https://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 58, No. 4, July–August 2024, pp. 726–740
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

An Iterative Sample Scenario Approach for the Dynamic Dispatch Waves Problem

Leon Lan,[a,*] Jasper M. H. van Doorn,[a] Niels A. Wouda,[b] Arpan Rijal,[b] Sandjai Bhulai[a]

[a] Department of Mathematics, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, Netherlands; [b] Department of Operations, University of Groningen, 9747 AE Groningen, Netherlands

*Corresponding author

Contact: l.lan@vu.nl, https://orcid.org/0000-0001-7479-0218 (LL); j.m.h.van.doorn@vu.nl, https://orcid.org/0009-0002-0233-367X (JMHvD); n.a.wouda@rug.nl, https://orcid.org/0000-0003-2463-0309 (NAW); a.rijal@rug.nl, https://orcid.org/0000-0001-8701-2757 (AR); s.bhulai@vu.nl, https://orcid.org/0000-0003-1124-8821 (SB)

Received: March 31, 2023
Revised: November 22, 2023
Accepted: February 20, 2024
Published Online in Articles in Advance: March 19, 2024

https://doi.org/10.1287/trsc.2023.0111

Abstract. A challenge in same-day delivery operations is that delivery requests are typically not known beforehand, but are instead revealed dynamically during the day. This uncertainty introduces a trade-off between dispatching vehicles to serve requests as soon as they are revealed to ensure timely delivery and delaying the dispatching decision to consolidate routing decisions with future, currently unknown requests. In this paper, we study the *dynamic dispatch waves* problem, a same-day delivery problem in which vehicles are dispatched at fixed decision moments. At each decision moment, the system operator must decide which of the known requests to dispatch and how to route these dispatched requests. The operator's goal is to minimize the total routing cost while ensuring that all requests are served on time. We propose *iterative conditional dispatch* (*ICD*), an iterative solution construction procedure based on a sample scenario approach. ICD iteratively solves sample scenarios to classify requests to be dispatched, postponed, or undecided. The set of undecided requests shrinks in each iteration until a final dispatching decision is made in the last iteration. We develop two variants of ICD: one variant based on thresholds, and another variant based on similarity. A significant strength of ICD is that it is conceptually simple and easy to implement. This simplicity does not harm performance: through rigorous numerical experiments, we show that both variants efficiently navigate the large state and action spaces of the dynamic dispatch waves problem and quickly converge to a high-quality solution. Finally, we demonstrate that the threshold-based ICD variant achieves excellent results on instances from the EURO Meets NeurIPS 2022 Vehicle Routing Competition, nearly matching the performance of the winning machine learning–based strategy.

History: This paper has been accepted for the *Transportation Science* Special Issue on DIMACS Implementation Challenge: Vehicle Routing Problems.

Funding: This work was supported by TKI Dinalog, Topsector Logistics, and the Dutch Ministry of Economic Affairs and Climate Policy.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2023.0111.

Keywords: same-day delivery • dynamic dispatch waves • vehicle routing • sample scenario approach • iterative conditional dispatch

1. Introduction

E-commerce retailers are increasingly offering same-day or even instant delivery services, with deliveries often occurring within two hours after order placement (Li, Archetti, and Ljubic 2023). The ability of retailers to offer fast and cost-effective deliveries heavily relies on making efficient dispatching and routing decisions, which involve balancing the trade-off between dispatching orders upon their arrival and waiting for future order arrivals (Klapp, Erera, and Toriello 2018b). Dispatching orders early ensures that customers receive their deliveries according to their preferred delivery time, whereas waiting allows for delivery consolidation and scale efficiency through improved routing decisions. However, as delivery requests are typically unknown beforehand, evaluating this trade-off requires taking into account the underlying uncertainty that arises from several factors, such as the variability of order arrival patterns, customer locations, the sizes of requests, and the time windows or deadlines of these requests. Accounting for all these sources of uncertainty in the same-day delivery environment, where decisions must be made quickly, poses considerable methodological challenges.

This paper considers the *dynamic dispatch waves problem* (*DDWP*). The DDWP is a same-day delivery problem

in which vehicles are dispatched at fixed decision moments (e.g., every hour). The DDWP is found in numerous applications, such as e-commerce (Klapp, Erera, and Toriello 2018a), urban consolidation centers (Van Heeswijk, Mes, and Schutten 2019), and grocery delivery services (Liu and Luo 2023). A set of new delivery requests arrives at each decision moment, and the system operator has to decide which requests to dispatch and how to route them, and the remaining requests are postponed to future decision moments. We consider a variant of the DDWP that was introduced in the EURO Meets NeurIPS 2022 Vehicle Routing Competition (Kool et al. 2023). In this variant, all delivery requests must be served within their requested time windows and a sufficient number of vehicles is available to serve all requests on time. The goal is to minimize the total distance traveled.

We propose the *iterative conditional dispatch* (*ICD*) algorithm to solve the DDWP quickly, within just a few minutes. The algorithm leverages ideas from progressive hedging (Rockafellar and Wets 1991) and sample scenario planning (Bent and Van Hentenryck 2004). The ICD algorithm constructs a dispatching solution through the iterative resolution of sample scenarios to classify requests as dispatched, postponed, or undecided. The set of undecided requests shrinks in each iteration until a final dispatching decision is made in the last iteration.

ICD can be seen as a generalization of the dynamic stochastic hedging heuristic (DSHH) introduced by Hvattum, Løkketangen, and Laporte (2006) for a dynamic pickup and delivery problem with fixed decision moments. The DSHH also uses similar concepts from progressive hedging to construct a solution iteratively, and the authors use a single-threshold-based mechanism that classifies which requests to serve next based on a sample scenario approach. These decisions are then enforced by restricting the customer time windows to serve the selected requests before the next decision moment. Although this adjustment is suitable for problems where vehicles are already en route and decisions concern selecting the next customers to visit, it falls short for problems involving dispatching decisions because it severely affects the quality of routes.

To overcome this issue, we use *dispatch windows* to enforce dispatching decisions, leaving the customer time window and overall problem structure intact. Moreover, we extend the single-threshold mechanism of the DSHH to a double-threshold function: besides classifying which requests to dispatch, we also classify which requests to postpone. Rigorous experiments on a large test bed of diverse instances demonstrate that the double-threshold method applied in ICD consistently outperforms the single-threshold variant by as much as 2%. To avoid tuning of the threshold parameter values, we also develop a nonparametric ICD variant that is as competitive as the single-threshold variant. Finally, we show that ICD performs competitively on instances from the EURO Meets NeurIPS 2022 Vehicle Routing Competition (Kool et al. 2023), nearly matching the performance of the winning machine learning–based approach by Baty et al. (2024) that ranked first in the competition. Although we develop and benchmark ICD for the special case of a DDWP in which all requests must be served, the idea of dispatch windows can easily be extended to more general variants of DDWPs with different objectives and constraints, such as maximizing the number of requests to be served or crowdsourced delivery.

The outline of this paper is as follows. Section 2 surveys the related literature. Section 3 formalizes the problem description. Section 4 presents the iterative conditional dispatch algorithm. Section 5 describes the numerical experiments and discusses the results. Section 6 concludes the paper and discusses future research ideas.

2. Literature Review

The DDWP can be seen as a particular variant of the same-day delivery problem (Voccia, Campbell, and Thomas 2019; Zhang and Van Woensel 2023). The same-day delivery problem is a stochastic dynamic vehicle routing problem (VRP) where not all customer requests are known beforehand, but at least some arrive dynamically over the planning horizon. The goal is to make dispatching decisions either when each customer request arrives or at a number of fixed moments in time (e.g., every hour). Because e-commerce companies are increasingly offering same-day delivery services to enhance customer satisfaction (Klapp, Erera, and Toriello 2020), the rapid growth of e-commerce has resulted in a burst of recent work on same-day delivery problems: Zhang and Van Woensel (2023) find that 90% of same-day delivery problem–related studies have been published within the last five years.

The most common way of modeling a stochastic and dynamic VRP is by formulating it as a Markov decision process (MDP). Because the state and action spaces of MDP formulations for such problems are typically very large, most studies resort to approximate algorithms. We refer to the survey of Soeffker, Ulmer, and Mattfeld (2022) for a classification of solution methods, and discuss here only two main ways of solving same-day delivery problems.

The first way is based on scenario sampling from a (known or approximated) stochastic process that models the dynamics of customer request arrivals. These sampled scenarios reduce to static VRP instances, which are considerably easier to solve. Solving each scenario in isolation, however, results in as many solutions as there are sampled scenarios. The main difficulty in

scenario sampling is how to combine the information from each scenario solution into a single, actionable decision for the original problem.

Bent and Van Hentenryck (2004) propose the use of a so-called *consensus function* to combine the scenario solutions into a single action. Their consensus function is based on a least-commitment approach (Stefik 1981). The results show a clear benefit over simply implementing the "best" scenario solution, suggesting a substantial benefit from finding consensus among the different scenarios. Hvattum, Løkketangen, and Laporte (2006) consider a dynamic pickup and delivery problem with fixed decision moments, and propose the DSHH to solve this problem. The DSHH algorithm iteratively solves a set of sample scenarios and computes the percentage of scenarios in which each request was served before the next decision moment. Hvattum, Løkketangen, and Laporte (2006) apply a threshold decision rule to these percentages to determine which request to serve. Ghiani, Manni, and Thomas (2012) consider a same-day delivery problem with a single vehicle and propose a consensus function that selects a solution with minimal pairwise Hamming distance from the set of scenario solutions. Voccia, Campbell, and Thomas (2019) consider a same-day delivery problem where dispatch decisions are made as each request arrives. Their consensus function identifies the solution that is most often observed in the scenario solutions. Azi, Gendreau, and Potvin (2012) study a same-day delivery problem with request acceptance and use a sample scenario approach to determine the relative benefit of accepting a request. Similarly, Dayarian and Savelsbergh (2020) use a sample scenario approach to determine the relative benefit of dispatching a request at the current dispatching epoch, or postponing the request for later (for a similar idea, see Hvattum, Løkketangen, and Laporte 2007).

Besides scenario-based approaches, there are many other ways to solve same-day delivery problems by approximating some aspect of the MDPs. Klapp, Erera, and Toriello (2018b) study a single-vehicle variant of the DDWP, in which requests are located on a line. They formulate a rollout policy and an approximate linear programming–based dynamic policy to solve the problem. Klapp, Erera, and Toriello (2018a) generalize this setting to general network topologies. Van Heeswijk, Mes, and Schutten (2019) formulate an MDP for a DDWP and solve the MDP using approximate dynamic programming with a value function approximation through basis functions. They test their method on instances with up to 40 customer requests arriving between decision moments and find that they outperform several myopic policies by 6%–20%. Ulmer, Thomas, and Mattfeld (2019) formulate an MDP for a same-day delivery problem in which vehicles are allowed to return to the depot before all assigned requests have been served. They propose a value function approximation and show that preemptive depot returns can substantially increase the number of deliveries. Ulmer (2020) integrates pricing and routing for same-day delivery in an e-commerce setting. They use a value function approximation based on state space aggregation to solve the problem efficiently. Liu and Luo (2023) present a dynamic programming formulation and structured approximation method for a DDWP. They prove several structural results on the quality of the resulting solution and validate the approach numerically through a real-world case in grocery and food delivery. Recently, Heitmann et al. (2023) combined the two approaches in a dial-a-ride setting. They used value function approximation to efficiently decide whether to accept a request, while using scenario sampling and the consensus function of Bent and Van Hentenryck (2004) to decide on the routing problem.

Scenario sampling and other MDP-based techniques both have to work around the issue that the underlying vehicle routing problem in same-day delivery problems is, in general, difficult to solve. This makes accurately estimating the (expected) cost of a dispatching decision a difficult task, because that typically requires solving many routing problems in a decision tree. This is not an option in the time-constrained setting of the DDWP we consider here. Several techniques have been developed to curtail the computational effort needed to explore the solution space. Dominance checks can be used to eliminate suboptimal nodes in the decision tree without sacrificing optimality (Goodson, Thomas, and Ohlmann 2016). Alternatively, Secomandi and Margot (2009) restrict the state space heuristically, and Zhang, Zhang, and Fan (2022) impose a priori rules that restrict the customer search space based on the current solution. In particular, they allow vehicles to visit subsequent customers only within a predefined radius. Hvattum, Løkketangen, and Laporte (2006) use principles from progressive hedging (Rockafellar and Wets 1991) to iteratively fix decisions for some requests. This progressively builds a solution of high quality, which can be much faster than exhaustively solving the scenario sampling or MDP problem.

3. Problem Description

In this section, we formulate the DDWP variant that was presented in the EURO Meets NeurIPS 2022 Vehicle Routing Competition (Kool et al. 2023). We introduce notation and provide a general problem outline in Section 3.1. Thereafter, we introduce the vehicle routing problem with time windows and dispatch windows (VRPTW-DW) in Section 3.2. Finally, in Section 3.3, we formulate the DDWP as an MDP.

3.1. General Outline

We consider a planning horizon of H time units, which is further divided into a set of decision epochs $\mathcal{T} = \{1, 2, \ldots, t_{\text{final}}\}$, where t_{final} represents the final epoch. We assume each epoch $t \in \mathcal{T}$ starts at time $T_t \geq 0$, with $T_t > T_{t-1}$ for $t > 1$.

A set of new delivery requests ω_t is revealed at the start of each decision epoch $t \in \mathcal{T}$. The set of requests ω_t has support Ω_t. We assume the support Ω_t is known, either through direct knowledge or by approximation from historical data. Each request $i \in \omega_t$ requires a service time of $\mu_i \geq 0$ time units when serviced at the request location. Service may only start in the request's time window $[e_i, l_i]$, with $T_t \leq e_i < l_i \leq H$. Early arrival at a request location is allowed, but service must wait until the time window opens. The demand of a request is denoted by $q_i \geq 0$ units of load capacity of vehicles. The release time of a request is denoted by $r_i = T_t$. The travel time between the locations of two requests i and j is denoted by $\tau_{ij} \geq 0$, and the cost of travel is denoted by $c_{ij} \geq 0$. Deliveries are made with a fleet of homogeneous vehicles $\mathcal{K}$, where each vehicle has a capacity of $Q > 0$ units. A feasible route dispatched in epoch t starts at the depot at time T_t and must return to the depot before the end of the planning horizon H. The sum of the requested demands on the route may not exceed the vehicle capacity.

We consider that operators always have access to sufficient vehicles at the start of each decision epoch. This assumption is different from the common restriction in same-day delivery problems that only a limited number of vehicles is available, while allowing for some requests not to be served (Klapp, Erera, and Toriello 2018b; Voccia, Campbell, and Thomas 2019). Despite these differences, our problem and solution approach capture the dynamics between waiting and dispatching decisions to improve operational efficiency, which is one of the core challenges of same-day delivery problems. To offer a more realistic view, we include an alternative model in Online Appendix C with limited fleet availability and accompanying numerical results.

At the start of each decision epoch, the operator must decide which of the known requests to dispatch and which ones to postpone to consolidate with new requests that may arrive in later epochs. The objective is to dispatch requests and route vehicles such that all requests are served within their time windows, and to minimize the total travel cost of the routes in current and future epochs. This is formalized in Section 3.3.

3.2. Static Vehicle Routing Problem with Dispatch Windows

Before the dynamic problem is formulated using MDP terminology, we present the VRPTW-DW as a preliminary problem. The VRPTW-DW is the static and deterministic version of the DDWP. Similar models have been presented before in Klapp, Erera, and Toriello (2018a) and Van Heeswijk, Mes, and Schutten (2019). Let $\mathcal{N}$ be the index set of delivery requests, and let $T \geq 0$ denote the earliest time that a request may be dispatched. The depot is represented by two indices 0 and $|\mathcal{N}| + 1$, which correspond to the start and end of a route, respectively. The depot has no service time ($\mu_0 = \mu_{|\mathcal{N}|+1} = 0$), no demand ($q_0 = q_{|\mathcal{N}|+1} = 0$), and a time window that spans the planning horizon ($[e_0, l_0] = [e_{|\mathcal{N}|+1}, l_{|\mathcal{N}|+1}] = [0, H]$). Let $\mathcal{V} = \{0, |\mathcal{N}| + 1\} \cup \mathcal{N}$ define the set of locations. A dispatch window $[r_i^-, r_i^+]$ is a time interval during which a route containing request $i \in \mathcal{N}$ must leave the depot. Unless specified otherwise, we assume $r_i^- = r_i$ and $r_i^+ = H$ for each request $i \in \mathcal{N}$. Two sets of decision variables are used to define a solution. First, $x_{ijk} \in \{0, 1\}$ tracks the routing decision: it is one if vehicle $k \in \mathcal{K}$ travels from location $i \in \mathcal{V}$ to $j \in \mathcal{V}$, and zero otherwise. Second, $\theta_{ik} \geq 0$ determines the time at which service starts at location $i \in \mathcal{V}$ by vehicle $k \in \mathcal{K}$.

Let

$$c(x, \theta) = \sum_{k \in \mathcal{K}} \sum_{i,j \in \mathcal{V}} c_{ij} x_{ijk}$$

denote the total (travel) cost of a decision pair (x, θ). A formulation for the VRPTW-DW is given by

$$\text{VRPTW-DW}(\mathcal{N}, T):$$

$$\min_{x,\theta} \quad c(x, \theta) \tag{1a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{ijk} = 1 \qquad \forall i \in \mathcal{N}, \tag{1b}$$

$$\sum_{j \in \mathcal{V}} x_{0jk} = 1 \qquad \forall k \in \mathcal{K}, \tag{1c}$$

$$\sum_{i \in \mathcal{V}} x_{i,|\mathcal{N}|+1,k} = 1 \qquad \forall k \in \mathcal{K}, \tag{1d}$$

$$\sum_{j \in \mathcal{N}} x_{ijk} = \sum_{j \in \mathcal{N}} x_{jik} \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tag{1e}$$

$$x_{ijk}(\theta_{ik} + \mu_i + \tau_{ij} - \theta_{jk}) \leq 0 \qquad \forall k \in \mathcal{K}, i, j \in \mathcal{V}, \tag{1f}$$

$$e_i \leq \theta_{ik} \leq l_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}, \tag{1g}$$

$$\max\{r_i^-, r_j^-\} x_{ijk} \leq \theta_{0k} \qquad \forall k \in \mathcal{K}, i, j \in \mathcal{N}, \tag{1h}$$

$$\theta_{0k} \leq \min\{r_i^+, r_j^+\} x_{ijk} \qquad \forall k \in \mathcal{K}, i, j \in \mathcal{N}, \tag{1i}$$

$$\theta_{0k} \geq T \qquad \forall k \in \mathcal{K} \tag{1j}$$

$$\sum_{i \in \mathcal{V}} q_i \sum_{j \in \mathcal{V}} x_{ijk} \leq Q \qquad \forall k \in \mathcal{K}, \tag{1k}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in \mathcal{K}, i,j \in \mathcal{V}, \quad (1l)$$

$$\theta_{ik} \geq 0 \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}. \quad (1m)$$

The objective (1a) minimizes the total travel cost. Constraints (1b) ensure that each request is assigned to exactly one vehicle. Constraints (1c) and (1d) define that each vehicle starts and ends at the depot, respectively. Constraints (1e) ensure flow conservation at each request location. Constraints (1f) and (1g) guarantee feasibility with respect to the arrival times and time windows, respectively. Constraints (1h) and (1i) ensure feasibility with respect to the request dispatch windows, and Constraints (1j) ensure that no vehicle is dispatched before time T. Finally, Constraints (1k) ensure feasibility with respect to vehicle capacity.

The problem formally described by (1) is relevant to the dynamic problem in three ways. First, and most importantly, it is repeatedly solved heuristically by our solution approach in Section 4 to determine dispatching actions in each epoch. Second, it is used to evaluate the cost of a dispatching decision at each decision moment (see Section 3.3.4). Third, with perfect information, the DDWP reduces to a static and deterministic VRPTW-DW problem. Let $\mathcal{N}_t = \cup_{t'=1}^{t} \omega_{t'}$ denote the set of all realized requests up to epoch t. Solving VRPTW-DW($\mathcal{N}_{t_{\text{final}}}, T_1$) now provides a lower bound on the cost of the dynamic counterpart for this realization. We refer to this large static problem as the *hindsight* problem and use it in Section 5 for benchmarking.

3.3. Markov Decision Process Formulation

We now formulate the DDWP as an MDP.

3.3.1. State Space. The state space specifies the current state of the system in reference to requests that are known but not yet dispatched. The state space at epoch t can be simply represented as a set $s_t = \{i \in \mathcal{N}_t | \text{request } i \text{ has not been dispatched}\}$.

3.3.2. Action Space. Given the current state s_t in epoch t, an action a_t is a subset of available requests that are chosen to be dispatched in epoch t, that is, $a_t \subseteq s_t$. Let $m_t \subseteq s_t$ denote the set of requests in epoch t that must be dispatched because of feasibility requirements. Specifically, we define

$$m_t = \{i \in s_t | \text{VRPTW-DW } (\{i\}, T_{t+1}) \text{ is infeasible}\}$$

as the set of requests that cannot be feasibly dispatched as a single request in the next epoch at time T_{t+1}. The complete action space $\mathcal{A}(s_t)$ is then specified as

$$\mathcal{A}(s_t) = \{a_t \subseteq s_t | m_t \subseteq a_t\}.$$

At its largest, the action space can be equal in size to $2^{|s_t|}$, the power set of the current state s_t.

Illustrative of the literature, Voccia, Campbell, and Thomas (2019) and Van Heeswijk, Mes, and Schutten (2019), among others, use more involved state space representations where route information such as return times is also specified. Also, Ulmer et al. (2020) argue for the use of a route-based representation of state space. We chose to simplify our representation for two reasons. First, unlike the aforementioned studies, our assumption of unlimited available vehicles ensures that the dispatching of vehicles in the current epoch does not impact decisions in later epochs. Therefore, including additional route-specific information would only serve to expand the state space. Second, the existence of multiple optimal routes with the same cost would increase the size of the action space. Our set representation avoids this symmetry issue and substantially reduces the size of the action space.

3.3.3. State Transitions. The transition to the next state s_{t+1} depends on the chosen action a_t and the unknown future realization ω_{t+1}. For each realization ω_{t+1} from the support Ω_{t+1}, the next epoch's state is given by $s_{t+1} = (s_t \setminus a_t) \cup \omega_{t+1}$.

3.3.4. Optimality Conditions. The direct cost $C(s_t, a_t)$ of a specific action a_t in state s_t is given by the cost of routing the requests in a_t at time T_t. Let (x^*, θ^*) be the optimal solution to VRPTW-DW(a_t, T_t). The direct cost is then given by $C(s_t, a_t) = c(x^*, \theta^*)$. The objective of the DDWP is to select, for each epoch $t \in \mathcal{T}$, a minimum cost action such that the following (Bellman) optimality conditions are satisfied:

$$V(s_t) = \begin{cases} C(s_t, s_t) & \text{if } t = |\mathcal{T}|, \\ \min\limits_{a \in \mathcal{A}(s_t)} [C(s_t, a) + \mathbb{E}_{\omega_{t+1}}[V(s_{t+1})]] & \text{otherwise.} \end{cases} \quad (2)$$

The conditions of (2) lend themselves naturally to a dynamic programming-based solution approach. However, there are three main difficulties in pursuing a solution approach in this direction. First, the number of actions $\mathcal{A}(s_t)$ is exponential in the size of the state s_t. Second, obtaining the costs $C(s_t, a_t)$ for a single state and action pair (s_t, a_t) requires solving the challenging VRPTW-DW of Section 3.2. Third, an accurate specification of the transition probabilities that are needed to evaluate the expected future costs is often not available in practice. In the next section, we propose an algorithm that is tailored to overcome these challenges.

4. The Iterative Conditional Dispatch Algorithm

This section presents the iterative conditional dispatch algorithm to solve the DDWP. The ICD algorithm is an iterative solution construction procedure based on a

sample scenario approach to tractably approximate the exponential state and action spaces outlined in Section 3.3. The pseudocode of the ICD algorithm is given in Algorithm 1 for a fixed epoch $t \in \mathcal{T}$. The algorithm maintains two sets of decisions: the set of dispatched requests d_t and the set of postponed requests p_t. At initialization, we initially set $d_t = m_t$ and $p_t = \emptyset$. Then, the ICD algorithm iteratively applies two steps: Step I and Step II. In Step I (Section 4.1), we first generate a set of sample scenarios. These sample scenarios each represent one potential realization of future requests. Then, with the previously fixed decisions given by d_t and p_t, we solve the sample scenarios as static VRPTW-DWs (Section 3.2) to identify the dispatch and postponement decisions that are made for each sample instance. In Step II (Section 4.2), we use the scenario solutions to classify requests that are not in $d_t \cup p_t$ into either the dispatch set d_t or the postponement set p_t, or leave the request undecided. Note that we maintain the invariant $d_t \cap p_t = \emptyset$, so a request cannot be classified as both dispatched and postponed. The classification of Step II reduces the number of undecided requests in subsequent iterations. Upon finishing a specified number of iterations or when all requests have been classified (i.e., when $s_t = d_t \cup p_t$), the dispatch action a_t is given by d_t. Sections 4.1 and 4.2 describe Steps I and II in more detail.

Algorithm 1 (Iterative Conditional Dispatch in Epoch $t \in \mathcal{T}$)

1: Initialize $d_t = m_t$ and $p_t = \emptyset$
2: **while** number of iterations not exceeded **and** $s_t \neq d_t \cup p_t$ **do**
3: **Step I**: generate scenarios $\mathcal{S}$ and solve them as VRPTW-DWs conditioned on d_t and p_t
4: **Step II**: update d_t and p_t based on scenario solutions
5: **end while**
6: $a_t = d_t$

4.1. Step I: Solving Sample Scenarios

Step I requires the generation of $|\mathcal{S}|$ sample scenarios and solving each of the resulting static routing problems. Formally, a single scenario S is constructed as follows. A scenario $S \in \mathcal{S}$ is a set of requests formed by the current epoch state s_t and a sample path realization $(\tilde{\omega}_{t+1}, \tilde{\omega}_{t+2}, \ldots, \tilde{\omega}_{t+L})$, where $\tilde{\omega}_{t'}$ sampled from $\Omega_{t'}$ denotes the set of future requests that arrive in decision epoch t' and L denotes the number of look-ahead epochs. Thus,

$$S = s_t \cup \bigcup_{t'=t+1}^{t+L} \tilde{\omega}_{t'}.$$

Each scenario is solved as the problem defined by VRPTW-DW(S, T_t). To incorporate the dispatch and postponement decisions d_t and p_t, we make the following adjustments to the dispatch windows $[r_i^-, r_i^+]$ for each request $i \in S$. For dispatched requests $i \in d_t$, the dispatch window is set to $[r_i^-, r_i^+] = [T_t, T_t]$, forcing these requests to be dispatched in the current epoch t at decision moment T_t. For postponed requests $i \in p_t$, the dispatch window is set to $[r_i^-, r_i^+] = [T_{t+1}, H]$, where T_{t+1} is the decision moment of the next epoch $t+1$. All other requests $i \in S \setminus (d_t \cup p_t)$ retain their default dispatch windows.

A feasible solution (x, θ) to VRPTW-DW(S, T_t) is readily mapped back to a *scenario solution* a_S. In particular, we can infer the request and vehicle pairings from the x decisions. Then, the epoch in which each request was dispatched can be read from the route starts θ_{0k} for each vehicle $k \in \mathcal{K}$. Formally, a request $i \in s_t$ is dispatched in epoch t if $\theta_{0k_i} = T_t$, where k_i is the vehicle to which i is assigned. We thus have

$$a_S = \{i \in s_t \mid \theta_{0k_i} = T_t\}.$$

One of the challenges in sample scenario approaches is that the scenario problems are often hard to solve optimally. We show in the numerical experiments (Section 5) that it is not necessary to solve these problems optimally to obtain good solutions using ICD.

4.2. Step II: Updating d_t and p_t Using Consensus Functions

Step II involves classifying requests based on the scenario solutions to update the action sets d_t and p_t. We use consensus functions to make the classification. Our first consensus function is based on thresholds, and our second on the Hamming distance.

4.2.1. Threshold-Based Consensus.

The threshold-based consensus function classifies each request based on how often it is dispatched in the scenario solutions. First, we define the dispatch score $0 \leq \phi(i) \leq 1$ of a known request $i \in s_t$ as the relative frequency of it being dispatched among the sample scenario solutions. Formally, the dispatch score $\phi(i)$ of a request $i \in s_t$ is given by

$$\phi(i) = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \mathbb{1}_{i \in a_S},$$

where $\mathbb{1}_{i \in a_S}$ is an indicator that is one if i is dispatched by scenario solution a_S and zero otherwise.

The threshold-based consensus function is characterized by two separate threshold parameters, ϵ_D and ϵ_P, that determine the cutoff points for the assignment of requests to action sets d_t and p_t. Given the dispatch scores $\phi(i)$, we update d_t and p_t as follows:

$$d_t = d_t \cup \{n \in s_t \mid \phi(i) \geq \epsilon_D\},$$

$$p_t = p_t \cup \{n \in s_t \mid \phi(i) < \epsilon_P\}.$$

The dispatch score $\phi(i)$ can be interpreted as a measure of the probability that a request i should be dispatched

in epoch t. If the dispatch score of a request is high, then the request is a good candidate for dispatching. Conversely, requests with high postponement probabilities $(1-\phi(i))$ should be postponed. The threshold parameters ϵ_D and ϵ_P reflect the tolerance levels for classifying a request as dispatched and postponed, respectively. Requests with a dispatch score in $[\epsilon_P, \epsilon_D)$ remain unclassified in the current iteration, as their dispatch probabilities are not high enough to be classified as dispatched, and not low enough to be classified as postponed. In later iterations, the dispatch scores of the unclassified requests are expected to change because of the updated action sets d_t and p_t.

Note that the disjoint property of d_t and p_t requires $\epsilon_D \geq \epsilon_P$, because otherwise a request can be marked for dispatch and postponement simultaneously, which violates the feasibility of the action. Also observe that when $\epsilon_D + \epsilon_P = 1$, the ICD reduces to a noniterative procedure, as all requests are marked in one iteration.

4.2.2. Similarity-Based Consensus. Determining the optimal threshold values for the threshold-based consensus function requires substantial parameter tuning. Therefore, we develop an additional parameter-free consensus function that selects a scenario solution with the lowest average Hamming distance to the other scenario solutions. The Hamming distance counts the number of requests that two solutions decide on differently in their dispatching actions. We update the partial dispatch set by selecting the scenario S^* whose solution a_{S^*} obtains the minimum average Hamming distance:

$$S^* = \arg\min_{S \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \sum_{S' \in \mathcal{S}} |a_S \setminus a_{S'}| + |a_{S'} \setminus a_S|,$$

$$d_t = d_t \cup a_{S^*}.$$

The intuition behind this particular update is as follows. When the solution that is most similar to all other scenario solutions dispatches a request, it is likely that most other solutions also made this decision, and dispatching is a good idea. This solution may also include requests that are not as frequently dispatched, as this may have been a good choice to balance out the direct cost and the future costs for the specific scenario. Thus, when fixing a postponement decision, we are more conservative: only when *all* scenario solutions postpone the request do we classify it as postponed. The partial postponement set is hence updated as

$$p_t = p_t \cup \{i \in s_t | i \notin a_S \text{ for all } S \in \mathcal{S}\}.$$

5. Numerical Experiments

This section presents the computational experiments that show the effectiveness of our ICD algorithm. All numerical experiments are conducted on a single thread of an AMD EPYC 7H12 central processing unit (CPU). Our code is implemented in Python and is released at https://github.com/leonlan/dynamic-dispatch-waves under the MIT license. The static VRPTW-DW problems are solved using PyVRP, version 0.6.3, a high-performance and open-source heuristic VRP solver implemented in C++ and Python (Wouda, Lan, and Kool 2024). It builds upon the hybrid genetic search algorithm of Vidal et al. (2013) and Vidal (2022). Specifically for this paper, we extend PyVRP with support for dispatch windows to solve the VRPTW-DW of Section 3.2. Implementation details of PyVRP and its dispatch window extension are given in Online Appendix B.

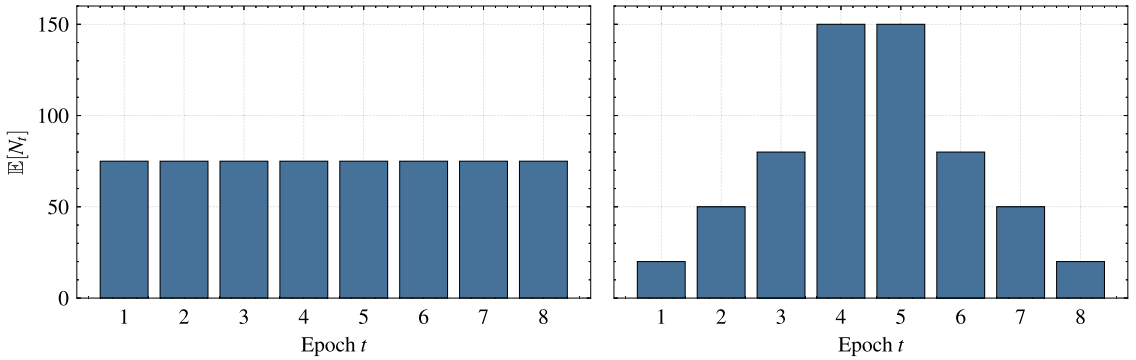
The rest of this section is structured as follows. We describe test instances in Section 5.1. Sections 5.2 and 5.3 describe the benchmark algorithms and their parameter settings, respectively. Section 5.4 presents the numerical results for our test instances and investigates the individual components of ICD. Finally, in Section 5.5, we evaluate ICD on instances from the EURO Meets NeurIPS 2022 Vehicle Routing Competition.

5.1. Benchmark Instances

Benchmark instances for the DDWP are based on the topologies of the extended Solomon instances (Homberger and Gehring 1999). We replace the existing time windows of these instances with a new time window generation process to better control the stochastic dynamics of the benchmark instances. We consider all of the three location dispersion sets of the Solomon instances: randomly distributed (R), clustered (C), and randomly distributed and clustered (RC). Two large static instances with 1,000 customers from each of these three categories are considered. Specifically, we select R1_10_1 and R2_10_1 for R, C1_10_1 and C2_10_1 for C, and RC1_10_1 and RC2_10_1 for RC. The Euclidean distance between each combination of two locations in these static instances is normalized such that the furthest location can be served within at most one epoch duration. Moreover, we assume that the travel cost and duration between locations are equal to the travel distance.

We now explain how the dynamic instances for the DDWP are created. The planning horizon of $H = 8$ hours is discretized into eight epochs, $\mathcal{T} = \{1, \ldots, 8\}$, of equal duration (one hour each). At each epoch t, we sample N_t requests and their attributes. To adequately capture the dynamics of different arrival processes, two types of arrival processes $\{N_t\}_{t \in \mathcal{T}}$ are considered. We call these two arrival processes *homogeneous* and *unimodal*, respectively, and illustrate them graphically in Figure 1. The number of arrivals in both arrival processes follows a discrete uniform distribution with bounded support on $[\lfloor 0.9\mathbb{E}[N_t] \rfloor, \lfloor 1.1\mathbb{E}[N_t] \rfloor]$. For the homogeneous arrival process, the expected number of arrivals remains constant at $\mathbb{E}[N_t] = 75$ requests for every epoch t in $\mathcal{T}$. The

Figure 1. (Color online) The Expected Number of Arrivals per Epoch for the (Left) Homogeneous and (Right) Unimodal Arrival Processes



unimodal process has epoch-dependent arrival rates with a peak in the middle of the planning horizon. The expected arrivals in the eight epochs are (20, 50, 80, 150, 150, 80, 50, 20) delivery requests, where the tth element in the sequence denotes $\mathbb{E}[N_t]$.

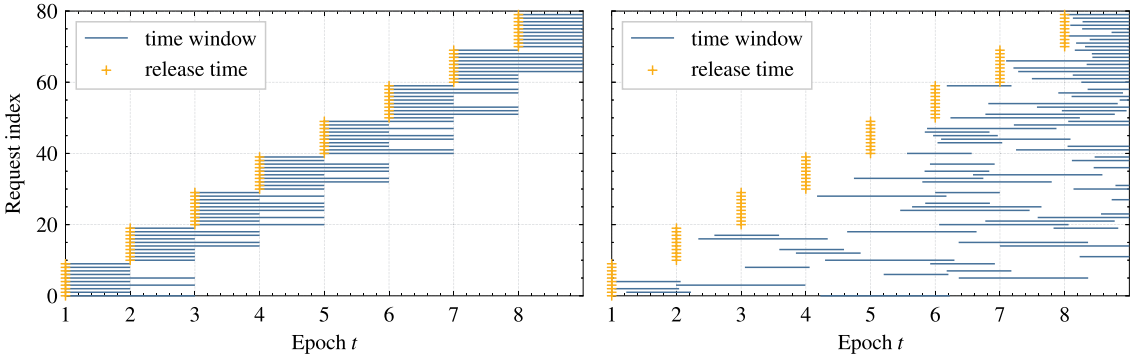
Request attributes corresponding to location, demand, and service time are sampled uniformly and independently from the static instance. For the time windows, we examine six variants, which are characterized by *width* and *type*. We consider three different maximum time window widths: two hours, four hours, and eight hours. For every request, the width of the time window is sampled from a discrete uniform distribution between one and the maximum time window width. Let W_i denote the width of the time window for request i. We examine two types of time windows: deadlines and regular time windows. Deadlines are defined such that the time window starts at the request's release time r_i and ends at $r_i + W_i$. Regular time windows begin at a time point uniformly sampled between the request's release time r_i and the planning horizon H. They end at this starting time plus W_i. Finally, all time windows are truncated to end no later than the planning horizon. We label the resulting variants as DL2, DL4, DL8, TW2, TW4, and TW8, where *DL* represents deadlines, *TW* represents regular time windows, and the digit indicates the maximum time window width in hours. The time window variants DL2 and TW2 are visualized in Figure 2.

In total, we have three topologies, two arrival processes, and six time window variants. We use a full factorial design of these different parameters and consider all 36 combinations of these parameters.

5.2. Benchmark Algorithms

We implement several variants of the ICD algorithm. The first variant uses the double-threshold consensus function of Section 4.2.1, which we refer to as *ICD-double*. The second variant uses the Hamming distance consensus function of Section 4.2.2, and we refer to this variant as *ICD-Hamming*. We also consider two variants that use only a single threshold, which allows us to study the marginal contributions of both dispatch and postponement thresholds. The ICD variant with only a dispatch threshold closely resembles the DSHH of Hvattum, Løkketangen, and Laporte (2006), and we similarly refer to this variant as *DSHH*. The other single-threshold variant uses only a postponement threshold, and we refer to this variant as *ICD-postpone*. Because the ICD-postpone variant does not explicitly classify dispatch requests, the final action a_t is determined by the requests that are not

Figure 2. (Color online) Two Dynamic Instances with 10 Requests per Epoch and a Maximum Time Window Width of Two



Note. The left panel shows an instance with deadlines, and the right panel shows an instance with regular time windows.

postponed, that is, $a_t = s_t \setminus p_t$. Finally, we also implement a rolling horizon (RH) policy for comparison. The RH policy samples only a single scenario and uses the solution to this instance to determine which requests to dispatch. Note that the RH policy can be interpreted as a variant of ICD-Hamming using just a single sample scenario and one iteration. All algorithms are compared against the cost of the solution obtained from heuristically solving the hindsight problem using PyVRP (see Section 3.2).

5.3. Parameter Settings

All of the ICD variants employ three iterations, 30 scenarios per iteration, and one look-ahead epoch. The ICD parameters were selected based on manual testing and observations from the related literature (Hvattum, Løkketangen, and Laporte 2006; Voccia, Campbell, and Thomas 2019). To find the best threshold parameters, we evaluated the threshold-based ICD variants on 36 dynamic instances featuring homogeneous arrival processes, and we generated five different request realizations for each. Four separate values for both the dispatch threshold (0.4, 0.5, 0.6, 0.7) and the postponement threshold (0.2, 0.3, 0.4, 0.5) were tested. ICD-double was evaluated on a full factorial design of these threshold values, whereas DSHH and ICD-postpone were evaluated using only the dispatch and postponement values, respectively. The best results for ICD-double were achieved with dispatch and postponement thresholds of $\epsilon_D = 0.5$ and $\epsilon_P = 0.2$, respectively. DSHH obtained the best results with a dispatch threshold of $\epsilon_D = 0.5$, and ICD-postpone obtained the best results with a postponement threshold of $\epsilon_P = 0.3$. For the RH policy, we use one look-ahead epoch.

The total solving time in a single epoch is set to just 120 seconds. We keep this time limited to ensure our methods can be applied in practical same-day delivery settings, where dispatching decisions also need to be made quickly. In each epoch, the ICD methods solve 90 scenarios in total, to which we assign a time budget of 90 seconds, meaning that ICD solves each scenario using a time limit of only one second. Then, an additional 30 seconds is spent on computing a route plan and associated cost for the selected dispatching action. Although our implementation does not leverage parallelization, it is important to note that the scenario-solving process could be easily parallelized to substantially reduce the total solving time. The RH policy uses the full 90 seconds to solve one scenario and 30 seconds to compute the route plan. The hindsight solution is obtained by solving the hindsight problem for 600 seconds.

An important element of the ICD framework is to solve the static scenarios as efficiently as possible. We refer the reader to Online Appendix B.3 for the parameter tuning procedure of the static solver to solve scenarios well.

5.4. Results on the Benchmark Instances

In this section, we evaluate performance on the 36 instance classes described in Section 5.1. For each instance class, we generate 200 different dynamic instances (100 per selected static instance in each of the R, C, and RC sets). This results in a total of 7,200 instances. Each method is run on all 7,200 instances, and we record the final cost of each such run. Table 1 presents an overview of the results. It presents the average percentage gap to the cost of the hindsight solution for each method and instance class. Entries in bold indicate the best average gap for each instance class (row). Furthermore, an asterisk next to the best result indicates that the superior performance of the method is significant at the 0.05 level compared with all other methods in pairwise t-tests (with Bonferroni correction). A complete table of the statistical test results can be found in Table A.1 in Online Appendix A.

Table 1 shows that among all instance classes, ICD-double solutions have an average gap of 8.05% compared with the hindsight solutions. On average, ICD-double has a lower gap by 1.5%, 0.7%, 1.6%, and 0.8% compared with RH, DSHH, ICD-postpone, and ICD-Hamming, respectively. When we consider results by instance class (an individual row in Table 1), ICD-Hamming and ICD-double are better on average in 33 out of 36 instance classes, and ICD-double's improvement is statistically significant for 14 out of 36 instance classes. Deadline instances are arguably easier and allow for less significant savings by correctly anticipating future orders, as evidenced by the lower gaps of all methods on these instances compared with the hindsight solutions.

Several observations emerge from the benchmark results. First, under a single-threshold consensus function (DSHH or ICD-postpone), it is generally better to operate with a dispatch threshold rather than a postponement threshold. We believe that this is because the postponement threshold depends only on sampled future request realizations. In contrast, dispatch thresholds also take into account immediate costs that are based on already-known requests across all scenarios. Therefore, dispatch thresholds make for a more reliable consensus function with less noise. Second, there is value in dispatching and postponing simultaneously: there is a notable difference in performance between iterative methods with one-sided decisions (DSHH and ICD-postpone) and iterative methods with two-sided decisions (ICD-Hamming and ICD-double). When we reduce the total expected number of requests from 600 to 450 and further to 300 (Table 2), these differences persist. This shows the benefits of simultaneous dispatching and postponement decisions, even when the

Table 1. Overview of Results

Instance			Methods				
Topology	Arrival	TW	RH	DSHH	ICD-postpone	ICD-Hamming	ICD-double
R	HOM	DL2	4.02	2.66	3.07	**2.46**	2.57
		DL4	8.66	6.84	8.00	6.38	**6.35**
		DL8	11.74	10.20	12.35	9.86	**9.73**
		TW2	13.38	12.40	13.47	12.64	**11.02***
		TW4	13.84	13.07	14.29	13.49	**11.76***
		TW8	13.52	13.17	14.10	13.54	**11.66***
	UNI	DL2	4.74	2.73	3.01	2.78	**2.69**
		DL4	10.20	8.03	8.87	7.79	**7.56**
		DL8	13.55	12.13	13.17	12.19	**11.41***
		TW2	14.15	15.12	14.91	15.39	**13.40***
		TW4	14.77	15.86	15.91	16.48	**14.46**
		TW8	14.63	16.24	16.20	16.90	**14.56**
C	HOM	DL2	1.80	**0.12**	0.16	0.15	0.12
		DL4	3.00	1.41	1.91	**1.29**	1.38
		DL8	4.28	3.12	4.45	**2.88**	3.08
		TW2	7.68	6.39	7.99	6.48	**5.99***
		TW4	7.29	6.38	8.11	6.39	**5.70***
		TW8	6.67	5.92	7.38	5.97	**5.29***
	UNI	DL2	2.20	0.12	0.16	0.17	**0.12**
		DL4	4.65	2.63	3.16	2.63	**2.55**
		DL8	6.77	5.10	6.39	5.12	**5.02**
		TW2	9.95	9.91	11.15	10.25	**9.21***
		TW4	9.98	10.14	11.14	10.75	**9.53***
		TW8	9.63	10.15	10.64	10.77	**9.26**
RC	HOM	DL2	3.33	2.29	2.76	**2.08**	2.23
		DL4	7.79	6.33	7.87	**5.94**	6.19
		DL8	10.83	9.76	12.27	9.39	**9.36**
		TW2	12.84	11.95	13.09	12.09	**10.67***
		TW4	13.57	12.95	14.48	12.92	**11.65***
		TW8	13.07	13.00	14.42	13.04	**11.59***
	UNI	DL2	4.41	2.49	2.75	2.37	**2.34**
		DL4	10.47	7.45	8.63	7.43	**7.28**
		DL8	14.37	11.60	13.34	11.59	**11.25**
		TW2	14.21	15.24	15.33	15.45	**13.61***
		TW4	**14.59**	16.19	16.38	16.30	14.67
		TW8	**14.37**	16.10	16.38	16.54	14.72
Average			9.58	8.76	9.66	8.83	**8.05**

Notes. The values represent the average percentage gaps with respect to the hindsight solutions over 200 different instances with 600 expected total requests. The best average gap in each row is marked in bold. An asterisk indicates that the method is statistically significantly better than the other methods for that instance at the 0.05 level (with Bonferroni correction). HOM, Homogeneous; UNI, unimodal.

scenario instances are considerably smaller. For smaller instances with 300 expected arrivals, the nonparametric ICD-Hamming solution obtains very good results. As the instances increase in size, however, the single-scenario solution that ICD-Hamming uses is outperformed by the threshold-based ICD-double algorithm, which uses information from all scenario solutions. The extended results for 300 and 450 total expected requests can be found in Online Appendix A. Third, the ICD algorithms (except ICD-postpone) generally perform better than RH, showing that there is value in considering multiple scenarios and iterations. Among instance classes with unimodal arrival processes and time windows, the noniterative RH policy obtains good results. In such instances, many requests can be postponed until the final epochs. This may lead to very large scenario instances, which, under tight time limits, may favor a single, optimized routing solution. Nevertheless, ICD-double still achieves better results than RH in the majority of the instance classes (34 out of 36).

Next, we conduct further experiments to shed light on the inner workings of ICD and highlight how important parameter settings affect the algorithm's performance. We restrict these additional experiments to the benchmark instances with unimodal arrival processes and time windows. All parameters are as in Section 5.3 unless mentioned otherwise.

5.4.1. Solving Time per Scenario. Varying the time allocated to solving sample scenarios, we evaluate the impact of ICD on the solution of sample scenarios and the impact on the overall solution of the complete

Table 2. The Average Percentage Gap to the Hindsight Solution for Different Total Expected Numbers of Requests

	RH	DSHH	ICD-postpone	ICD-Hamming	ICD-double
300 arrivals	12.58	10.57	11.84	**9.93**	10.03
450 arrivals	11.12	9.61	10.76	9.36	**8.96**
600 arrivals	9.58	8.76	9.66	8.83	**8.05**

dynamic problem. We vary the amount of time spent solving a single scenario among 0.25, 0.5, 0.75, 1, 2, and 4 seconds (recall that the default is 1 second). The time to compute the direct cost remains fixed at 30 seconds for all considered scenario time limits. We compare all results to the hindsight solutions obtained after 600 seconds. The results are presented in Figure 3, which shows the average gap with respect to the hindsight solution versus the solving time for a single scenario. As the sample scenarios are solved with more time, the resulting solutions provide a better trade-off in minimizing the immediate and future costs. When increasing the solving time from 0.25 seconds to 1 second, the average percentage gap decreases by 2.5%–4% across all methods. Increasing the solving time even further continues to improve the solutions, but at diminishing returns. These observations are consistent with the convergence curve of our static solver on general VRP instances, showing that the improvement from obtaining better static solutions translates to its dynamic counterpart.

5.4.2. Number of Iterations. In the next experiment, we alter the number of iterations while keeping all other ICD and consensus function parameters constant. We test the number of iterations using the values 1, 2, 3, 5, 10, and 15. Figure 4 illustrates the average gap with respect to the hindsight solution versus the number of iterations. For the threshold-based consensus functions, the optimal number of iterations is found to be three. This also supports the choices made in the initial parameter value selection of Section 5.3. For ICD-Hamming, the ideal number of iterations in this experiment is one. We note that on the complete set of instances, as presented in Table 1, the overall performance of ICD-Hamming with three iterations was better than with one iteration, because of its superior performance on the deadline instances. Remarkably, a single iteration for the ICD-Hamming method performs substantially better than the other methods using only one iteration. This can be in part explained by the fact that the Hamming distance consensus function selects a complete scenario solution for which the immediate reward is known. The threshold-based methods select requests that do not necessarily form a coherent route plan, as the requests are selected based on their relative dispatching frequency, not how often they are dispatched together. The dispatching action

Figure 3. (Color online) Average Percentage Gaps to the Hindsight Solution vs. the Solving Time in Seconds per Scenario

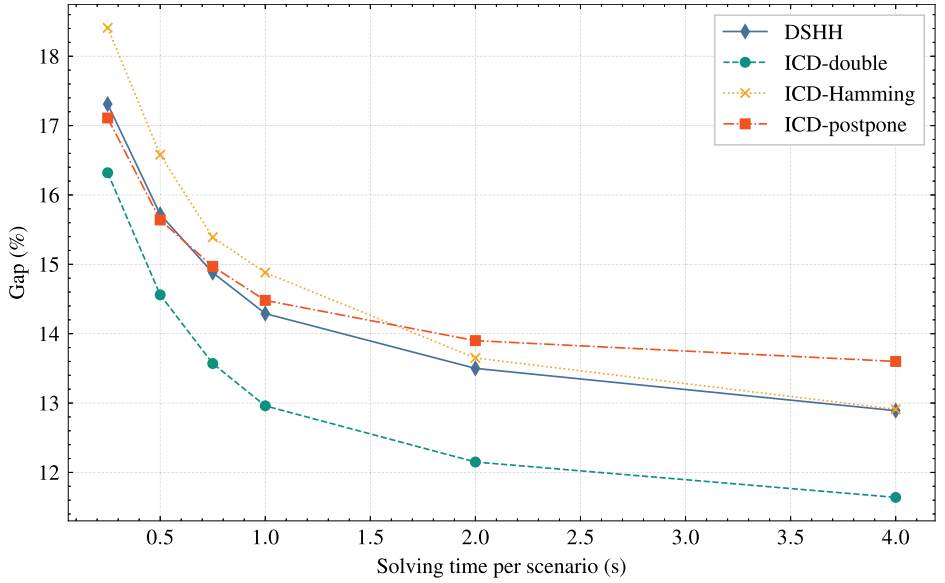
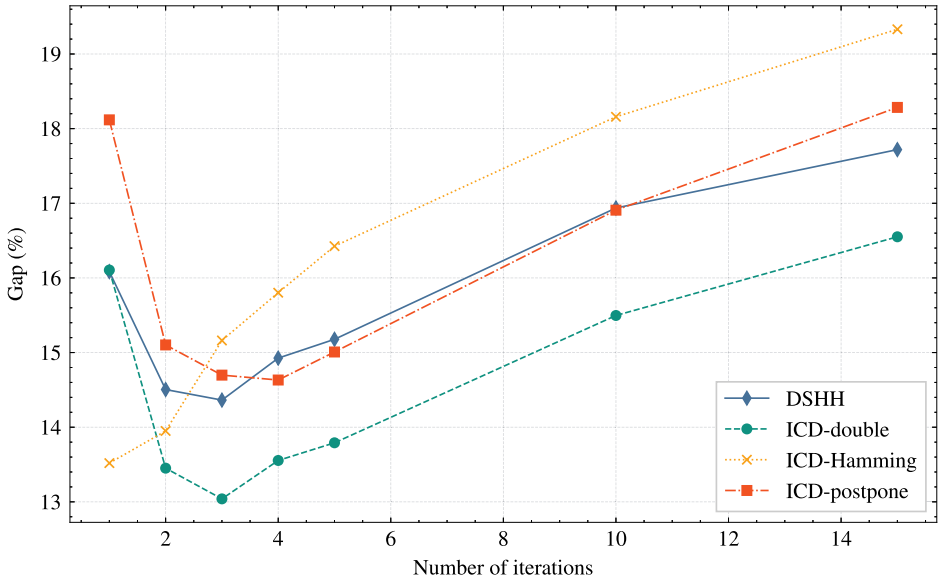


Figure 4. (Color online) Average Percentage Gaps to the Hindsight Solution vs. the Number of Iterations in the ICD Method



after one iteration thus does not necessarily result in efficient route plans. Instead, the threshold-based methods refine this dispatching action in subsequent iterations, given that part of the dispatching action is already determined. As the number of iterations increases beyond three, the trade-off between refining the dispatching action and solving scenarios becomes unfavorable. For example, using 10 iterations with the selected parameters results in solving 300 scenarios instead of 90 scenarios within the same time budget. This aligns with the earlier observation that allocating too little time to solving scenarios leads to worse overall performance.

5.5. Results on the EURO Meets NeurIPS 2022 Vehicle Routing Competition Instances

This final section evaluates the ICD-double algorithm on instances from the EURO Meets NeurIPS 2022 Vehicle Routing Competition (Kool et al. 2023). In the following, we abbreviate the competition name as EURO–NeurIPS.

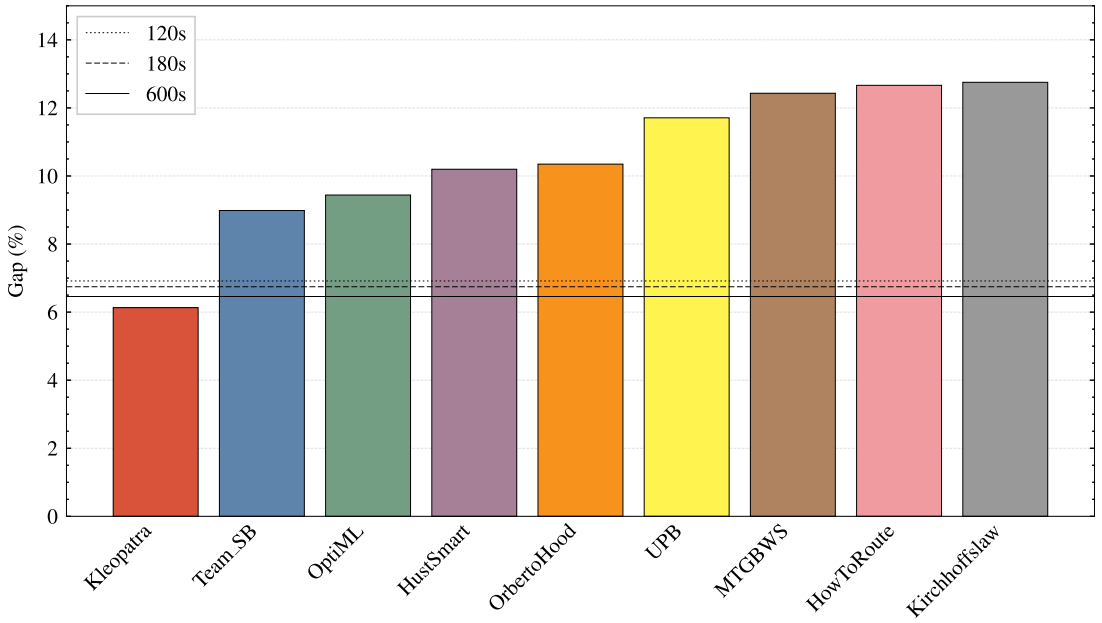
The EURO–NeurIPS competition presented two VRP variants that needed to be solved: a static VRPTW and a dynamic version thereof. We focus on the dynamic problem variant, which is also precisely what we formulated as the DDWP in this work. The main differences between the instances used in the competition and our work are the characteristics of the static instances and how the requests are sampled. In the competition, an instance of the DDWP is created by sampling at each decision epoch 100 requests from a static VRPTW instance. However, sampled requests that cannot be served on time are disregarded, meaning that the number of requests to be served is often much lower, especially in later epochs. Depending on the static instance, the total number of epochs ranges between five and nine, leading to a total number of requests ranging between 350 and 600.

After a four-month qualification phase, the 10 best algorithms were evaluated on 100 final hidden instances with a time limit of 120 seconds per epoch. We evaluate ICD-double on these 100 instances for the same time limit of 120 seconds, as well as longer time limits of 180 and 600 seconds, and compare with results from the competition. In the competition, the algorithms were evaluated on a single core of an Intel Xeon Gold 5118 processor with a CPU PassMark score of 1,808, which is slightly lower than the CPU PassMark score of 2,014 of the CPU that we used. To provide a fair comparison, we compensate for the CPU differences by reducing our time limits by a factor of $1,808/2,014$. Moreover, we did not change our parameters or implementation from the ones used in the benchmark instances in previous sections.

Figure 5 shows the average percentage gap to the hindsight solution for each final team, along with three lines of the ICD-double algorithm for a given epoch time limit. The ICD-double algorithm with the 120-second time limit would have ranked second and has a 0.8% gap over the best result. With a slightly longer time limit of 180 seconds, ICD-double is only 0.6% away, and increasing the time limit even further to 600 seconds decreases this gap to 0.3%. These results align with our earlier observations (see Figure 3), where increasing the time to solve a single scenario improves overall performance.

We briefly summarize the contributions of the top six teams. It is important to note that all teams were provided with a state-of-the-art VRPTW solver by Kool

Figure 5. (Color online) Results for the Final 100 Instances of the EURO–NeurIPS Competition



Note. Each line indicates the average percentage gap to the hindsight solution obtained by ICD-double for a given time limit (120, 180, or 600 seconds).

et al. (2022). PyVRP is built on top of the same static solver with marginal performance differences, meaning that differences between our ICD-double and other methods are mainly because of differences in the dynamic aspect of the algorithm. Team Kleopatra (Baty et al. 2024) proposed a machine learning pipeline to predict a prize associated with each request, quantifying how favorable it is to dispatch a request in the current epoch. They trained their model on the solutions of hindsight instances and solved a prize-collecting variant of the VRPTW in each decision epoch using the predicted prizes. Team_SB (Kwon et al. 2022) also proposed a machine learning–based approach. They trained two neural networks, one that predicted priorities for requests and one that selected requests that must be dispatched. With the predicted priority values, they solved a VRPTW with a modified cost function from which they determined which requests to dispatch. We, the author team, participated under the team name OptiML (van Doorn et al. 2022), and our submitted approach was a variant of the ICD-postpone algorithm with parameters tuned for the EURO–NeurIPS instances that were known to us during the qualification phase of the competition. HustSmart (Li et al. 2022) heuristically defined penalties for each request based on the static instance data. They then solved a variant of the VRP with penalties and dropping visits, which can be seen as a variant of the prize-collecting VRP. OrbertoHood (Hildebrandt et al. 2022) was the only other finalist who also used an online sampling-based approach. Their approach samples a set of scenarios and then solves a sample average approximation problem to estimate the marginal costs of dispatching or postponing each request. They solved the problem using column generation and used the resulting solution to determine which requests to dispatch. Finally, UPB (Dieter 2023) proposed a regret policy that analytically computes a regret value for each optional request. The regret value is the potential value that would be missed if a request were dispatched in the current epoch, and their algorithm postpones all requests with a regret value above a certain threshold.

6. Conclusions

This paper studies a variant of the dynamic dispatch waves problem in which the goal is to minimize the total routing costs while serving all incoming delivery requests on time. We propose the iterative conditional dispatch algorithm, which iteratively solves sample scenarios to classify requests to be dispatched or postponed, or leaves this decision to be determined for the next iterations. The most competitive variant of the algorithm uses a double-threshold consensus function, which classifies requests based on their dispatching frequency in the resolved sample scenarios. We additionally develop a parameter-free consensus function based on the Hamming distance. Numerical experiments on a wide variety of instances demonstrate that ICD with the Hamming distance consensus function generates competitive solutions compared with the dynamic stochastic hedging heuristic of Hvattum, Løkketangen, and

Laporte (2006), whereas ICD with a double threshold improves over both methods by as much as 2% on average for some instance classes. Additional experiments show that spending sufficient computational time to solve scenario instances is important to obtain overall good performance, but that there are clear diminishing returns. We find that the threshold-based variants of ICD obtain their best performance with three iterations, and increasing the number of iterations typically results in inferior solutions because of the reduced time available for solving scenarios. Finally, we show using instances from the EURO Meets NeurIPS 2022 Vehicle Routing Competition that our ICD method with the double threshold achieves excellent results.

The performance and simplicity of the ICD algorithm show a promising avenue for extension to other dynamic vehicle routing problem variants. In future work, we intend to apply ICD to other problems, including routing problems such as the same-day delivery problem with continuous arrivals (Voccia, Campbell, and Thomas 2019) and dispatching problems with ad hoc fleet arrivals (Arslan et al. 2019).

References

Arslan AM, Agatz N, Kroon L, Zuidwijk R (2019) Crowdsourced delivery—A dynamic pickup and delivery problem with ad hoc drivers. *Transportation Sci.* 53(1):222–235.

Azi N, Gendreau M, Potvin JY (2012) A dynamic vehicle routing problem with multiple delivery routes. *Ann. Oper. Res.* 199(1):103–112.

Baty L, Jungel K, Klein PS, Parmentier A, Schiffer M (2024) Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transportation Sci.*, ePub ahead of print February 14, https://doi.org/10.1287/trsc.2023.0107.

Bent RW, Van Hentenryck P (2004) Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Oper. Res.* 52(6):977–987.

Dayarian I, Savelsbergh M (2020) Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Production Oper. Management* 29(9):2153–2174.

Dieter P (2023) A regret policy for the dynamic vehicle routing problem with time windows. Daduna JR, Liedtke G, Shi X, Voß S, eds. *Computational Logistics.* Lecture Notes in Computer Science, vol. 14239 (Springer Nature, Cham, Switzerland), 235–247.

Ghiani G, Manni E, Thomas BW (2012) A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Sci.* 46(3):374–387.

Goodson JC, Thomas BW, Ohlmann JW (2016) Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Sci.* 50(2):591–607.

Heitmann RJO, Soeffker N, Ulmer MW, Mattfeld DC (2023) Combining value function approximation and multiple scenario approach for the effective management of ride-hailing services. *EURO J. Transportation Logist.* 12:100104.

Hildebrandt FD, Roberti R, Thomas BW, Ulmer MW (2022) A two-stage set partitioning approach for the EURO Meets NeurIPS 2022 Vehicle Routing Competition. Accessed September 29, 2023, https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/ORberto_Hood_and_the_Barrymen.pdf.

Homberger J, Gehring H (1999) Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Inform. Systems Oper. Res.* 37(3):297–318.

Hvattum LM, Løkketangen A, Laporte G (2006) Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Sci.* 40(4):421–438.

Hvattum LM, Løkketangen A, Laporte G (2007) A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks* 49(4):330–340.

Klapp MA, Erera AL, Toriello A (2018a) The dynamic dispatch waves problem for same-day delivery. *Eur. J. Oper. Res.* 271(2):519–534.

Klapp MA, Erera AL, Toriello A (2018b) The one-dimensional dynamic dispatch waves problem. *Transportation Sci.* 52(2):402–415.

Klapp MA, Erera AL, Toriello A (2020) Request acceptance in same-day delivery. *Transportation Res. Part E Logist. Transportation Rev.* 143:102083.

Kool W, Juninck JO, Roos E, Cornelissen K, Agterberg P, van Hoorn J, Visser T (2022) Hybrid genetic search for the vehicle routing problem with time windows: A high-performance implementation. *12th DIMACS Implementation Challenge Workshop* (DIMACS, Rutgers University, Piscataway, NJ).

Kool W, Bliek L, Numeroso D, Zhang Y, Catshoek T, Tierney K, Vidal T, Gromicho J (2023) The EURO Meets NeurIPS 2022 Vehicle Routing Competition. Ciccone M, Stolovitzky G, Albrecht J, eds. *Proc. NeurIPS 2022 Competitions Track.* Proceedings of Machine Learning Research, vol. 220 (ML Research Press), 35–49.

Kwon YD, Choo J, Bae D, Kim J, Hottung A (2022) Cost shaping via reinforcement learning for vehicle routing problems. Accessed September 29, 2023, https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/Team_SB.pdf.

Li Y, Archetti C, Ljubic I (2023) Emerging optimization problems for distribution in same-day delivery. Working paper, IDS Department, ESSEC Business School, Cergy-Pontoise, France.

Li Y, Zhang J, Wang Y, Shao W, Su Z, Lü Z (2022) Customer penalty-based heuristic algorithm for the EURO Meets NeurIPS 2022 Vehicle Routing Competition. Accessed September 29, 2023, https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/HustSmart.pdf.

Liu S, Luo Z (2023) On-demand delivery from stores: Dynamic dispatching and routing with random demand. *Manufacturing Service Oper. Management* 25(2):595–612.

Rockafellar RT, Wets RJB (1991) Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* 16(1):119–147.

Secomandi N, Margot F (2009) Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Oper. Res.* 57(1):214–230.

Soeffker N, Ulmer MW, Mattfeld DC (2022) Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *Eur. J. Oper. Res.* 298(3):801–820.

Stefik M (1981) Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence* 16(2):111–139.

Ulmer MW (2020) Dynamic pricing and routing for same-day delivery. *Transportation Sci.* 54(4):1016–1033.

Ulmer MW, Thomas BW, Mattfeld DC (2019) Preemptive depot returns for dynamic same-day delivery. *EURO J. Transportation Logist.* 8(4):327–361.

Ulmer MW, Goodson JC, Mattfeld DC, Thomas BW (2020) On modeling stochastic dynamic vehicle routing problems. *EURO J. Transportation Logist.* 9(2):100008.

van Doorn J, Lan L, Pentinga L, Wouda NA (2022) Solving a static and dynamic VRP with time windows using hybrid genetic search and simulation. *EURO Meets NeurIPS 2022 Vehicle Routing Competition, 2022.* Accessed March 29, 2023, https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/OptiML.pdf.

Van Heeswijk WJA, Mes MRK, Schutten JMJ (2019) The delivery dispatching problem with time windows for urban consolidation centers. *Transportation Sci.* 53(1):203–221.

Vidal T (2022) Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Comput. Oper. Res.* 140:105643.

Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Comput. Oper. Res.* 40(1):475–489.

Voccia SA, Campbell AM, Thomas BW (2019) The same-day delivery problem for online purchases. *Transportation Sci.* 53(1):167–184.

Wouda NA, Lan L, Kool W (2024) PyVRP: A high-performance VRP solver package. *INFORMS J. Comput.*, ePub ahead of print January 29, https://doi.org/10.1287/ijoc.2023.0055.

Zhang J, Van Woensel T (2023) Dynamic vehicle routing with random requests: A literature review. *Internat. J. Production Econom.* 256:108751.

Zhang X, Zhang J, Fan X (2022) Offline approximate value iteration for dynamic solutions to the multivehicle routing problem with stochastic demand. *Comput. Oper. Res.* 146:105884.