Supplementary materials: An iterative sample scenario approach for the dynamic dispatch waves problem

Leon Lan, Jasper van Doorn
Department of Mathematics, Vrije Universiteit Amsterdam, {l.lan, j.m.h.van.doorn}@vu.nl,

Niels A. Wouda, Arpan Rijal
Department of Operations, University of Groningen, {n.a.wouda, a.rijal}@rug.nl,

Sandjai Bhulai
Department of Mathematics, Vrije Universiteit Amsterdam, s.bhulai@vu.nl.

Appendix A: Additional results

This section presents additional results to the numerical experiments of Section 5. Table A.1 presents the p-values of the pair-wise T-tests for the benchmark results (Table 1). Table A.2 and A.3 show the results of the benchmark instances for 300 and 450 expected number of arrivals, respectively.

Table A.1 Overview of p-values of the pair-wise T-tests. The method with the best average gap in each row is used as the benchmark and represented with a dash (-).

Instance			Methods				
Topology	Arrival	TW	RH	DSHH	ICD-postpone	ICD-Hamming	ICD-double
R	HOM	DL2	<0.001	0.019	<0.001	-	0.165
		DL4	<0.001	<0.001	<0.001	0.826	-
		DL8	<0.001	0.005	<0.001	0.445	-
		TW2	<0.001	<0.001	<0.001	<0.001	-
		TW4	<0.001	<0.001	<0.001	<0.001	-
		TW8	<0.001	<0.001	<0.001	<0.001	-
	UNI	DL2	<0.001	0.653	<0.001	0.348	-
		DL4	<0.001	0.002	<0.001	0.128	-
		DL8	<0.001	<0.001	<0.001	<0.001	-
		TW2	<0.001	<0.001	<0.001	<0.001	-
		TW4	0.119	<0.001	<0.001	<0.001	-
		TW8	0.726	<0.001	<0.001	<0.001	-
C	HOM	DL2	<0.001	-	0.014	0.044	0.911
		DL4	<0.001	0.132	<0.001	-	0.260
		DL8	<0.001	0.019	<0.001	-	0.075
		TW2	<0.001	0.001	<0.001	<0.001	-
		TW4	<0.001	<0.001	<0.001	<0.001	-
		TW8	<0.001	<0.001	<0.001	<0.001	-
	UNI	DL2	<0.001	0.936	0.006	0.006	-
		DL4	<0.001	0.452	<0.001	0.497	-
		DL8	<0.001	0.526	<0.001	0.429	-
		TW2	<0.001	<0.001	<0.001	<0.001	-
		TW4	0.002	<0.001	<0.001	<0.001	-
		TW8	0.017	<0.001	<0.001	<0.001	-
RC	HOM	DL2	<0.001	0.052	<0.001	-	0.170
		DL4	<0.001	0.022	<0.001	-	0.130
		DL8	<0.001	0.043	<0.001	0.849	-
		TW2	<0.001	<0.001	<0.001	<0.001	-
		TW4	<0.001	<0.001	<0.001	<0.001	-
		TW8	<0.001	<0.001	<0.001	<0.001	-
	UNI	DL2	<0.001	0.169	<0.001	0.826	-
		DL4	<0.001	0.317	<0.001	0.359	-
		DL8	<0.001	0.092	<0.001	0.109	-
		TW2	0.004	<0.001	<0.001	<0.001	-
		TW4	-	<0.001	<0.001	<0.001	0.736
		TW8	-	<0.001	<0.001	<0.001	0.159

Table A.2 Overview of results with 300 expected total number of requests. The values represent the average percentage gap with respect to the hindsight solution over 200 different instances. The best average gap in each row is marked in bold. An asterisk (*) indicates the method is statistically significantly better than the other methods for that instance at the 0.05 level (with Bonferroni correction).

Instance			Methods				
Topology	Arrival	TW	RH	DSHH	ICD-postpone	ICD-Hamming	ICD-double
R	HOM	DL2	6.23	4.00	4.23	**3.95**	4.01
		DL4	11.58	8.73	9.94	**8.07***	8.65
		DL8	15.25	12.81	15.30	**11.46***	12.54
		TW2	17.05	14.31	16.23	13.61	**13.32**
		TW4	17.65	15.20	17.25	13.98	**13.95**
		TW8	16.70	14.80	16.95	13.82	**13.59**
	UNI	DL2	6.50	4.21	4.49	**3.98**	4.15
		DL4	12.35	9.53	10.69	**8.84**	9.24
		DL8	16.26	13.85	15.81	**12.81**	13.36
		TW2	16.98	15.61	16.62	14.97	**14.53**
		TW4	17.05	16.63	17.40	15.70	**15.33**
		TW8	16.56	16.93	16.81	15.81	**15.35**
C	HOM	DL2	3.18	1.41	1.58	1.43	**1.41**
		DL4	6.96	4.18	4.63	**3.99**	4.08
		DL8	8.49	6.11	7.19	**5.61***	6.08
		TW2	11.28	8.91	10.73	**8.39**	8.53
		TW4	10.63	8.59	10.57	**7.91**	8.09
		TW8	9.79	8.11	9.85	**7.51**	7.64
	UNI	DL2	4.20	1.49	1.51	**1.49**	1.49
		DL4	8.70	5.61	6.05	**5.29**	5.38
		DL8	10.94	8.26	9.61	**7.72**	8.13
		TW2	13.25	11.92	13.52	**11.17**	11.26
		TW4	13.06	11.93	13.51	11.52	**11.13**
		TW8	12.77	11.80	12.66	11.39	**11.00**
RC	HOM	DL2	5.96	3.63	4.03	**3.56**	3.73
		DL4	11.48	8.33	9.53	**7.67***	8.28
		DL8	15.36	12.60	15.19	**11.65***	12.55
		TW2	17.29	14.43	16.52	13.76	**13.56**
		TW4	17.77	14.67	17.09	14.14	**13.85**
		TW8	17.12	14.48	17.04	13.61	**13.52**
	UNI	DL2	6.15	4.21	4.41	**3.96**	4.24
		DL4	11.83	9.56	10.69	**8.78***	9.47
		DL8	15.28	13.57	15.64	**12.73**	13.27
		TW2	17.10	16.21	17.41	15.26	**15.01**
		TW4	17.32	16.72	18.00	15.99	**15.58**
		TW8	16.72	17.19	17.69	15.98	**15.80**
Average			12.58	10.57	11.84	**9.93**	10.03

Table A.3 Overview of results with 450 expected total number of requests. The values represent the average relative gap (in percentages) with respect to the hindsight solution over 200 different instances. The best average gap in each row is marked in bold. An asterisk (*) indicates the method is statistically significantly better than the other methods for that instance at the 0.05 level (with Bonferroni correction).

Instance			Methods				
Topology	Arrival	TW	RH	DSHH	ICD-postpone	ICD-Hamming	ICD-double
R	HOM	DL2	5.04	3.36	3.80	**3.18**	3.34
		DL4	9.76	7.76	9.17	**7.12***	7.53
		DL8	13.27	11.31	13.72	**10.58**	10.87
		TW2	14.40	13.05	15.02	12.95	**12.04***
		TW4	14.68	14.15	15.84	13.79	**12.86***
		TW8	14.15	14.06	15.84	14.05	**12.47***
	UNI	DL2	5.30	3.54	3.83	**3.48**	3.49
		DL4	10.97	8.67	9.74	8.34	**8.30**
		DL8	14.48	12.89	14.56	12.60	**12.36**
		TW2	15.55	15.89	16.00	15.86	**14.28***
		TW4	15.79	16.84	16.98	16.48	**15.18***
		TW8	**14.85**	16.85	16.54	16.65	15.02
C	HOM	DL2	1.77	0.35	0.37	0.34	**0.33**
		DL4	4.75	2.67	3.31	**2.50**	2.74
		DL8	6.19	4.57	5.83	**4.08***	4.48
		TW2	9.24	7.78	9.55	7.38	**7.31**
		TW4	8.78	7.51	9.40	7.19	**7.01**
		TW8	8.25	6.99	8.68	6.73	**6.31***
	UNI	DL2	2.91	0.40	0.41	**0.37**	0.38
		DL4	6.41	4.17	4.69	**4.01**	4.06
		DL8	8.51	6.50	7.97	**6.38**	6.41
		TW2	11.60	10.75	12.30	10.70	**10.10***
		TW4	11.59	11.00	12.28	11.08	**10.32***
		TW8	11.27	10.84	11.40	10.87	**9.87***
RC	HOM	DL2	4.70	2.85	3.36	**2.66**	2.80
		DL4	9.15	7.19	8.84	**6.52**	6.95
		DL8	12.64	10.79	13.54	**10.03**	10.39
		TW2	14.95	13.07	15.06	12.86	**11.93***
		TW4	15.64	13.69	15.81	13.47	**12.57***
		TW8	15.13	13.61	15.64	13.51	**12.48***
	UNI	DL2	5.68	2.96	3.29	**2.86**	2.95
		DL4	12.05	8.12	9.48	**7.91**	8.01
		DL8	15.68	12.27	14.18	**11.68**	11.79
		TW2	18.24	15.82	16.64	15.63	**14.64***
		TW4	18.52	16.87	17.32	16.48	**15.60***
		TW8	18.38	16.73	17.06	16.48	**15.30***
Average			11.12	9.61	10.76	9.36	**8.96**

Appendix B: Implementation details of solving static VRPTW-DW

To solve static VRPTW-DWs, we rely on PyVRP (Wouda, Lan, and Kool 2024), which implements a hybrid genetic search (HGS) algorithm that builds upon HGS-CVRP (Vidal 2022). We first present a general outline of HGS in Section B.1, and we then describe how to extend it to solve routing problems with dispatch windows in Section B.2. Section B.3 presents our parameter tuning procedure and the resulting parameters.

B.1. Algorithm outline

HGS combines elements of a genetic algorithm and a local search algorithm, resulting in an effective synergy between the genetic algorithm's ability to explore a diverse set of solutions and local search to identify local optima. In particular, HGS manages a population of candidate solutions, which are iteratively recombined and improved using a local search procedure. An outline of the algorithm is given in Algorithm 1.

Algorithm 1 PyVRP's hybrid genetic search

Input: initial solutions $s_1, \ldots, s_p$
Initialize s^* to the initial solution with the best objective value
while time limit is not met **do**
    Select two parent solutions from the population using binary tournament
    Apply a crossover operator on the parent solutions to generate an offspring solution
    Improve the offspring using a local search algorithm to obtain a candidate solution s^c
    Add the candidate solution to the population
    if s^c has better objective value than s^* **then**
        $s^* \leftarrow s^c$
    end if
    if population size exceeds maximum size **then**
        Remove the solutions with the lowest fitness until the population is at minimum size
    end if
end while
Output: the best-found solution s^*

The HGS algorithm works as follows. HGS initializes its population by a set of solutions given as input to the algorithm. In each iteration, the algorithm selects two existing parent solutions from the population using a binary tournament, favoring solutions with higher fitness (i.e., a weighted combination of cost and diversity value). A crossover operator then combines the two parent solutions to generate an offspring solution that inherits features from both parents. After the crossover, the offspring solution is further improved using a local search procedure. The resulting candidate solution is first added to the population. If the candidate solution improves over the best solution found so far, it is also registered as the new best solution. Upon reaching the maximum population size, a survivor selection mechanism removes the least fit solutions until

the population is back at the minimum size. The algorithm continues until a provided stopping criterion is met, returning the best solution found.

B.2. Dispatch windows extension

In this section, we describe how to extend PyVRP to solve VRPs with dispatch windows. PyVRP implements the path concatenation schemes of Vidal et al. (2013), which enables it to evaluate local search moves involving time windows in amortized $O(1)$ time. The scheme relies on infeasible solutions that incur additional penalties based on *time-warping* (Nagata, Bräysy, and Dullaert 2010): late arrivals at customer locations are warped back in time and penalized.

We first outline the concatenation scheme as introduced in Vidal et al. (2013) and later we describe how we extend this scheme to take into account dispatch windows.

Local search moves can be seen as a separation of routes into subsequences of location visits, which are then concatenated together into new routes. Define $\sigma = (\sigma_0, \dots, \sigma_l)$ as a sequence of visits (including depot). The operator $\oplus$ concatenates two subsequences σ, σ' together, forming a new sequence of visits. For each subsequence σ, we maintain the following data: the minimum duration $D(\sigma)$, the time warp $TW(\sigma)$, the earliest $E(\sigma)$ and latest $L(\sigma)$ departure time to the first location with minimum duration and minimum time warp use, the cumulative distance $C(\sigma)$ and the cumulative load $Q(\sigma)$. For a sequence $\sigma^0 = \{i\}$ consisting of a single visit $i \in V$, the data is initialized as $D(\sigma^0) = \mu_i, TW(\sigma^0) = 0, E(\sigma^0) = e_i, L(\sigma^0) = l_i, C(\sigma^0) = 0$, and $Q(\sigma^0) = q_i$.

The following proposition shows how to compute the same data for a concatenation of sequences:

PROPOSITION 1 **(Concatenation of two sequences Vidal et al. (2013))**. *Let* $\sigma = (\sigma_i, \dots, \sigma_j)$ *and* $\sigma' = (\sigma'_{i'}, \dots, \sigma'_{j'})$ *be two subsequences of visits. The concatenated subsequence* $\sigma \oplus \sigma'$ *is characterized by the following data:*

$$D(\sigma \oplus \sigma') = D(\sigma) + D(\sigma') + \delta_{\sigma_j \sigma_{i'}} + \Delta_{WT} \tag{1}$$

$$TW(\sigma \oplus \sigma') = TW(\sigma) + TW(\sigma') + \Delta_{TW} \tag{2}$$

$$E(\sigma \oplus \sigma') = \max\{E(\sigma') - \Delta, E(\sigma)\} - \Delta_{WT} \tag{3}$$

$$L(\sigma \oplus \sigma') = \min\{L(\sigma') - \Delta, L(\sigma)\} + \Delta_{TW} \tag{4}$$

$$C(\sigma \oplus \sigma') = C(\sigma) + C(\sigma') + c_{\sigma_j \sigma'_{i'}} \tag{5}$$

$$Q(\sigma \oplus \sigma') = Q(\sigma) + Q(\sigma') \tag{6}$$

where $\Delta = D(\sigma) - TW(\sigma) + \delta_{\sigma_j \sigma'_{i'}}, \Delta_{WT} = \max\{E(\sigma') - \Delta - L(\sigma), 0\}$ *and* $\Delta_{TW} = \max\{E(\sigma) + \Delta - L(\sigma'), 0\}$.

Local search moves, which generally consist of up to five concatenation procedures, can be evaluated in constant time by computing the relevant data using this proposition.

We now describe how the concatenation scheme can be extended for dispatch windows. For a sequence σ, we additionally compute the earliest $R^-(\sigma)$ and the latest $R^+(\sigma)$ dispatch time. For a sequence $\sigma^0 = \{i\}$

consisting of a single visit $i \in V$, the data is initialized as $R^-(\sigma^0) = r_i^-$ and $R^+(\sigma^0) = r^+$. The concatenation scheme of two subsequences σ and σ' is extended using the following expressions:

$$R^-(\sigma \oplus \sigma') = \max\{R^-(\sigma), R^-(\sigma')\} \tag{7}$$

$$R^+(\sigma \oplus \sigma') = \min\{R^+(\sigma), R^+(\sigma')\} \tag{8}$$

Having introduced the dispatch window data, we aim to introduce additional time warp penalties when there is a violation of the time window data $E(\sigma), L(\sigma)$ and dispatch window data $R^-(\sigma), R^+(\sigma)$. To this end, we rely on the following proposition:

PROPOSITION 2 **(Vidal et al. (2013))**. *There exists a set of starting dates t that minimize both the time warp use and the duration to service a sequence σ. This set is a segment, notated as $\mathcal{T}^{MIN}(\sigma) = [E(\sigma), L(\sigma)]$. The minimum duration $D(\sigma)(t)$ and time warp use $TW(\sigma)(t)$ as a function of t can be expressed as follows, where $D(\sigma)$ and $TW(\sigma)$ represent the minimum duration and time warp use, respectively:*

$$D(\sigma)(t) = D(\sigma) + \max\{E(\sigma) - t), 0\} \tag{9}$$

$$TW(\sigma)(t) = TW(\sigma) + \max\{t - L(\sigma), 0\} \tag{10}$$

The interval $[E(\sigma), L(\sigma)]$ is a time interval during which a vehicle should dispatch in order to minimize the duration and time warp use. Equation (9) describes how additional duration is incurred when dispatching earlier than $E(\sigma)$ (that is, due to additional waiting times). Since we aim to minimize the total cost, we can always dispatch earlier without introducing additional costs. Equation (10) introduces an additional time warp penalty when starting later than $L(\sigma)$. In particular, whenever a vehicle dispatches at its earliest dispatch time with $R^-(t) > L(\sigma)$, it is warped back in time to dispatch at time $L(\sigma)$.

We compute the total time warp involving penalties due to dispatch windows as follows:

$$TW_{DW}(\sigma) = TW(\sigma) + \max\{R^-(\sigma) - L(\sigma), 0\} + \max\{R^-(\sigma) - R^+(\sigma), 0\} \tag{11}$$

The first term is the time warp penalty incurred from the concatenation scheme in Proposition 1. The second term is the time warp penalty incurred from starting later than $L(\sigma)$, which follows from Proposition 2. Finally, the third term adds an additional time warp penalty when the earliest dispatch time exceeds the dispatch time, i.e., whenever $R^-(\sigma) > R^+(\sigma)$. Since there is a violation in the dispatch times, this expression introduces a "forward" time warp, meaning that the vehicle is warped from the latest dispatch time $R^+(\sigma)$ to the moment of earliest dispatch time $R^-(\sigma)$. The total time warp (11) is then used to evaluate local search moves. For more details on the local search procedure, we refer to Vidal et al. (2013) and Wouda, Lan, and Kool (2024).

B.3. Parameter tuning

This section outlines the parameter tuning procedure for the static solver and presents the resulting parameters. PyVRP's HGS algorithm has a large number of parameters (see Table B.4). The default parameters of PyVRP are primarily optimized for solving larger VRP instances where more time is allowed. Consequently, we use these parameters to compute the routing cost. However, solving scenarios requires slightly different parameters to find good solutions within significantly shorter time limits of just a few seconds.

Because of the large number of parameters and operators in HGS, we opted for a prescriptive parameter tuning approach. First, we derived reasonable lower and upper bounds for the values of each parameter. Then, we split the parameters into five groups: (1) those related to population management, (2) those related to penalty management, (3) those related to neighborhood, (4) those related to the local search node operators, and finally, (5) those related to the local search route operators. The genetic algorithm parameters were not tuned as they were irrelevant for solving scenario instances. We generated 499 parameter configurations for a single group using a Latin hypercube design, and added the default setting as a control. We used the set of 120 benchmark instances from Gehring and Homberger with 200 and 400 customers. We solved each instance ten times using a different seed, for each of the 500 configurations. We then took the best configuration and made that the default for the first group. With this default, we then repeated this approach for the remaining groups. Finally, we manually tuned the parameters using observations from the tuning process. As shown in Table B.4, the resulting parameter values for solving scenario instances are a smaller population size (allowing for faster convergence to good solutions), more frequent updates of the penalties, and the removal of many local search operators. Tuning the parameters resulted in a respectable performance improvement when compared to the default parameters: on average, the solutions were 3% better with a time limit of two seconds.

Table B.4 Parameters of PyVRP's solver, including the default parameter values and the parameter values for solving scenarios. Scenario parameter values are marked in bold if they differ from the default value.

Category	Parameter	Default	Scenario
Genetic algorithm	Repair probability	80%	80%
	Number of non-improving iterations before restart	20000	20000
Population	Minimum population size	25	**5**
	Population generation size	40	**3**
	Number of elite solutions	4	**2**
	Number of close solutions	5	**2**
	Lower bound diversity	0.1	0.1
	Upper bound diversity	0.5	0.5
Penalty management	Initial capacity penalty	20	20
	Initial time warp penalty	6	6
	Repair booster	12	12
	Number of registrations between penalty updates	50	**10**
	Penalty increase factor	1.34	**1.30**
	Penalty decrease factor	0.32	**0.50**
	Target feasible	0.43	0.43
Neighborhood	Number of neighbors	40	40
	Weight waiting time	0.2	0.2
	Weight time warp	1.0	1.0
	Symmetric proximity	True	True
	Symmetric neighbors	False	False
Node operators	Include $(1,0)$-exchange	True	True
	Include $(2,0)$-exchange	True	**False**
	Include $(3,0)$-exchange	True	**False**
	Include MoveTwoClientsReversed	True	**False**
	Include $(1,1)$-exchange	True	**False**
	Include $(2,1)$-exchange	True	**False**
	Include $(2,2)$-exchange	True	**False**
	Include $(3,2)$-exchange	True	**False**
	Include $(3,3)$-exchange	True	**False**
	Include 2-OPT operator	True	True
Route operators	Include RELOCATE* operator	True	**False**
	Include SWAP* operator	True	True

Appendix C: Limited fleet

In the problem formulation presented in Section 3, we assume that there is an unlimited fleet of vehicles available at each epoch to serve all requests. This assumption was introduced in the *EURO meets NeurIPS* 2022 vehicle routing competition (Kool et al. 2022a) to ensure that all requests could be served on time, but it clearly makes the problem less realistic since using more vehicles incurs no additional cost. In this appendix, we discuss how to modify the original DDWP formulation to account for a limited number of cost-free vehicles, while ensuring that all requests can still be served on time. Our approach is similar to Van Heeswijk, Mes, and Schutten (2019), making a distinction between a primary and secondary fleet of vehicles.

We assume that at each epoch t, there is a planned number of *primary* vehicles that can be dispatched to serve requests. Using vehicles from the primary fleet incurs no fixed costs, and unused primary vehicles from one epoch carry over to the next epoch. In case the number of primary vehicles in an epoch is not enough to serve all requests, one can decide to hire vehicles from an external party (e.g., through crowd-sourcing). We call vehicles from this fleet *secondary*, and hiring additional secondary vehicles incur a fixed cost in addition to the routing cost. In this modified problem, we can still ensure that all requests are served on time, but using more vehicles than planned is now penalized. This change requires some adjustments to the static model and Markov decision process (MDP) since we must distinguish between a *heterogeneous* fleet of vehicles. The adjustments are described in the next subsections.

C.1. Static model

The modified static model extends the VRPTW-DW (3.2) as follows. The fleet $\mathcal{K}$ now consists of the primary vehicles and the secondary vehicles, each with capacity Q. Let $f_k \geq 0$ denote the fixed cost of using vehicle k, which is zero if k is a primary vehicle and nonzero otherwise. Each vehicle k has an earliest time moment E_k at which it is available for delivery. This attribute is necessary to distinguish between vehicles that are available at the current epoch t, or some later epoch $t' > t$. Indeed, whenever we sample a scenario instance for epoch $t' > t$, we must impose an earliest time moment $T_{t'}$ on primary vehicles that are only planned in future epochs to prevent them from serving requests in the current epoch. The modified static model is then formulated as follows:

$$\min_{x,\theta} \quad \sum_{k\in\mathcal{K}} f_k \sum_{j\in\mathcal{N}} x_{0jk} + \sum_{k\in\mathcal{K}} \sum_{i,j\in\mathcal{V}} c_{ij} x_{ijk} \tag{12a}$$

$$\text{s.t.} \quad E_k \leq \theta_{0k} \qquad \forall k \in \mathcal{K}, \tag{12b}$$

$$(1b) - (1m). \tag{12c}$$

Objective (12a) minimizes the total vehicle fixed costs and the total routing cost. Constraints (12b) ensure that a vehicle is dispatched no earlier than when it is available. All constraints from the VRPTW-DW model (1) still apply.

C.2. Markov decision process

C.2.1. State space The state is now a tuple $z_t = (s_t, K_t^p)$, where s_t represents the set of requests that are known but not yet dispatched and K_t^p denotes the number of primary vehicles at epoch t.

C.2.2. Action space The action is now a tuple $b_t = (a_t, k_t^p)$, where $a_t \in \mathcal{A}(s_t)$ represents the subset of requests to dispatch and $k_t^p \in \{0, 1, \ldots, K_t^p\}$ is the number of primary vehicles to use for dispatching the requests a_t. The action space in state $z_t = (s_t, K_t^p)$ is thus defined as

$$\mathcal{B}(z_t) = \mathcal{A}(s_t) \times \{0, 1, \ldots, K_t^p\}.$$

C.2.3. State transition Let P_t denote the primary vehicles that become available at epoch t. The transition to the next state z_{t+1} depends on the selected action (a_t, k_t^p) and the unknown future realization ω_{t+1}. The state transitions from $z_t = (s_t, K_t^p)$ to $z_{t+1} = ((s_t \setminus a_t) \cup \omega_{t+1}, (K_t^p \setminus k_t^p) \cup P_{t+1})$.

C.2.4. Optimality equation The direct cost $C(z_t, b_t)$ of a specific action $b_t = (a_t, k_t^p)$ in state z_t is given by the cost of routing the requests in a_t at time T_t and using k_t^p primary vehicles. The direct cost is obtained by solving (12) with a_t as the set of requests and the fleet of vehicles $\mathcal{K}$ that consists of k_t^p primary vehicles (with no fixed cost) and a large number of secondary vehicles (with fixed cost). 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:

$$V(z_t) = \begin{cases} \min\limits_{b \in \mathcal{B}(z_t)} C(z_t, b) & \text{if } t = |\mathcal{T}|, \\ \min\limits_{b \in \mathcal{B}(z_t)} \left[C(z_t, b) + \mathbb{E}_{\omega_{t+1}}[V(z_{t+1})] \right] & \text{otherwise.} \end{cases} \tag{13}$$

C.3. Numerical experiment

We now evaluate the ICD methods on the modified DDWP with a limited primary fleet, following an identical experimental setup as used in Section 5. In addition to what is described in the main body, we must determine the number of planned primary vehicles per epoch. To this end, we first solve each instance assuming an unlimited primary fleet of vehicles using a greedy strategy that immediately dispatches all requests when revealed. The resulting solution determines the number of planned primary vehicles for each epoch. Indeed, it can be expected that operators plan a certain number of vehicles based on the expected demand, which is precisely what we aim to capture using this greedy strategy. Furthermore, we assume that hiring a vehicle from the secondary fleet has a fixed upfront cost of two hours (recall that the routing cost equals the travel time).

We use the set of instances with time window characteristics and unimodal arrivals. We solve these instances first assuming an unlimited fleet, and also assuming a limited fleet (obtained by the procedure outlined above). We then compute the gap with respect to the hindsight solution obtained after 600 seconds, assuming an unlimited fleet. Figure 1 plots the gap to the hindsight solution assuming an unlimited fleet and limited fleet for each method. The results show that having only a limited number of primary vehicles based on a greedy strategy achieves a near-identical performance compared to the case with an unlimited fleet. This means that limiting the vehicles (to a sufficient number to serve all requests) does not result in significantly more costs.

Finally, it is not uncommon in the literature to use primary vehicles multiple times (Voccia, Campbell, and Thomas 2019, Klapp, Erera, and Toriello 2018a, Van Heeswijk, Mes, and Schutten 2019). This can be incorporated into the static problem formulation by redefining it as a multi-trip VRP, while also tracking the times at which the vehicle returns to the depot in the state description of the MDP. We reserve this extension for future work.

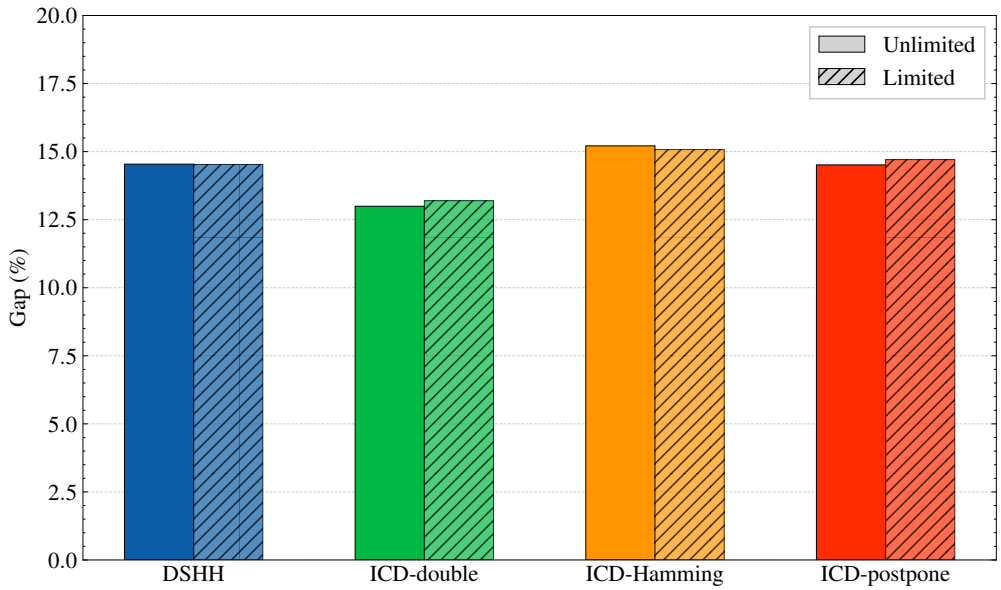

Figure 1 **Average percentage gaps for different fleet configurations.**