Operations Research Proceedings

Natalia Kliewer · Jan Fabian Ehmke
Ralf Borndörfer *Editors*

Operations Research Proceedings 2017

Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Freie Universiät Berlin, Germany, September 6–8, 2017

GOR
Gesellschaft für Operations Research e.V.

Springer

Operations Research Proceedings

GOR (Gesellschaft für Operations Research e.V.)

More information about this series at http://www.springer.com/series/722

Natalia Kliewer · Jan Fabian Ehmke
Ralf Borndörfer
Editors

Operations Research Proceedings 2017

Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Freie Universiät Berlin, Germany, September 6–8, 2017

Springer

Editors
Natalia Kliewer
Department of Information Systems
Freie Universität Berlin
Berlin
Germany

Ralf Borndörfer
Department of Mathematics
Freie Universität Berlin
Berlin
Germany

Jan Fabian Ehmke
Management Science Group
Otto-von-Guericke Universität Magdeburg
Magdeburg
Germany

ISSN 0721-5924 ISSN 2197-9294 (electronic)
Operations Research Proceedings
ISBN 978-3-319-89919-0 ISBN 978-3-319-89920-6 (eBook)
https://doi.org/10.1007/978-3-319-89920-6

Library of Congress Control Number: 2018938366

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book contains a selection of refereed short papers presented at the Annual International Conference of the German Operations Research Society (OR2017), which took place at the Freie Universität Berlin, Germany, September 6–September 8, 2017. Over 900 participants attended the conference—practitioners and academics from mathematics, computer science, business administration and economics, and related fields. The scientific program included about 600 presentations. The conference theme, Decision Analytics for the Digital Economy, placed emphasis on the process of researching complex decision problems and devising effective solution methods toward better decisions. This includes mathematical optimization, statistics, and simulation techniques. Yet, such approaches are complemented by methods from computer science for the processing of data and the design of information systems. Recent advances in information technology enable the treatment of big data volumes and real-time predictive and prescriptive business analytics to drive decision and actions. Problems are modeled and treated under consideration of uncertainty, behavioral issues, and strategic decision situations.

Altogether, 100 submissions have been accepted for this volume (acceptance rate 63%), including papers from the GOR doctoral dissertation and master's thesis prize winners. The submissions have been evaluated by the stream chairs for their suitability for publication with the help of selected referees. Final decisions have been made by the editors of this volume.

We would like to thank the many people who made the conference a tremendous success, in particular the members of the organizing and the program committees, the stream chairs, the 14 invited plenary and semi-plenary speakers, our exhibitors and sponsors, our host Freie Universität Berlin, the many people organizing the conference behind the scenes, and, last but not least, the participants from about 46 countries. We hope that you enjoyed the conference as much as we did.

Berlin, Germany — Natalia Kliewer
Magdeburg, Germany — Jan Fabian Ehmke
Berlin, Germany — Ralf Borndörfer
January 2018

Contents

Part XXII Business Track

Part I
Awards

Solving the Time-Dependent Shortest Path Problem Using Super-Optimal Wind

Adam Schienle

1 Introduction

With air travel steadily on the rise and the increased fuel burn associated to it, it is ever more important that aircraft fly efficient routes. Planning such routes is a fundamental process of flying: commonly, a route is planned a few hours before the flight, focussing on key factors such as overfly costs and fuel burn. According to the Air Transport Action Group [1], around 1.5 billion barrels of fuel are burnt every year, corresponding to 93.75 billion USD [6]. A decrease of just 0.25% would add up to 234.375 million USD. There is also a visible impact for airlines: Lufthansa's total fuel consumption in 2016 amounted to 9 055 550 tons [7]. Decreasing this by 0.25% leads to 22 639 tons less fuel being burnt, or savings of almost 11.67 million USD per year. In terms of CO_2, this is equivalent to a reduction of more than 70 tons per year [7].

The need for efficient routes gives rise to the Flight Planning Problem (FPP), which is the problem of finding a minimum cost trajectory between two airports on the Airway Network, a directed graph. In general, the objective function consists of several summands, such as fuel costs, overfly costs and crew costs. In this paper, however, we shall concentrate on minimising the fuel costs. We further assume that aircraft fly levelly on a given altitude and neither climb nor descend. In this setting, fuel consumption is equivalent to flight time, which reduces FPP to the *Horizontal Flight Planning Problem* (HFPP). Since winds have a strong impact on flight time and because of the time-dependency of the weather, we can model HFPP as a Time-Dependent Shortest Path Problem (TDSPP).

TDSPP has been extensively studied in the literature, with particular emphasis on road networks. Dijkstra's algorithm yields an optimal solution in polynomial time; however, for large networks, several speedup techniques have been developed,

A. Schienle (✉)
Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany
e-mail: schienle@zib.de

N. Kliewer et al. (eds.), *Operations Research Proceedings 2017*, Operations Research Proceedings, https://doi.org/10.1007/978-3-319-89920-6_1

allowing to curb runtimes by several orders of magnitude with respect to Dijkstra's algorithm [2]. Most of them rely on a preprocessing phase, in which either some shortest paths or other auxiliary data is precomputed and stored to speed up the query. For a comprehensive survey, see [2].

Throughout this paper, a *weighted graph* will always refer to a pair (G, T), consisting of the actual (directed) graph G and a possibly time-dependent weight function $T: A \times [0, \infty) \to [0, \infty)$, mapping an arc $a \in A$ and a time $\tau \in [0, \infty)$ to the travel time $T(a, \tau)$ on a.

The ground distance $d_G(a)$ of an arc $a \in A$ on the Earth's surface is constant, and we assume that aircraft fly with constant air speed[1] v_A. In contrast, the *ground speed* $v_G(a, \tau)$ of an aircraft is dependent on the prevailing wind conditions on the arc and given by the formula

$$v_G(a, \tau) = \sqrt{v_A^2 - w_C(a, \tau)^2} + w_T(a, \tau) \quad \forall a \in A, \tau \in [t_0, t_r], \tag{1}$$

where $w_C(a, \cdot)$ and $w_T(a, \cdot)$ are the crosswind and trackwind components of the wind vector, i.e., the components perpendicular and parallel to the current flight direction. Ground speed and ground distance are linked via the relation

$$T(a, \tau) = \frac{d_G(a)}{v_G(a, \tau)}. \tag{2}$$

2 Super-Optimal Wind

We are looking to solve the TDSPP model of HFPP to optimality by using an appropriate shortest path algorithm. A natural choice would be Dijkstra's algorithm; in practice, however, the time to plan a flight is limited and for the most part, this process takes place shortly before the aircraft departs. In particular, this means that query times should be as short as possible. In this paper, we restrict ourselves to the discussion of the A* algorithm, introduced in [5]. For an overview of other algorithms and their applicability to HFPP, see [3].

The intricacy with A* is to find a suitable potential function $\pi_t: V \to [0, \infty)$, which for every $v \in V$ underestimates the cost of a shortest v-t-path in (G, T). We define the reduced cost of an arc $(u, v) \in A$ at time τ as

$$T'((u, v), \tau) = T((u, v), \tau) - \pi_t(u) + \pi_t(v), \tag{3}$$

and call π_t *feasible on* (G, T) if for every arc $(u, v) \in A$ and for every $\tau \geq 0$, we have $T'((u, v), \tau) \geq 0$. If π_t is feasible, running A* is equivalent to running Dijkstra's algorithm on G using the reduced costs.

[1] Speed relative to the surrounding air mass.

To obtain a feasible potential function, we have to find a lower bound for the travel time on the arcs. To this end, we introduce the concept of *Super-Optimal Wind* to underestimate the travel time. While it is possible to minimise the travel time function directly, this takes too long for practical purposes. Furthermore, it requires knowledge of the airspeed in advance, as opposed to constructing the Super-Optimal Wind vector.

We assume that weather is given for a finite set of times $\{t_0, t_1, \dots, t_r\}$, and between the t_i, the weather data is interpolated to obtain the wind vector $w(a, \tau)$. Let $t_0 = \tau_0 < \tau_1 < \cdots < \tau_n = t_r$ be a discretisation of $[t_0, t_r]$ such that $\tau_i - \tau_{i-1} = \triangle$ for some $\triangle > 0$ and for all $i = 1, \dots, n$. To ensure that for every $i \in \{0, \dots, n-1\}$ we always find a $j \in \{0, \dots, r-1\}$ such that $[\tau_i, \tau_{i+1}] \subset [t_j, t_{j+1}]$, we require that $r \mid n$. We then define for $i = 1, \dots, n$

$$\underline{w}_C^{(i)}(a) = \min_{\tau\in[\tau_{i-1},\tau_i]} |w_C(a,\tau)| \quad \text{and} \quad \overline{w}_T^{(i)}(a) = \max_{\tau\in[\tau_{i-1},\tau_i]} w_T(a,\tau),$$

which are the minimum crosswind and maximum trackwind on each discretisation step. The vector defined through its cross- and trackwind components

$$w_{\text{s-opt}}^{(i)}(a) = (\underline{w}_C^{(i)}(a), \overline{w}_T^{(i)}(a))$$

is called *Super-Optimal Wind* vector, and is used to overestimate the ground speed (note that by (2), this is equivalent to underestimating the travel time). We define

$$\overline{v}_G^{(i)}(a) = \sqrt{v_A^2 - \underline{w}_C^{(i)}(a)^2} + \overline{w}_T^{(i)}(a),$$

and let $\overline{v}_G(a) := \max_{i\in\{1,\dots,n\}} \overline{v}_G^{(i)}(a)$. It is easy to prove the following lemma:

Lemma 1 *The inequality $v_G(a,\tau) \le \overline{v}_G(a)$ holds for all $\tau \in [t_0, t_r]$.*

Note that in particular, if $v_G^*(a) = \max_{\tau\in[t_0,t_r]} v_G(a,\tau)$ denotes the maximum ground speed in $[t_0, t_r]$, we also have

$$\overline{v}_G(a) \ge v_G^*(a). \tag{4}$$

Define $r_a^* := \max_{\tau\in[t_0,t_r]} \sqrt{w_C(a,\tau)^2 + w_T(a,\tau)^2}$, the maximum overall wind speed on $a \in A$. Under the condition that $v_A \ge 2r_a^*$, which in practice is always the case, we obtain

Theorem 1 *Suppose $v_A \ge 2r_a^*$. Then there exists a constant $C > 0$ such that*

$$0 \le \overline{v}_G(a) - v_G^*(a) \le C\triangle.$$

The first inequality follows directly from (4), and the proof for the second inequality can be found in [3]. Analogous to the ground speed, we define

$$\underline{T}(a) = \min_{i \in \{1,\dots,n\}} \underline{T}^{(i)}(a) := \min_{i \in \{1,\dots,n\}} \frac{d_G(a)}{\overline{v}_G^{(i)}}.$$

Letting $T_a^* = \min_{\tau \in [t_0, t_r]} T(a, \tau)$ and following (2), one readily obtains

Corollary 1 *Suppose $v_A \geq 2r_a^*$. Then there exists a constant $C' > 0$ such that for any arc $a \in A$, we have*

$$0 \leq T_a^* - \underline{T}(a) \leq C' \triangle.$$

In particular, $\underline{T}(a)$ underestimates the travel time needed to traverse an arc, and the error is bounded linearly in the discretisation step.

2.1 The Super-Optimal Wind Potential Function

For the A* algorithm, we seek to find a good and feasible potential function. For HFPP, we can exploit the fact that in our application, there is a small number of possible target nodes (corresponding to airports). Since our objective in HFPP is to minimise travel time, we construct the weighted graph $(G, \underline{T})$, where $\underline{T}\colon A \to [0, \infty)$ maps an arc $a \in A$ to the underestimated travel time $\underline{T}(a)$ obtained through the Super-Optimal Wind computation, i.e., $\underline{T}(a) \leq T(a, \tau)$ for all $\tau \in [t_0, t_r]$ and all arcs $a \in A$. Note that $(G, \underline{T})$ is a weighted graph with static arc weights, and we can without effort compute an all-to-one shortest path tree for each target node t. We then define a potential function for HFPP as

$$\pi_t(v) = \min \left\{ \textstyle\sum_{a \in P} \underline{T}(a)\colon\ P \text{ is a } (v, t) - \text{path} \right\}.$$

Note that this is equivalent to running the ALT-Algorithm [4] with the target node as the only landmark.

Theorem 2 *The following two statements hold:*

(i) $\pi_t(\cdot)$ *is feasible in* $(G, \underline{T})$.
(ii) $\pi_t(\cdot)$ *is feasible in* (G, T).

For details on the proof, see [8]. In particular, Theorem 2 yields that running the A* algorithm on (G, T) is equivalent to running Dijkstra's algorithm on the reduced cost graph (G, T') obtained from (3), and A* visits at most as many nodes as Dijkstra's algorithm.

2.2 Validation of Super-Optimal Wind

Theorem 1 and Corollary 1 state that the absolute error of the overestimated ground speed with respect to the optimum ground speed is bounded linearly in the dis-

Table 1 Errors and runtimes of Super-Optimal Wind computation

Altitude (ft)	Segments (#)	Av. error (%)	Max. error (%)	Computation time (s)
37000	344936	0.041	5.263	2.50
34000	344920	0.045	5.882	1.59
31000	338567	0.045	8.333	2.52

cretisation step. To assess the quality of the travel time underestimation with Super-Optimal Wind computationally, we ran it on several real-world instances (cf. [3]), each instance using 28 threads.

As our weather prognoses are given at times t_i all spaced three hours apart, a natural choice for the discretisation step is $t_{i+1} - t_i = \tau_{i+1} - \tau_i = \triangle = 3h$. We found this choice to already yield excellent results, as shown in Table 1, which contains the average and maximum values of the relative error $\rho(a) = \frac{T(a)-T_a^*}{T_a^*} \forall a \in A$. The results show that the Super-Optimal Wind is an excellent underestimator in practice, and can be computed fast.

3 A Case Study

In the following, we investigate the effect of wind on a route. In particular, we consider a flight between Taipei-Taoyuan (TPE) and New York-John F. Kennedy (JFK). We use weather data from the 25th April 2017, starting the route on the same day at 0300 UTC. We assume an aircraft flying at 37 000ft (≈11 277 m).

Often, routes lie close to the geodesic, but if aircraft can take advantage of strong tailwinds, they commonly divert to areas with more favourable winds. In Fig. 1, we observe that the search space for A* is doughnut-shaped, which is due to the fact that on that day, there was an unusually strong jetstream on the Northern Pacific, rendering the Pacific route shown in green more efficient than the polar route (red), which would seem a more natural choice. When one compares the ground distances of the northerly route to the Pacific route, one finds the red route to be almost 1880 km shorter than the green route – but considering wind, the green route is 131 s faster than the red route, or roughly 0.26% of the total travel time. As this translates directly to fuel burn, it makes sense to favour the seemingly longer Pacific route over the polar route.

In Fig. 1, we also observe that A* visits significantly fewer arcs than Dijkstra's algorithm. This also impacts the runtime: between TPE and JFK, A* yields a speedup factor of 11 over Dijkstra's algorithm. For a more detailed discussion on the speedup of A* over many instances, we refer the reader to [8].

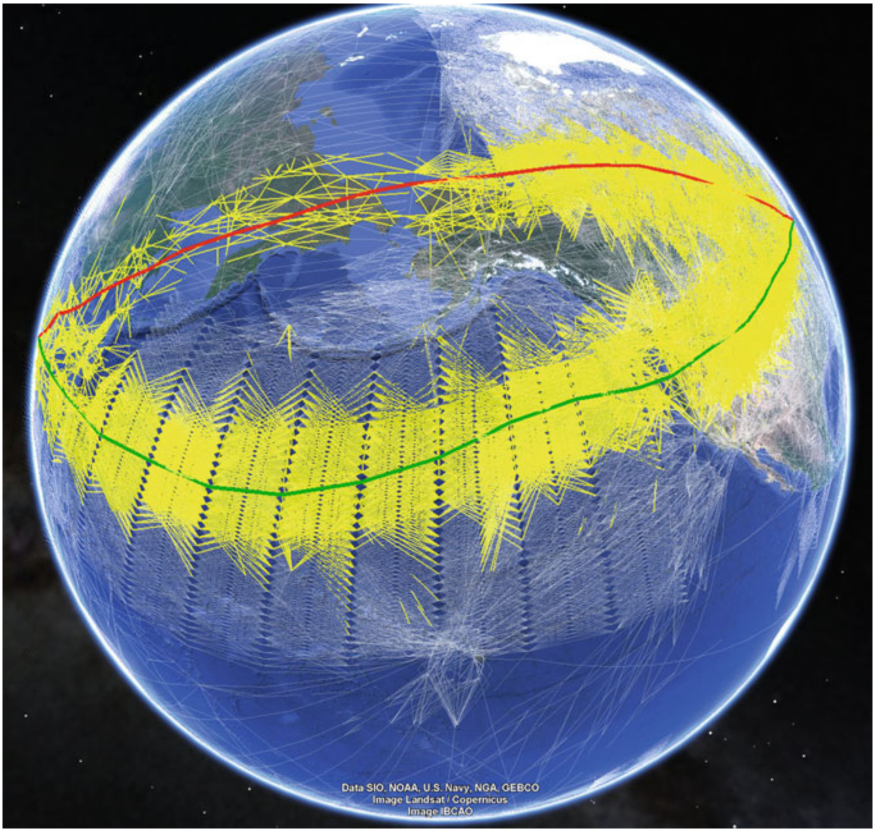

Fig. 1 Search spaces for Dijkstra's algorithm (white) and A* (yellow) between TPE and JFK. The route closest to the geodesic is marked red, the shortest route shown in green (Map data: Google, Landsat/Copernicus/IBCAO)

References

1. Air Transport Action Group (ATAG). (2017). Facts and figures. http://www.atag.org/facts-and-figures.html. Retrived 07 Aug 2017.
2. Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., & Werneck, R. F. (2015). Route planning in transportation networks. Technical report, Microsoft Research. Updated version of the technical report MSR-TR-2014-4.
3. Blanco, M., Borndörfer, R., Hoang, N. D., Kaier, A., Schienle, A., Schlechte, T., & Sclobach, S. (2016). Solving time dependent shortest path problems on airway networks using super-optimal wind. *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, *54*. (epub ahead of print).
4. Goldberg, A. V., & Harrelson, C. (2004). Computing the shortest path: A* search meets graph theory. Technical Report MSR-TR-2004-24, Microsoft Research, Vancouver, Canada, July 2004.
5. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*, 100–107.

6. International Air Transport Association. IATA Price Analysis. (2017). https://www.iata.org/publications/economics/fuel-monitor/Pages/price-analysis.aspx. Retrived 07 Aug 2017.
7. Lufthansa Group. (2017). Balance – Nachhaltigkeitsbericht der Lufthansa Group 2016. https://www.lufthansagroup.com/fileadmin/downloads/de/verantwortung/LH-Nachhaltigkeitsbericht-2017.pdf. Retrived 07 Aug 2017.
8. Schienle, A. (2016). Shortest paths on airway networks. Master's thesis, Freie Universitt Berlin.

Anticipation in Dynamic Vehicle Routing

Marlin W. Ulmer

1 Motivation

Decision making in real-world routing applications is often conducted under incomplete information. Vehicle dispatchers deal with uncertainty in travel times, service times, customer demands, and customer requests. This information is only revealed successively during the execution of the route. Technological advances allow dispatchers to adapt their decisions to new information [13]. These developments pave "the way for models of a dynamic nature" [1]. Nevertheless, current decisions influence later outcomes. Anticipation, that is, "incorporating information about the uncertainty of future events" [8] is necessary to avoid myopic decision making. These advances and challenges lead to the field of stochastic and dynamic vehicle routing problems (SDVRPs), a field gaining growing attention in the research community. This attention is reflected in the increasing amount of research on SDVRPs [5]. As [7] state, addressing these new developments and therefore SDVRPs "may necessitate new views, paradigms, and models for decision support." In essence, the field of SDVRPs poses many challenges for the research community in both models and algorithms and has not been studied comprehensively yet.

The canonical model for SDVRPs is a Markov decision process (MDP, [6]). MDPs model subsequent decision states connected by decisions and stochastic realizations of information. Solving the MDP for SDVRPs is challenging due to the "Curses of Dimensionality" [4]. Generally, state space, decision space, and transition space are vast. Methods of approximate dynamic programming (ADP) address these challenges. Still, these methods are not yet established in the field of SDVRP due to the high complexity of the routing problems [10]. In the following, we recall the func-

M. W. Ulmer (✉)
Technische Universität Braunschweig, Braunschweig, Germany
e-mail: m.ulmer@tu-braunschweig.de

N. Kliewer et al. (eds.), *Operations Research Proceedings 2017*, Operations Research Proceedings, https://doi.org/10.1007/978-3-319-89920-6_2

tionality and notation of the MDP. We then define and tailor methods of ADP to the specific needs of SDVRPs. We show how ADP enables substantial improvements compared to state-of-the-art benchmark policies.

2 Markov Decision Process

Within the (finite) MDP, a number of decision points $\mathcal{K} = \{0, \ldots, K-1\}$ occurs subsequently. Here, K can be a random variable. For each decision point $k \in \mathcal{K}$, a set of states $\mathcal{S}_k$ is given, combined in the finite set of states $\mathcal{S}$. State $S_0 \in \mathcal{S}$ denotes the initial state and state $S_K \in \mathcal{S}$ denotes the termination state. For each decision point $k \in \mathcal{K}$ and for each state $S_k \in \mathcal{S}$, a subset of decisions $\mathcal{X}(S_k) \subseteq \mathcal{X}$ of the overall decision space $\mathcal{X}$ is given. The combination of a state S_k and a decision $x \in \mathcal{X}(S_k)$ leads to a (deterministic) post-decision state (PDS) $S_k^x \in \mathcal{P}$ with $\mathcal{P}$ the overall set of post-decision states. It further leads to an immediate reward (or costs) $R(S_k, x)$ with $R : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$. Given PDS S_k^x, a stochastic transition $\omega_k \in \Omega$ leads to the next state $(S_k^x, \omega_k) = S_{k+1} \in \mathcal{S}$.

A solution for the MDP is a *decision policy* $\pi \in \Pi$. Decision policies determine the decision to be selected given a specific state. A decision policy $\pi \in \Pi$ is a sequence of decision rules $(X_0^\pi, X_1^\pi, \ldots, X_{K-1}^\pi)$ for every decision point $k \in \mathcal{K}$. Each decision rule $X_k^\pi(S_k)$ specifies the decision to be selected in state S_k. Optimal decision policies $\pi^* \in \Pi$ select decisions leading to the highest expected rewards and therefore maximize the sum of expected rewards. In a specific state S_k, the optimal decision $X_k^{\pi^*}(S_k)$ can be derived by maximizing the sum of immediate and expected future rewards as shown in the Bellman Equation (1):

$$X_k^{\pi^*}(S_k) = \underset{x \in \mathcal{X}(S_k)}{\arg\max} \left\{ R(S_k, x) + \mathbb{E}\left[\sum_{j=k+1}^{K} R(X_j^{\pi^*}(S_j)) | S_k \right] \right\}. \tag{1}$$

The expected future rewards are also known as the value $V(S_k^x)$ of PDS S_k^x.

3 Approximate Dynamic Programming

For small MDPs, the values can be calculated recursively to eventually obtain an optimal policy. Still, for SDVRPs, this is usually hardly possible due to the "Curses of Dimensionality" [4]. The state, decision, and transition spaces are generally vast. Thus, solution methods aim on approximating the values by means of simulations and approximate dynamic programming (ADP). In the following, we present two ADP-methods, the dynamic lookup table (DLT) and the offline–online rollout algorithm, capturing the complexity of SDVRPs.

3.1 The Dynamic Lookup Table

One way of approximating the values is value function approximation (VFA).[1] The VFA procedure starts with initial values $\widehat{V}^0$. These values define an initial policy π^0 with respect to the Bellman Equation (1). The VFA then frequently simulates MDP-realizations. In every simulation run i, the VFA uses the current policy π^{i-1} for decision making within the simulation. After the simulation run, the values $\widehat{V}^{i-1}$ are updated with respect to the observed values. The new values $\widehat{V}^i$ then define a new policy π^i. This procedure is continued until a stopping criterion is reached. Subsequently, the VFA approximates the real values and the optimal policy.

The advantage of VFAs is that the simulations are conducted only once *offline* before the actual implementation of the policy. Thus, these methods allow immediate responses to new information, for example, customer requests. Still, to apply VFA, the value for every PDS needs to be stored. For SDVRPs, the number of PDSs is vast and an aggregation is necessary. Thus, PDSs are reduced to a vector of state features (like point of time). This vector space is then partitioned to a lookup table (LT). Conventional LT-partitionings are static. The partitioning is defined a-priori. This leads to disadvantages in the approximation process because "important" LT-areas are represented in insufficient detail while other areas are not sufficiently observed. To alleviate these shortcomings, we propose a dynamic lookup table. This table starts with an initial partitioning and subsequently refines the partitioning in "important" areas. Areas are important if a sufficient number of observations allows and a high variance in the observed values demands a refinement. Thus, the DLT is able to adapt to the approximation process. For a detailed definition and algorithmic procedure of the DLT, we refer to [14].

3.2 Offline–Online Rollout Algorithm

One shortcoming of VFA in general is that not all but only a few state features can be considered in the evaluation. For SDVRPs, VFAs are usually not able to capture spatial information such as customer and vehicle locations [14]. To integrate these details in the evaluation of a PDS, the simulation needs to be conducted *online* in the actual decision state. One prominent online simulation method is the post-decision rollout algorithm (RA). Originating from a particular PDS, an RA simulates a number of trajectories into the future. To determine decisions within the simulations, a base policy is used. The PDS is evaluated with respect to the observed rewards in the simulation. This evaluation is then used in the Bellman Equation to determine the actual decision.

Because the simulations are conducted online, RAs have the disadvantage that the time for simulations is highly limited. Usually, the base policy is a runtime-

[1] Notably, in the following, we present *non-parametric* VFA because parametric VFAs are often not able to capture the complex value function structure of SDVRPs [11].

efficient rule of thumb. This inferior decision making within the simulations leads to a discrepancy between simulated and realized outcome. Thus, the evaluation of the PDS may be distorted. To alleviate this disadvantage, we integrate the DLT-policy as base policy in the RA. This leads to an offline–online RA. Within the simulation, decision making is conducted by the offline DLT. The simulation's outcome is then used to determine the actual decision. Thus, the simulation is reinforced and provides better approximation and/or less simulation runs. We further improve the RA's performance by integrating the well-known *Fully Sequential Procedure for Indifference Zone Selection* (IZS) by [3]. For a detailed definition of IZS and the offline–online RA, we refer to [15].

4 Case Study

In the following, we apply DLT and offline–online RA to the dynamic vehicle routing problem with stochastic requests (VRPSR) by [9].

4.1 Problem Definition and Markov Decision Process

In the VRPSR, a vehicle serves customers in a service area within a shift. The vehicle starts and ends its tour at a depot. The customers request service during the shift and are unknown beforehand. Decisions are made about the acceptance or rejection of the new requests and the according routing update. The objective is to maximize the expected number of accepted requests. In the according MDP, a state occurs when the vehicle served a customer. A state S_k consists of the point of time t_k, the currently planned tour θ_k, and the set of new requests $\mathcal{C}_k^r$. Tour θ_k starts at the vehicle's current location, traverses the customers still to serve, and ends at the depot. A decision x determines the subset of requests to accept $\mathcal{C}_k^a$ and the according routing update θ_k^x. The reward is the number of accepted requests: $R(S_k, x) = |\mathcal{C}_k^a|$. The PDS contains the point of time t_k, and the new routing θ_k^x. The stochastic transition ω_k updates the origin of θ_k^x and provides a set of new requests.

4.2 Computational Experiments

In the following, we describe how we tune DLT and offline–online RA to the needs of the VRPSR. We present the benchmark policies and the results. For the DLT, we select the features point of time t_k and free time budget b_k^x. The free time budget reflects the amount of time available to serve additional requests. It is defined as the difference between remaining time in the shift and the tour duration of θ_k^x. The DLT is therefore two-dimensional. We run 1 million approximation runs and update the

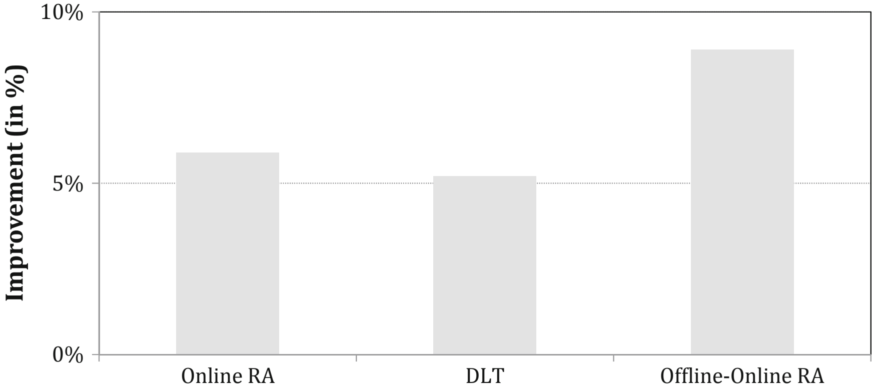

Fig. 1 Average improvement of the policies compared to AI

values with a running average. The partitioning starts with equidistant intervals of 16 min and refines entries of the DLT by splitting them equally into 2×2 new entries. The smallest entry-size is 1 min. For the offline–online RA, we run 16 simulations runs in every state. As benchmark policies, we draw on the current state-of-the-art policy, anticipatory insertion (AI) by [2]. We also compare our methods with the online RA by [12]. The online RA draws on a myopic base policy within the simulations. We compare our methods for a variety of instance settings differing in the number of dynamic requests and the spatial distribution of the requests. The average improvement of the three policies DLT, offline–online RA, and online RA compared to AI are depicted in Fig. 1. We observe that all three ADP-methods outperform the benchmark policy by more than 5%. Notably, the offline DLT is able to achieve similar results as the online RA while not requiring any online runtime. The offline–online RA combines the advantages of the DLT's extensive offline simulations with the online RA's detailed state consideration and achieves improvements of about 9%.

5 Conclusion

Stochastic dynamic vehicle routing problems gain significant interest in the research community. To solve the according MDPs, we have proposed two novel methods of ADP. For the dynamic vehicle routing problem with stochastic requests, we have shown how these methods significantly outperform conventional policies. Future research in stochastic dynamic vehicle routing should focus on applications and methodology. Promising research areas with high dynamism are the growing fields of same-day delivery and shared mobility. The proposed ADP-methodology can be generalized. Instead of a LT-structure, the state space may be dynamically partitioned by means of clustering algorithms. Further, the combination of online and offline ADP may be determined based on a state's value variance and number of observations.

References

1. Gendreau, M., Jabali, O., & Rei, W. (2016). 50th anniversary invited article-future research directions in stochastic vehicle routing. *Transportation Science*, *50*(4), 1163–1173.
2. Ghiani, G., Manni, E., & Thomas, B. W. (2012). A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, *46*(3), 374–387.
3. Kim, S.-H., & Nelson, B. L. (2001). A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *11*(3), 251–273.
4. Powell, W. B. (2011). *Approximate dynamic programming: solving the curses of dimensionality* (2nd ed.)., Wiley series in probability and statistics New York: Wiley.
5. Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, *67*(1), 3–31.
6. Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. New York: Wiley.
7. Savelsbergh, M., & Van Woensel, T. (2016). 50th anniversary invited article-city logistics: Challenges and opportunities. *Transportation Science*, *50*(2), 579–590.
8. Speranza, M. G. (2016). Trends in transportation and logistics. *European Journal of Operational Research*.
9. Thomas, B. W. (2007). Waiting strategies for anticipating service requests from known customer locations. *Transportation Science*, *41*(3), 319–331.
10. Ulmer, M. W. (2017). *Approximate dynamic programming for dynamic vehicle routing* (Vol. 61)., Operations research/computer science interfaces series Berlin: Springer.
11. Ulmer, M. W. & Thomas, B. W. (2017). Meso-parametric value function approximation for dynamic customer acceptances in delivery routing (submitted).
12. Ulmer, M. W., Mattfeld, D. C., Hennig, M., & Goodson, J. C. (2015). A rollout algorithm for vehicle routing with stochastic customer requests. *Logistics management* (pp. 217–227)., Lecture notes in logistics Berlin: Springer.
13. Ulmer, M. W., Heilig, L., & Voß, S. (2017). On the value and challenge of real-time information in dynamic dispatching of service vehicles. Business & Information. *Systems Engineering*, *59*(3), 161–171.
14. Ulmer, M. W., Mattfeld, D. C., & Köster, F. (2017). Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science*.
15. Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Hennig, M. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science* (to appear).