions.

4.2. Reward function

Although a precise objective function is not known in our scenario, the quality of a given planning can be judged in the same way that human planners assess the quality of their work - not by evaluating a mathematical function in their mind but by checking for the presence of certain desirable properties. Among them: (1) The number of tours in the solution should be as low as possible, (2) The overall distance driven should be as low as possible. (3) The average tour should be as full as possible (in the sense that it should not be possible to add many more orders without violating constraints) (4) Orders with delivery locations in geographical vicinity should be in the same tour.

The properties given above are not necessarily comprehensive and not necessarily hard rules, but rather guidelines, raised when imitating the reasoning of expert tour planners. Properties (1) and (3), as well as (2) and (4) may share similar goals. On the one hand, (1) and (2) take a global view on the entire solution and on the other hand

(3) and (4) take a local view on individual tours in the solution. As our proposed system build tours one by one, the reward function defined below focuses on the individual properties (3) and (4) rather than (1) and (2).

Depending on which action is performed in which state, a reward or punishment (i.e. negative reward) is given according to a predefined reward function. The reward function posed in the following has several components. The first component, r_o, is the reward the agent receives when assigning an order to a tour. It is a step-wise function, which is roughly inversely proportional to the change in distance Δd of the given tour, precisely specified by equation 2:

$$r_o = \begin{cases} 10, & \Delta d = 0 \\ 3, & \Delta d \leq 10 \\ 0.5, & \Delta d \leq 50 \\ 0.1, & \Delta d \leq 100 \\ 0.01, & otherwise \end{cases} \tag{2}$$

This serves to satisfy property (4) described above.

To address property (3), whenever a tour is closed, the agent is punished proportionally to the amount of space s or weight w left in the vehicle, whichever is lower (see equation 3).

$$r_t = -\min\left(1 - \frac{s}{s_{max}}, 1 - \frac{w}{w_{max}}\right) \tag{3}$$

To encourage the agent to assign each order to a tour, the agent receives a negative reward equal to the number of unassigned orders $\mathcal{O}_u$ when an episode is terminated (see equation 4).

$$r_e = -|\mathcal{O}_u| \tag{4}$$

In order to limit the amount of skip actions the agent chooses, a small penalty of -0.01 is given whenever the agent skips an order. Once the agent has skipped over the complete list of unassigned orders without adding an order or closing a tour, the next skip action is associated with a penalty of -10 and the current tour is closed. A penalty of -10 is also applied whenever the agent violates one of the hard constraints. Additionally, constraint violations lead to the termination of the current episode.

4.3. Agent

The agent presented here utilizes a neural network to compute Q-values. This network is an ordinary feed-forward network with three hidden layers having 30, 25, and 12 units, respectively. The hidden layers use Rectified Linear Units (ReLUs) (Nair & Hinton, 2010) as activation functions while the final output layer uses a simple linear activation. To stabilize training, the agent makes use of a target network which is updated every 10,000 steps as well as experience replay with a replay buffer of length 100,000. During training, the agent follows an ϵ-greedy policy with $\epsilon = 0.1$, i.e. it chooses the best action (by its own estimation) 90% of the time and a random action the remaining proportion of the time. In the experiments and results presented in the following section, the agent was trained for 500,000 steps.

5. Results

The results in this section are obtained by training the agent described in the previous section on historical tour data spanning a period of one year, i.e. round about 27000 orders, and one hub location. This historical data includes information about actual orders including their characteristics, such as weight and destination, as well as how these were delivered in tours during the daily operations of a mid-sized logistics company. Selected parts of this data are excluded during training to serve as a test set for later evaluation of the agent's performance.

To perform this evaluation, we let the agent create tours on the excluded test set and compare those to the actual expert tours associated with the data. Hence, for a given set of orders $\mathcal{O}$, the agent generates a set of tours $\mathcal{T}$ including a tour $t_{|\mathcal{O}|+1}$ containing all unassigned orders. A mapping between the orders and their corresponding tours can be defined as follows.

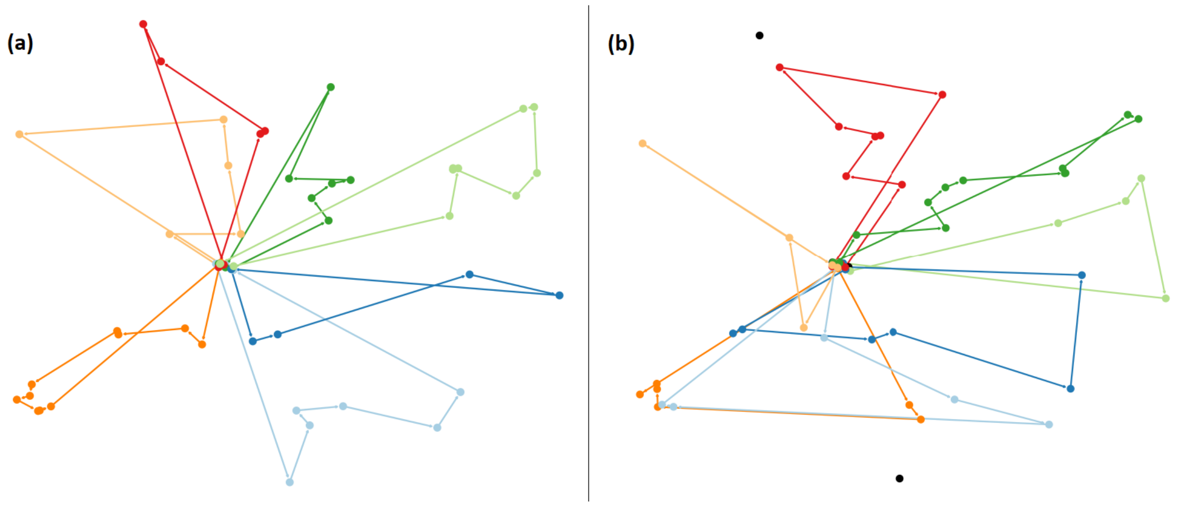

Fig. 2: (a) tours built by expert planners; (b) tours built by the agent

Definition 1 *Let $\mathcal{O}$ be a set of orders and $\mathcal{T}$ the generated tours. Then the function*

$$T: \mathcal{O} \to \mathcal{T}$$
$$T(o) \coloneqq \{\ t \in \mathcal{T} \mid\ o \text{ is transported on } t\} \tag{5}$$

is called Order-To-Tour-function (OTT-function).

We visualize tours with such an OTT-function computed by the agent in Fig. 2 (b), where each vertex represents an orders' delivery location, the edges in between them indicate the order in which these vertices were visited. Each color represents a distinct output of $T(o)$. The agent managed to assign all but two of the 46 given orders without breaking any restrictions. Overall, the orders within a tour form a coherent picture, i.e. their delivery locations are within some limited geographical area. Especially in the bottom half of the plot, some tours heavily overlap with each other, while the expert tours in Fig. 2 (a) maintain a much clearer spatial separation between the different tours. However, two of the expert tours break hard restrictions that our agent does not break, giving the experts an advantage.

To measure this geographical separability of tours mathematically, we define the *Neighborhood Relation Quotient* (NRQ)

Defintion 2 *Let $\mathcal{O}$ be a set of orders, $T(o)$ the OTT-function (c.f. definition 1), and $\mathcal{N}(o)$ a defined neighborhood for each order $o \in \mathcal{O}$.*

Then the Neighborhood Relation Quotient $NRQ_{\mathcal{N}}(\mathcal{O})$ is given by

$$NRQ_{\mathcal{N}}(\mathcal{O}) \coloneqq \frac{1}{\sum_{o \in \mathcal{O}} |\mathcal{N}(o)|} \sum_{o \in O} \sum_{o' \in \mathcal{N}(o)} \delta\big(T(o), T(o')\big) \tag{6}$$

where

$$\delta\big(T(o), T(o)\big) \coloneqq \begin{cases} 1, & if\ T(o) = T(o') \\ 0, & otherwise \end{cases} \tag{7}$$

Intuitively, it measures how often orders with geographically close delivery locations (i.e. in the same neighborhood) are assigned to the same tour. For our evaluation we use two alternative definitions of the neighborhood. One of them, dubbed $\mathcal{N}_r(o)$, describes the neighborhood as the subset of orders within a radius r from o.

$$NRQ_r(\mathcal{O}) \coloneqq \frac{1}{\sum_{o \in \mathcal{O}} |\mathcal{N}_r(o)|} \sum_{o \in O} \sum_{o' \in \mathcal{N}_r(o)} \delta\big(T(o), T(o')\big) \tag{8}$$

The other one, dubbed $\mathcal{N}_k(o)$, defines the neighborhood as the k geographically nearest orders to o using the Euclidean distance.

$$NRQ_k(\mathcal{O}) := \frac{1}{k|\mathcal{O}|}\sum_{o\in O}\sum_{o'\in\mathcal{N}_k(o)} \delta\big(T(o),T(o')\big) \tag{9}$$

If the neighborhoods are all empty or contain too many orders the NRQ loses its informative value. Thus, finding a suitable radius for NRQ_r requires careful fine-tuning to each individual instance of the problem while choosing a neighborhood size k is more straightforward. NRQ_r is still of interest, however, since it yields more accurate results when order destinations are spread out very irregularly, which is the case with the data at hand.

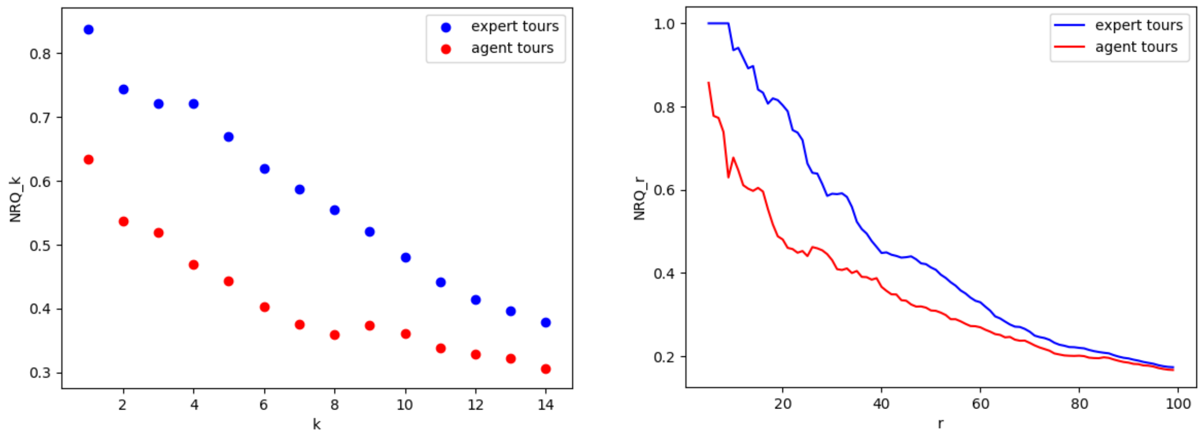

Fig. 3 (a) Neighborhood Relation Quotient with k-neighborhoods; (b) Neighborhood Relation Quotient with r-neighborhoods

As can be seen in Fig. 3 (a), regardless of how k is chosen, the NRQ_k scores of the expert tours are consistently higher than those of the agent tours. The same observation can be made in 4 (b) which shows the NRQ_r scores for both solutions with varying values for the radius r. Comparing NRQ_r and NRQ_k scores, it is apparent that NRQ_r scores are generally higher due to the irregular distribution of order destinations. Additionally, NRQ_r scores show a smaller difference between expert and agent tours. These results are consistent with our observations from Fig. 2.

Possibly related to the issue of tour separability is the total sum distance that the delivery vehicles need to travel. For the expert tours, the total distance is 1,750km while the agent's tours have a total travel distance of 1,975km (excluding both unassigned orders). Again, the agent achieves a reasonable result, but cannot quite reach the quality of the tours created by experts. Some of the supposed reasons and further opportunities for improvement are discussed within the next section.

One aspect where the agent can clearly outperform human experts is the time required to create tours (see Fig. 4). On average, it can assign up to 600 orders in a minute. Although the build duration is exponential in the number of orders, it is still extremely fast compared to amount of time a human expert spends on the planning of tours, which is on the order of multiple hours. The agent can still create solutions to problems with 2,000 orders within minutes.
As can be seen in Fig. 5, the training duration of 500,000 steps is sufficient for the average Q-values to converge, which corresponds to a training duration of approximately half an hour. Thus, the adaptability of the methodology is extremely high, since the agent can easily be retrained every time there is a change in external conditions or constraints. But even without retraining, the trained agent was successfully applied to another hub location and generated results with the same overall quality.

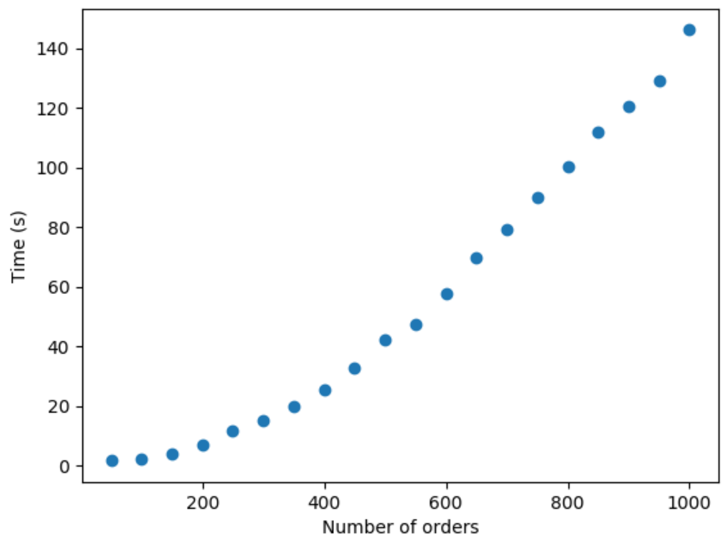

Fig. 4: Average time to build tours on a notebook (quad-core CPU with 2.20GHz) depending on the number of orders

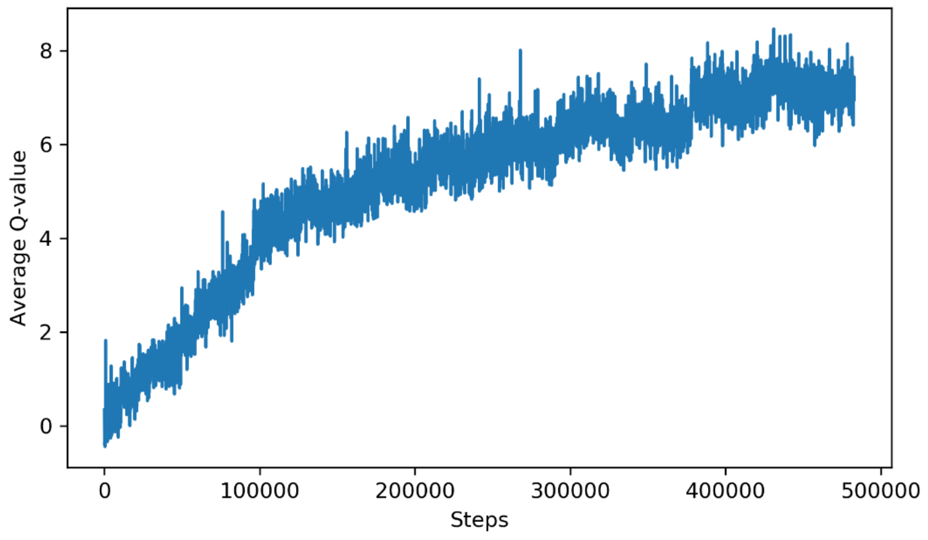

Fig. 5: Development of the best action's Q-value over time. Each datapoint shown here is an average over one episode

6. Discussion

Our approach presented in section 4 can be used to automatically create initial tours, which come reasonably close to the quality of final tours created by experts, as seen in section 5. Thus, the overall time spent creating daily tours is minimized, by delivering a support tool, which computes high-quality initial tours. A few challenges and opportunities for improvement can be identified.

First, note that the way we built our test data favors the expert tours because only orders from finished tours are included. In reality, not all given orders are assigned to a tour on a given day, instead, many orders are moved to different days. Since our data does not include information about which orders were moved to different days, we can only evaluate the agent against the "perfect" end result. In other words, human experts have the ability to defer orders to a future date, while the agent does not. Additionally, two of the expert tours break hard restrictions that our agent does not break, giving the experts an advantage.

Nevertheless, the quality of the tours created by the agent offers room for improvement. We observe that the quality of the tours depends strongly on how the order list is sorted initially. While this poses a non-negligible challenge to the robustness of agent-based tour creation, it also offers the potential to integrate expert feedback into the system. Every manual change made to a created tour by the domain expert can be saved and provide a basis to a two-level approach. The second layer can be seen as learning agent with the goal to improve the tour creation by changing the order of the incoming list and thereby generating insights on how the assorting influences the tour generation process. Hence, this second agent can learn from the changes made by domain experts and adapt the sorting policy accordingly to reflect these changes in the future.

An alternative approach to address the problem of sensitivity to the ordering of the list might be to circumvent it entirely by simply evaluating the utility of all the orders in parallel and selecting the most promising one. This may negatively affect the average run time, since compared to the current approach, at least equally as many but likely more computations need to be performed. These computations, however, can be performed in parallel rather than sequentially, thus potentially limiting the additional computation time given suitable hardware. More investigation is needed to characterize the trade-off between run time and solution quality with this approach more precisely.

It is also conceivable that the quality of the agent's decision-making can be further improved by providing it with a set of observations that describe the state of the problem more thoroughly. This is challenging because the size of the problem, i.e. the number of orders given as input is not determined in advance and can vary in each episode. A possible way to approach this issue may be the use of some type of recurrent neural network such as Long Short-Term Memory networks (Hochreiter & Schmidhuber, 1997), which can be presented with multiple inputs one after the other, while retaining information about previous inputs in their states.

The reward function presented in subsection 4.2 is built to reflect insights of the tour planning experts. Although this makes it unlikely to discover novel, advantageous patterns in the tour planning process, it guarantees a certain compatibility between automatically and manually created tours. In this first automated tour planning approach the main goal was to show the capability of a reinforcement learning agent to learn the necessary patterns to generate valid tours which are comparable to manually created ones. In the future, we will investigate agents which receive less specific guidance in terms of very granular rewards for each possible action. Instead, they will only receive information regarding hard constraints and rewards pertaining to the goal of minimizing the number of tours required to serve all given orders. They will thus have more freedom to discover useful rules and behavioral patterns themselves.

Finally, as the system presented here is only a first proof-of-concept, many more details and additions to the problem can be considered and implemented. For instance, we only consider short-range deliveries here, i.e. orders which can be delivered within a single day. Of course, this does not reflect reality, as there are also depots from which mostly long-range deliveries are dispatched, as well as depots from which a mixture of short- and long-range deliveries are dispatched. A different aspect related to this issue is the size of the utilized vehicles, i.e. larger vehicles can often be used for long-range deliveries due to regulations. Hence, for a more mature system, a method like ours should be able to work with vehicles of varying sizes, ideally not predefined.

7. Conclusion

In the preceding text, we present a novel approach to solve practical tour planning problems based on reinforcement learning. This approach drastically reduce the overall time required to plan initial tours for any given day even with big varieties between delivery days, order sizes or locations. This is of great significance due to the highly dynamic nature of the logistics sector, where orders are often received or changed on short notice, such that tour planning cannot start far in advance.

With our approach, initial tours can be created within minutes in such a way that only minimal modifications by human experts are required afterwards. Hence, the time to properly plan a full day of deliveries is significantly reduced.

We show that these initial tours created by our reinforcement learning agent are not only valid (i.e. the agent successfully learns not to violate any hard constraints), but also that their quality, measured by the total travel distance of the tours as well as the introduced Neighborhood Relation Quotient (defined in section 5), comes close to the quality of the manually created and conducted tours. Thus, while speed is the main advantage of the automated approach, it can also compete with human experts in terms of quality. By using reinforcement learning, our approach is highly adaptable to a variety of situations, such as different hub location or number of orders to be delivered.

Further, we point out that better results can potentially be achieved by modifying the set of observations received by the agent, exploring different reward functions, and taking a more sophisticated approach to the ordering of the initial order list. These additions and modifications have the potential to reduce the planning time even further.

Nevertheless, even without further improvements, our approach can already be successfully applied to real-world scenarios, creating added value for businesses for two reasons. First because human domain experts do not need to start the tour planning from zero at any given point in time. Second the developed solution can be transferred and easily adapted to other locations and transport conditions.

References

Bengio, Y., Lodi, A., & Prouvost, A. (2018). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *arXiv preprint arXiv:1811.06128*.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research, 12*, 568-581.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research, 6*, 791-812.

Eurostat. (2018). *Energy, transport and environment indicators – 2018 edition.* Luxembourg: Publications Office of the European Union.

Francis, P., Smilowitz, K., & Tzur, M. (2008). The period vehicle routing problem and its extensions. In *The vehicle routing problem: latest advances and new challenges* (pp. 73-102). Springer.

Giannikos, I. (1998). A multiobjective programming model for locating treatment sites and routing hazardous wastes. *European Journal of Operational Research*.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, (pp. 249-256).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*, 1735-1780.

Kumar, S., & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. (S. R. Publishing, Ed.) *Intelligent Information Management*.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research, 59*, 345-358.

Lee, T.-R., & Ueng, J.-H. (1999). A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*.

Lenstra, J. K., & Kan, A. H. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11*, 221-227.

Lin, L.-J. (1993). *Reinforcement learning for robots using neural networks.* Tech. rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015, 2). Human-level control through deep reinforcement learning. *Nature, 518*, 529-533.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, (pp. 807-814).

Nazari, M., Oroojlooy, A., Snyder, L., & Takac, M. (2018). Reinforcement Learning for Solving the Vehicle Routing Problem. *Advances in Neural Information Processing Systems*, (pp. 9861-9871).

Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in Neural Information Processing Systems*, (pp. 9839-9849).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* Cambridge: MIT press.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem.* SIAM.

Watkins, C. J. (1989). *Learning from delayed rewards.* Ph.D. dissertation, King's College, Cambridge.

Zirour, M., Liong, C. Y., Ismail, W. R., & Omar, K. (2008). Vehicle routing problem: models and solutions. *Journal of Quality Measurement and Analysis JQMA, 4*, 205-218.