A Novel Generalised Meta-Heuristic Framework for Dynamic Capacitated Arc Routing Problems

Hao Tong, Leandro Minku, Stefan Menzel, Bernhard Sendhoff, Xin Yao

2022

Preprint:

This is an accepted article published in IEEE Transactions on Evolutionary Computation. The final authenticated version is available online at: https://doi.org/10.1109/TEVC.2022.3147509

A Novel Generalised Meta-Heuristic Framework for Dynamic Capacitated Arc Routing Problems

Hao Tong[1], Leandro L. Minku[1], Stefan Menzel[2],
Bernhard Sendhoff[2], and Xin Yao[1]

[1] School of Computer Science, University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK
{hxt922, L.L.Minku, x.yao}@cs.bham.ac.uk
[2] Honda Research Institute Europe GmbH, 63073 Offenbach, Germany
{stefan.menzel,bs}@honda-ri.de

The article has been accepted for publication in IEEE Transactions on Evolutionary Computation.

DOI: https://doi.org/10.1109/TEVC.2022.3147509

A Novel Generalised Meta-Heuristic Framework for Dynamic Capacitated Arc Routing Problems

Hao Tong, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao

***Abstract*—The capacitated arc routing problem (CARP) is a challenging combinatorial optimisation problem abstracted from typical real-world applications, like waste collection and mail delivery. However, few studies considered dynamic changes during the vehicles' service, which can make the original schedule infeasible or obsolete. The few existing studies are limited by dynamic scenarios that can suffer single types of dynamic events, and by algorithms that rely on special operators or representations, being unable to benefit from the wealth of contributions provided by the static CARP literature. Here, we provide the first mathematical formulation for dynamic CARP (DCARP) and design a simulation system to execute the CARP solutions and generate DCARP instances with several common dynamic events. We then propose a novel framework able to generalise all existing static CARP optimisation algorithms so that they can cope with DCARP instances. The framework has the option to enhance optimisation performance for DCARP instances based on a restart strategy that makes no use of past history, and a sequence transfer strategy that benefits from past optimisation experience. Empirical studies are conducted on a wide range of DCARP instances. The results highlight the need for tackling dynamic changes and show that the proposed framework significantly improves over existing algorithms.**

***Index Terms*—Dynamic capacitated arc routing problem, Virtual task, Restart strategy, Sequence transfer strategy, Generalised optimisation framework.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) is a classical and important combinatorial optimisation problem with a range of applications in the real world, such as waste collecting [1] and winter gritting [2]. A map, represented as a graph, contains a set of vertices and edges, where each edge has a travel cost and some edges also have demands required to be served by vehicles. The edges with demands are called tasks. By optimizing CARP, a number of vehicles with limited capacities are assigned to the graph to serve all tasks spending the lowest possible total travel cost.

Over the past decades, there has been much research focusing on solving CARP. Constructive heuristic methods, such as Ulusoy's split [3] and Path-Scanning [4], were proposed to construct feasible executable solutions for CARP based on an optimised sequence of tasks. Tabu search [5], memetic algorithms [6], and others were also proposed to optimise the cost of CARP solutions. In addition, many efficient algorithms have been proposed to tackle the large scale CARP with a very large number of tasks and vertices [7], [8]. Besides, many different variants of CARP have been investigated in literature [9], [10]. For example, uncertain CARP considered the environment's stochastic nature that the cost or demand of edges is a random variable during the optimisation [11], [12], which aimed at obtaining the solution being robust for the uncertain environment. Multi-depot CARP considers several different depots in the graph [13], and open CARP allows the routes to be open with different starting and ending nodes [14]. In addition, there are many valuable CARP algorithms proposed for different practical scenarios. For example, algorithms for split-delivery CARP where the edge demand can be served by several vehicles [15], algorithms for periodic CARP where the tasks are required to be served with a certain number of times over a given multiperiod horizon [16], as well as algorithms for time CARP where the time restriction instead of the volume restriction limits the vehicles' capacity [17].

However, all these studies concentrate on static CARP, where the problem is constant over time. In real applications, dynamic changes usually happen when vehicles are in service, thus influencing the vehicles' service. For example, a road may be closed or congested due to an accident, or new tasks may emerge during the vehicles' service. When that happens, a new graph, i.e. a new problem instance, is formed, in which vehicles stop at different locations, labelled as outside vehicles, with various amounts of remaining capacities. We aimed at re-scheduling the service plan when dynamic events happen, that have influence on the current schedule. This is referred to as Dynamic CARP (DCARP) in our paper[1]. For clarity, the following three different concepts are used throughout our paper:

- **DCARP:** A variant of CARP where the status of a graph is changed due to dynamic events occurring *during a CARP solution's execution*.
- **DCARP Instance:** The updated graph with some outside vehicles after the dynamic events happen.
- **DCARP Scenario:** The scenario contains a series of DCARP instances with the whole service process, starting from executing an initial solution in the original CARP map until all tasks are served.

DCARP, to the best of our knowledge, was firstly investigated in [18] when considering the salting route optimisation

Hao Tong and L.L.Minku are with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK. (email: hxt922@cs.bham.ac.uk, L.L.Minku@cs.bham.ac.uk.)

Stefan Menzel and Bernhard Sendhoff are with the Honda Research Institute Europe GmbH, 63073 Offenbach, Germany. (email: stefan.menzel@honda-ri.de, bs@honda-ri.de)

Xin Yao (*the corresponding author*) is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. (email:xiny@sustech.edu.cn)

[1]Mei et al. [11] used the term DCARP to denote the uncertain CARP, which is different from the DCARP investigated in this paper.

problem, but few studies in the literature have focused on DCARP so far. Mariam et al. [19] solved DCARP with time-dependent service costs motivated from winter gritting applications. Liu et al. [20] defined some possible changes in DCARP and proposed a benchmark generator for DCARP [21]. Furthermore, Marcela et al. [22] deal with the rescheduling for DCARP, which considered the failure of vehicles, and Wasin et al. [23] considered new tasks in DCARP. Besides, a robot path planning problem [24] and our previous work [25] focused on the split scheme in DCARP. Split schemes convert an ordered task sequence into an executable solution with multiple explicit routes.

Even though DCARP has been defined in existing work, there still is a lack of a formal mathematical formulation of DCARP to the best of our knowledge. Therefore, in this paper, we provide the first mathematical formulation of DCARP. Then, a simulation system is designed to simulate the behaviour of vehicles' service processes in the real world as the research platform for DCARP. In [21], Liu et al. have proposed a benchmark generator for DCARP. However, their generator lacks the important step of executing solutions and, thus, is unsuitable for our DCARP scenarios investigated in this paper. In contrast, our simulation system generates new DCARP instances by executing the CARP solution on a given map and simulates several commonly occurring dynamic events. Furthermore, our simulator system considers the dynamic events much closer to the reality. For example, tasks will not disappear once generated unless being served by vehicles, the cost of an edge will not be smaller than a basic cost, etc.

Based on our simulation system, we design a novel framework capable of generalising almost all algorithms designed for static CARP in the literature to optimise DCARP. The idea of virtual tasks is proposed to make all outside vehicles in a dynamic instance return to the depot virtually. Then, a virtual task is constructed to represent the state of the corresponding outside vehicle. Thus, a dynamic DCARP instance is converted into a 'static' CARP instance so that existing algorithms designed for static CARP can be applied to solve DCARP directly. At the same time, as a dynamic scenario is composed of a series of DCARP instances, the similarity between DCARP instances may help to solve a new DCARP instance. Therefore, we propose two strategies for generating initial solutions in our framework, namely a *sequence transfer strategy* and a *restart strategy*, to solve the new DCARP instance. The sequence transfer strategy generates a potentially good solution based on the previous optimisation experience by transferring the sequence of remaining unserved tasks. The restart strategy starts from scratch without using any information and optimises each DCARP instance independently of each other.

The remainder of this paper is organised as follows. Section II discussed the related work on DCARP and this paper's motivation. After that, a general mathematical formulation of DCARP and a simulation system for DCARP are provided in Section III. Section IV introduces the main algorithm of our proposed generalised optimisation framework for DCARP. Section V presents the empirical study on the proposed framework to evaluate its efficiency. Section VI concludes the paper.

II. Related work and motivation

In the literature, there are two main research fields, which target the (re)scheduling of vehicles in dynamic environments: Dynamic CARP (DCARP) and dynamic vehicle routing problem (DVRP). As main difference, DCARP focuses on serving arcs while DVRP focuses on serving vertices. With respect to DCARP, only few approaches heve been proposed to solve the dynamic problem. Liu et al. [20] proposed a memetic algorithm with a new distance-based split scheme (MASDC) for DCARP. However, its performance is limited since it suffers from noise in the fitness evaluation due to the impact of random splits, as well as the neglecting available vehicles placed in the depot. Monroy et al. [22] only considered the broken down vehicles and presented a heuristic to minimise the operations and disruption cost. Padungwech et al. [23] only considered the new tasks during the vehicles' service. They applied tabu search to optimise the DCARP, in which the solution is represented as routes with different start vertices.

As stated above, the dynamic vehicle routing problem (DVRP) focuses on routing planning and targets the rescheduling of vehicles to serve tasks (customers) in a dynamic environment. Research work in the area of DVRP [26] mainly comprises two categories called dynamic deterministic VRP and stochastic VRP according to if problem knowledge is used during the optimisation or not. The dynamic deterministic VRP assumes that no knowledge about the new DVRP instances is available before a change happens. Therefore, periodic or continuous re-optimisation are used to handle the dynamic instances. The continuous re-optimisation re-schedules the current routes whenever the available data changes, but the periodic re-optimisation uses an event manager to maintain new customers and re-optimise periodically. For example, Montemanni et al. [27] applied the ant colony optimisation to periodically re-optimise new DVRP instances. The ants select the next vertex according to a probability distribution based on the pheromone trail, maintained and updated during the optimisation. Hanshar et al. [28] clustered the served customers to a node which was combined with new customers to form the chromosome of the genetic algorithm (GA). Then, GA reproduction operators are applied to periodically re-optimise the current routes. On the other hand, the dynamic stochastic VRP benefits from the problem's prior knowledge. Three strategies, including stochastic modelling [29], sampling method and look-ahead dynamic routing [30] were mainly used in the literature.

Even though it might be possible to transform DVRP instances into DCARP instances, this has not been attempted before in the literature, and the transformation form capacitated VRP (CVRP) to CARP increases the problem's dimension because the number of required nodes obtained in CVRP is greater than CARP's tasks [31]. Therefore, it is unknown how well algorithms for DVRP would work for DCARP problems. Moreover, most DVRP work considered only changes corresponding to the addition of new customers.

On the other hand, the existing works for DCARP proposed algorithms for DCARP from their perspectives and all of them require the design of special operators or representations to

tackle the constraints of DCARP. Since both static CARP and DCARP aim to optimise the task sequence and the schedule of routes for vehicles, it would be valuable if the operators designed for static CARP could also be used for DCARP, enabling a wealth of existing CARP algorithms to be applied to DCARP. This motivated us to propose a generalised optimisation framework for DCARP.

There are two representations commonly used in the optimisation algorithm for CARP in the literature. The first type provides all explicit routes in the solution, separated by a dummy task [6], while another type is an ordered list of tasks without separation. These two types of representations are usually used together in the algorithm for CARP. For example, constructive heuristics, such as Path-Scanning and Augment-Merge, generate solutions with explicit routes. This representation is easy to apply with a local search operator, such as *Single Insertion*, *Double Insertion* and *Swap*. The representation with an ordered list of tasks is used in some meta-heuristic algorithms with crossover operators, like memetic algorithms [1], [6]. The split scheme, Ulusoy's split [3], is always used to convert an ordered list of tasks to a solution with explicit routes.

The solutions of DCARP can also be represented in two different ways. However, the calculation for cost and capacity violation of the routes corresponding to the outside vehicles are required to be specifically considered due to the fact that outside vehicles have different locations and remaining capacities. In addition, it is much more complex to use an ordered list of tasks as the solution representation during the optimisation because the Ulusoy's split is not suitable anymore [3] and specific split schemes are required for DCARP. Even though new split schemes have been proposed in our previous work [25], the high complexity along with computational costs and lack of optimality limits their performance. Overall, generalising algorithms for static CARP to DCARP by specifically dealing with the routes corresponding to outside vehicles is a significantly complex approach. A framework to generalise existing algorithms from CARP would be desirable. In this paper, we propose a novel framework, which enables the adoption of existing CARP algorithms to DCARP. In the next section (Section III), we will introduce the mathematical formulation and a newly designed simulation system, followed by our novel framework in Section IV.

III. Problem Formulation and Simulation System

In this section, we provide the first mathematical formulation for DCARP. The mathematical notations used in this paper are summarised in Table I. A new simulation system is then designed to generate benchmark sets from an existing CARP benchmark as platform for testing DCARP algorithms.

A. Notations and Mathematical Formulation

A DCARP scenario is composed of a series of DCARP instances: $\mathcal{I} = \{I_0, I_1, ..., I_m, ..., I_M\}$. Each DCARP instance corresponds to a problem state, which contains all the information regarding the state of the map and vehicles involved in the routing problem, and highly depends on the previous instance and the solution's execution. The initial problem instance I_0 is a conventional static CARP, in which all vehicles are located at the depot and have the same capacities. We can obtain an initial solution from the initial static CARP and execute this solution in the graph. During the execution, some changes [20] happen at random time points when vehicles are in service, thus changing the problem instance and requiring a new solution to be computed. Vehicles then continue to serve tasks from the positions they had stopped (stop points) following the new solution. DCARP terminates when all tasks are served, and all vehicles have returned to the depot. In a DCARP scenario, the key objective is to achieve a schedule cost, which is as low as possible for each DCARP instance. Therefore, we mainly focus on the mathematical formulation of one DCARP instance.

TABLE I
Glossary of Mathematical Notations Used in this Paper

Symbols	Meaning
G	Graph $G = (V, A)$
V	Set of vertices.
A	Set of arcs.
v_0	The depot.
$dm(u)$	The demand of a arc u.
$dc(u)$	The deadheading cost of a arc u.
$sc(u)$	The serving cost of a arc u.
R	Set of tasks.
N_t	The number of real tasks.
N_m	The maximum number of vehicles.
Q	The capacity of empty vehicles.
OV	The set of outside vehicles.
N_{ov}	The number of outside vehicles.
q_k	The remaining capacity of the kth outside vehicle.
$mdc(v_i, v_j)$	The minimal total deadheading cost from vertex v_i to v_j.
$head_t$	The head node of task t.
$tail_t$	The tail node of task t.
S	A DCARP solution.
r_k	The kth route.
l_k	The number of tasks in kth route.
RC	A route's total cost.
TC	A solution's total cost.

The map for any DCARP instance I_m is regarded as a graph G. Suppose the map of a DCARP instance I_m is represented by $G = (V, A)$ with a set of vertices V, arcs (directed links) A. There is a depot $v_0 \in V$ in the graph, which contains vehicles that are not yet serving any tasks. The set A is represented by

$$A = \{< v_i, v_j > | v_i, v_j \in V\}$$

where for each arc $< v_i, v_j >$, v_i is the head vertex and v_j is the tail vertex. A given arc $< v_i, v_j >$ only exists if it is possible to traverse from vertex v_i to vertex v_j without passing through other vertices. Each arc u in the graph is associated with a deadheading cost dc_u, a serving cost sc_u and a demand dm_u. The deadheading cost of an arc means that the vehicle just traverse it without serving while the serving cost is the cost when vehicles serve this arc. The deadheading cost has been included in the serving cost such that the deadheading cost is not required to be calculated when the vehicle serves an arc. A subset $A_R \in A$ contains all arcs required to be served in the graph. The arc $u \in A_R$ is named as 'task' and have

a positive demand $dm_u > 0$. All these tasks together form a task set $R = \{t | t \in A_R\}$.

The DCARP instance I_0 only contains vehicles at the depot. As for DCARP instances $I_i | i > 1$, in addition to vehicles that are currently at the depot, there may also be several outside vehicles with remaining capacities. These are vehicles that had already started to serve tasks when a given dynamic event occurred. Suppose there are N_m vehicles in total with a maximum capacity Q at the depot and N_{ov} ($N_{ov} \leq N_m$) outside vehicles with remaining capacities $\{q_1, q_2, ..., q_{N_{ov}}\}$. The stop points (locations) of the outside vehicles are labelled as $OV = \{v_1, v_2, ..., v_{N_{ov}}\}$. The optimisation of DCARP aims to reschedule the plan to serve available tasks with minimal cost considering both, outside and depot vehicles.

A DCARP solution $S = \{r_1, r_2, ..., r_{N_{ov}}, ..., r_K\}$ contains K routes, where the routes r_1 to $r_{N_{ov}}$ start from outside vehicles while routes $r_{N_{ov}+1}$ to r_K start from the depot. The task sequence represented by a given route r_k can be expressed as $r_k = \{v_k, t_{k,1}, t_{k,2}, ..., t_{k,l_k}, v_0\}$, where the vehicle starts from stop point v_k and returns to the depot v_0, and l_k denotes the number of tasks served by route r_k. For routes r_k, where $k > N_{ov}$, v_k equals to v_0. In addition, a DCARP solution has to satisfy three constraints which are the same as constraints in static CARP:

- Each route served by one vehicle must return to the depot.
- Each task has to be served once.
- The total demand for each route served by one vehicle cannot exceed the vehicle's capacity Q.

Even though the maximum number of vehicles is also a constraint in DCARP, it does not affect the objective function. Therefore, we do not consider this limitation during the optimization process, such that the algorithm can obtain a solution with as many routes as possible. The maximum number of vehicles will be considered after the optimisation process finishes, when an executable solution is generated. This is explained in the deployment policy, which will be discussed in Section III-B. In addition, due to the different remaining capacities for outside vehicles, the capacity constraint is required to be formulated for each outside vehicle separately. As a result, the objective function for DCARP is given as follows:

$$\begin{aligned} \textbf{Min} \quad & TC(S) = \sum_{k=1}^{K} RC_{r_k} \\ \textbf{s.t.} \quad & \sum_{k=1}^{K} l_k = N_t \\ & t_{k_1,i_1} \neq t_{k_2,i_2}, \forall (k_1, i_1) \neq (k_2, i_2) \\ & \sum_{i=1}^{l_k} dm(t_{k,i}) \leq q_k, \forall k \in \{1, 2, ...N_{ov}\} \\ & \sum_{i=1}^{l_k} dm(t_{k,i}) \leq Q, \forall k \in \{N_{ov}+1, ..., K\} \end{aligned} \tag{1}$$

where N_t is the number of tasks and RC_{r_k} denote the total cost of route r_k and is computed according to Eq. 2:

$$\begin{aligned} RC_{r_k} = & \, mdc(v_k, tail_{t_{k,1}}) + mdc(head_{t_{k,l_k}}, v_0) + \\ & \sum_{i=1}^{l_k} mdc(head_{t_{k,i}}, tail_{t_{k,i+1}}) + \sum_{i=1}^{l_k} sc(t_{k,1}) \end{aligned} \tag{2}$$

where $head_t$, $tail_t$ denotes the head and tail vertices of the task, and $mdc(v_i, v_j)$ denotes the minimal total deadheading cost traversing from node v_i to node v_j. The first two constraints in Eq. (1) guarantee that all tasks are served only once and the other two constraints are formulated to satisfy the capacity constraint.

B. Simulation System for DCARP

In order to test optimisation algorithms for DCARP, a simulation system that includes some common dynamic events is required. Even though a benchmark generator for DCARP has been proposed by Liu et al. [21], it contains shortcomings, which prevent it to be used as research platform. Intuitively, a DCARP instance should be generated during a solution's execution and from the dynamic change of a previous DCARP instance, such as road congestion or recovering from the congestion. However, these essential details are not considered in the existing benchmark generator. Besides, the existing benchmark generator generates the static CARP instance by their own policies instead of using the existing or real maps, which is not suitable for a general research platform for different researchers. Therefore, we designed a simulation system which includes eight commonly occurring events and generates the DCARP instances from the existing static CARP benchmark [2]. Nine events and their corresponding mathematical forms with their probabilities of occurrence, are listed in Table II, and the simulation system is presented in Fig. 1.

TABLE II
TYPES OF DYNAMIC EVENTS AND THEIR MATHEMATICAL FORMS FOR DCARP. THE EVENTS WITH * ARE NEW EVENTS CONSIDERED IN THIS PAPER.

Event types	Changes
1. Vehicle break down	$d_k = 0 \rightarrow d_k > 0$
2. Road closure	$c_{ij} < Inf \rightarrow c_{ij} = Inf$
3. Congestion	$c_{ij} \rightarrow c_{ij}^{'}$, where $c_{ij}^{'} > c_{ij}$
4. Recover from roads closure *	$c_{ij} = Inf \rightarrow c_{ij} < Inf$
5. Recover from congestion *	$c_{ij} \rightarrow c_{ij}^{'}$, where $c_{ij}^{'} < c_{ij}$
6. Congestion become worse *	$c_{ij} \rightarrow c_{ij}^{'}$, where $c_{ij}^{'} > c_{ij}$
7. Congestion become better *	$c_{ij} \rightarrow c_{ij}^{'}$, where $c_{ij}^{'} < c_{ij}$
8. Demand increases	$d_{ij} \rightarrow d_{ij}^{'}$, where $d_{ij}^{'} > d_{ij}$
9. Added tasks	$d_{ij} = 0 \rightarrow d_{ij} > 0$

In order to make the simulator similar to the real world to the greatest extend, we have added several dynamic events, which have not been considered in the literature so far [20]. For example, the road can recover from a closure or a road

[2]Github link to be added after review process.

congestion, which has been marked with a star (*) in Table II. For example, if a vehicle break down occurs we can assume that this vehicle k already served d_k loads. Consequently, the demand of the edge, where the break down happened, increases from 0 to d_k to include the additional loads. Nevertheless, this event usually rarely happens in the real world. From all summarised mathematical forms, we can easily find that all dynamic events impact the cost or demand of edges. Therefore, we designed the *cost changer* and *demand changer* in our simulation system to simulate these events (Figure 1).

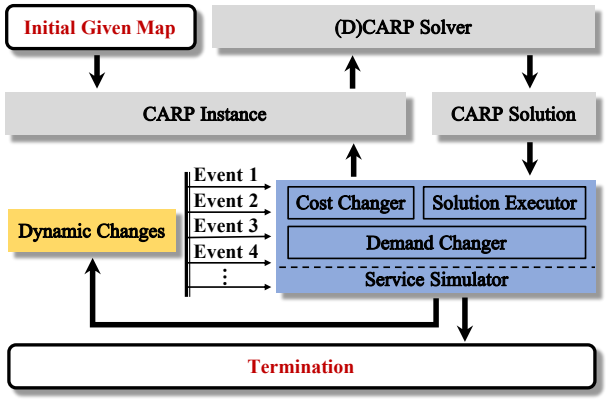

Fig. 1. The structure of our simulation system.

In Figure 1, the system starts from a provided initial graph, which could be taken from existing benchmarks for static CARP. Then, any CARP solver, such as memetic algorithms [6], can be used to obtain the initial solution, i.e. a first schedule for the vehicles. The core part of the system is the *Service Simulator* in Figure 1, which is used to execute the CARP solution and update the graph. The pseudocode is presented in Algorithm 1. During the execution of the CARP solution, the maximum number of vehicles in the depot is considered. If the number of routes in the schedule exceeds the predefined maximum number of vehicles, the route with the smallest cost will be served first and the remaining routes will be served after some vehicles return to the depot. When the solution is executed, some dynamic changes will happen and influence the graph at random points in time.

Once the dynamic change happens, the service simulator stops the execution of the current solution, and the *cost changer* and *demand changer* will update the DCARP instance. First, as broken down vehicles influence only a specific edge, we simulate Event 1 separately from other dynamic events. The algorithm selects n vehicles randomly to break down, and updates the graph accordingly, as shown in Lines 4-5. Events 2 to 7 will influence the cost of several edges so that the cost changer mainly simulates these five events, as shown in Lines 7-23. Each edge e_i has a property, $e_i.change$, recording whether it is currently in a changed state. If it is not, Event 2 or 3 happen in the edge depending on probabilities p_{event} and p_{road}. If $e_i.change == 2$, the road has broken down before so that it recovers with a probability p_{event}. If $e_i.change == 3$, the road is in congestion. It may either totally recover with a probability p_{crr}, or the traffic jam may ease or get worse (by a random cost) with a probability p_{crbb} or $1 - p_{crbb}$, respectively. Compared to Events 2 to 7, Events 8 and 9 are much easier to implement, because the demand of tasks can only increase. Therefore, the demand changer is very simple, as shown in Lines 25-29. Event 8 may happen to a task with a probability p_{icd}, increasing the demand by a random amount. For edges with no demand, Event 9 will happen with a probability p_{add}.

Algorithm 1: The pseudo code of the service simulator

Input: Executable solution s, Time of change: τ, Previous graph G

1 Set: $\{n, p_{event}, p_{road}, p_{bdrr}, p_{crr}, p_{crbb}, p_{icd}, p_{add}\}$;
2 Determine the stopping point for each vehicle according to s, τ, G;
3 Update graph, and remove all served tasks.
4 **Event 1** Randomly select n vehicles to break down.
5 Update the graph.
6 **/**** Cost Changer ****/**
7 **for** *each edge* e_i **do**
8 **if** $e_i.change == 0$ *and* $rand() > p_{event}$ **then**
9 **if** $rand() < p_{road}$ **then**
10 **Event 2** happens: $e_i.cost = Inf$, $e_i.change = 2$
11 **else**
12 **Event 3** happens: Increase cost of e_i, $e_i.change = 3$
13 continue;
14 **else if** $e_i.change == 2$ *and* $rand() > p_{bdrr}$ **then**
15 **Event 4** Recover edge e_i, $e_i.change == 0$;
16 **else if** $e_i.change == 3$ *and* $rand() > p_{event}$ **then**
17 $r = rand()$
18 **if** $r < p_{crr}$ **then**
19 **Event 5** Recover edge e_i, $e_i.change == 0$
20 **else if** $r < p_{crbb}$ **then**
21 **Event 6** Decrease cost of e_i
22 **else**
23 **Event 7** Increase cost of e_i
24 **/**** Demand Changer ****/**
25 **for** *each edge* e_i **do**
26 **if** $Edge.demand > 0$ *and* $rand() < p_{icd}$ **then**
27 **Event 8** happens: $e_i.change = 8$
28 **if** $Edge.demand == 0$ *and* $rand() < p_{add}$ **then**
29 **Event 9** happens: $e_i.change = 9$

Output: The new graph Gx

Finally, we will get a new DCARP instance to be optimised, and the solver generates a new DCARP solution. The system terminates after all tasks are served.

IV. A Generalised Optimisation Framework for DCARP

In this section, we propose a virtual-task strategy to change a DCARP instance to a 'virtual static' instance. After that, a generalised optimisation framework based on a virtual-task

strategy for DCARP with two different initialisation strategies is proposed, which allows to apply all algorithms for static CARP to solve DCARP.

A. Virtual task

As discussed in the previous section, the main challenge of scheduling vehicles for DCARP by using algorithms designed for static CARP is to take the outside vehicles with different locations and remaining capacities into account. We propose a virtual task strategy that forces all outside vehicles to virtually return to the depot such that all vehicles locate at the depot. In this way, algorithms for static CARP, which assume that all vehicles start at the depot, can be adopted. However, despite virtually returning to the depot, the outside vehicles are still required to start from the stop location when executing the new schedule after a change, so that these virtual returned vehicles have to first virtually move to their stop location in the new schedule. Therefore, the vehicles must serve some virtual paths in the new schedule to reach this stop location. These virtual paths can be regarded as virtual tasks being optimised along with the normal tasks by a static CARP algorithm. We also need all vehicles in the depot to have the same full capacities to be able to use static CARP algorithms. Therefore, we assign the previous demands that have been served by an outside vehicle to the corresponding virtual task. As a result, a DCARP instance is converted to a 'static' CARP instance, in which all vehicles are located at the depot with the same capacities. The pseudocode of constructing the virtual task is presented in Algorithm 2.

Algorithm 2: Pseudocode of constructing virtual tasks

Input: Task set $R = \{t_1, t_2, ..., t_{N_t}\}$,
Stop locations of outside vehicles: OV,
Remaining capacity of outside vehicles: RQ.
$OV = \{v_1, v_2, ..., v_{N_{ov}}\}$,
$RQ = \{q_1, q_2, ..., q_{N_{ov}}\}$

for *each outside vehicle* k **do**
- Construct an Arc with virtual task vt_k;
- Head: $vt_k.head$ = v_0, where v_0 is the depot;
- Tail: $vt_k.tail$ = v_k;
- Deadheading cost: $vt_k.dc$ = Inf;
- Serving cost: $vt_k.sc$ = $mdc(v_0, v_k)$;
- Demand: $vt_k.dm$ = $Q - q_k$, where Q is the original capacity of vehicles;
- Add this virtual task into task set: $R = R \cup vt_k$.

Output: The updated task set: R

A virtual task can also be regarded as a representation of an outside vehicle's previous serving process, including the total cost, served demand and stop location before the occurrence of the dynamic change. During the optimisation, the virtual tasks are regarded as arcs to be assigned to routes when being rescheduled. These arcs need to be served, so that some depot vehicles will actually correspond to the outside vehicles. Once a depot vehicle serves a virtual task, its remaining capacity will become the same as the remaining capacity of the corresponding outside vehicle, and so will its stop location. However, vehicles that are not serving these virtual arcs should be unable to traverse them, because these virtual arcs are not actual physical paths that can be used by vehicles. This is achieved by assigning a traversing cost of $dc = Inf$ to these arcs. Note that this infinite traversing cost is not included as part of the serving cost as it normally would for normal tasks.

The serving cost of a virtual task is zero. This is because in reality, the outside vehicles are already in the stop locations and have already served some tasks. They should not incur any extra cost to get there again, because this cost was already taken into account in the previous DCARP instance optimisation process. However, in our strategy, a virtual task's serving cost is set as the distance between between the depot and the vehicle stop location, because some algorithms such as the five rules in Path-Scanning use this cost as a denominator, e.g., when deciding which task to assign to the current route [4]. To avoid this cost being counted towards the total cost in the objective function, the additional cost will be subtracted from the actual total cost during the optimisation. Besides, the demand is set as the amount corresponding to the demand already served by the vehicle, i.e. $Q - q_k$, to avoid the total demand of tasks in its new route exceeding the vehicle's remaining capacity q_k.

In order to improve the understanding of how virtual tasks in DCARP instances are constructed, an example is provided in Fig. 2. One vehicle traverses from v_0 (depot) and serves the task (v_0, v_1). When the dynamic change happens, the vehicle is located at v_2, and then a virtual arc task is constructed between v_0 and v_2. After that, the new DCARP instance with the remaining task (v_3, v_4) and the virtual task $< v_0, v_2 >$ will be optimised using one of the algorithm which are available for static CARP, in which the task (v_0, v_1) is removed because it has been served. We allow the static CARP algorithm to place virtual tasks in positions that may not be the first positions of a given route. This will be fixed later on when converting a solution provided by the algorithm to an executable solution, as explained at the end of this section.

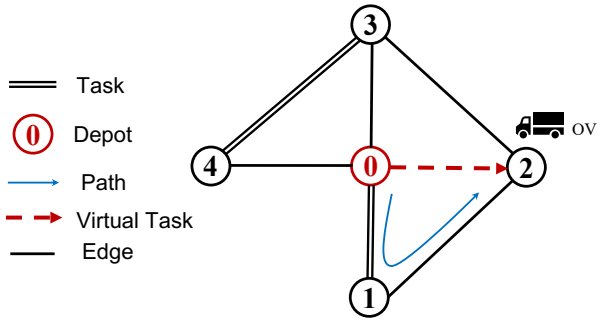

Fig. 2. An example of constructing virtual tasks.

After the conversion based on using the virtual-task strategy, the route formulation in a DCARP solution $S = \{r_1, r_2, ..., r_{N_{ov}}, ..., r_K\}$ becomes:

$$r_k = \{v_0, vt_k, t_{k,2}, t_{k,3}, ..., t_{k,l_k}, v_0\}, k = 1, 2, ..., N_{ov}.$$
$$r_k = \{v_0, t_{k,1}, t_{k,2}, ..., t_{k,l_k}, v_0\}, k = N_{ov} + 1, ..., K.$$

The new formulation of the optimisation objective for a DCARP instance is given in Eq. 3:

$$\begin{aligned} \textbf{Min} \quad & TC(S) = \sum_{k=1}^{K} RC_{r_k} - \sum_{k=1}^{N_{ov}} mdc(v_0, v_k). \\ \textbf{s.t.} \quad & \sum_{k=1}^{K} l_k = N_t + N_{ov}. \\ & t_{k_1,i_1} \neq t_{k_2,i_2}, \forall (k_1,i_1) \neq (k_2,i_2). \\ & \sum_{i=1}^{l_k} dm(t_{k,i}) \leq Q, \forall k = 1, ..., K. \end{aligned} \tag{3}$$

where for route $\{r_k | k = 1, 2, ..., N_{ov}\}$, $t_{k,1} = vt_k$.

These adjustments enable a new schedule for the converted 'static' CARP instance to be obtained by directly using meta-heuristic algorithms for static CARP. An executable solution is obtained by removing the virtual tasks from the service routes in the new schedule and assigning it to the corresponding outside vehicles. If a given virtual task is not the first task of a service route found by the static CARP algorithm, the route will be split into two routes with one route served by a vehicle from the depot and another served by the corresponding outside vehicle. However, the total cost of the CARP solution will not be influenced by splitting the routes or not. An example of converting an obtained new solution to an executable service plan is provided below:

$$\begin{gathered} \{\mathbf{v_0}, vt_1, t_2, \mathbf{v_0}, t_3, t_4, vt_5, t_6, \mathbf{v_0}\} \\ \downarrow \\ \{\mathbf{v_0}, vt_1, t_2, \mathbf{v_0}, t_3, t_4, \mathbf{v_0}, vt_5, t_6, \mathbf{v_0}\}. \end{gathered}$$

In the above example, the solution is specified two vehicles, one which is outside the depot and one which is located at the depot. When converting the solution to an executable plan, it increases to three vehicles, two which are outside the depot and one which is located at the depot.

B. Proposed Framework based on Virtual-Task Strategy

There are generally two commonly used strategies in dealing with dynamic optimisation. One is to One is to re-start the optimization which is potentially extended with some additional diversity enhancing techniques [32]. The other is to migrate some good solutions in the old environment to the new environment and initialize the starting individuals with them. [33], [34].

These two strategies can also be used in the optimization framework of a DCARP scenario. The restart strategy is easy to be applied for a new DCARP instance after a change occurred. However, the current strategies for using search history are not suitable for DCARP scenarios because they are all designed for the dynamic scenario where the condition of the objective remains the same after the dynamic events. The dynamic events influence the problem's dimension and the previous solution's feasibility in a DCARP scenario. For example, the served tasks and added tasks change the total number of tasks, and the potential road closure event makes the previous solution infeasible in the new DCARP instance.

Therefore, we propose a new sequence transfer strategy for the DCARP scenario (Algorithm 3), which can benefit from the previous DCARP instance's best solution. Given an instance I_m, all scheduled routes in the best solution S_{m-1} of the previous instance I_{m-1} are concatenated to construct an ordered list of tasks P_{m-1}. All tasks that have been served are then removed from the list in P_{m-1}, and the remaining tasks keep their orders. After that, the newly added tasks are inserted into P_{m-1} greedily, i.e., they are inserted into the positions with the smallest increased cost. Finally, a corresponding transferred solution is generated by using the split scheme to convert the ordered list to an explicit-route solution. The newly transferred solution is used as one of the initial solutions when optimising the new DCARP instance for the population-based algorithm and replaces the original initial solution if applied to an individual-based algorithm.

Algorithm 3: Pseudocode of sequence transfer strategy

```
Input: Previous best solution S_{m-1}
Convert S_{m-1} into an ordered list of tasks P_{m-1};
Remove all served tasks from P_{m-1};
P_m = P_{m-1}, l_P = |P_m|;
The newly added tasks set
  NT = {nt_1, nt_2, ..., nt_{N_NT}};
for each nt_i ∈ NT do
    for each position in P_m do
        Calculate the increased cost after insertion;
    Obtain the position p with the smallest increased
      cost;
    Insertion: P_m = [t_1, ..., t_p, nt_i, t_{p+1}, ..., t_{l_P}];
    l_P = l_P + 1;
    NT = NT \ {nt_i}.
Use split scheme in P_m.
Output: A newly transferred solution S_m
```

On the basis of the virtual-task strategy and two initialisation strategies, i.e. the restart strategy and the sequence transfer strategy, a generalized optimization framework with virtual tasks (GOFVT) is proposed to generalize static CARP algorithms to dynamic scenarios. There are four main steps in this framework:

1) Construct virtual tasks
2) Apply the restart strategy (randomly generate initial solutions) or the sequence transfer strategy (generate one transferred solution for individual-based algorithms and additionally generated random solutions for population-based algorithms)
3) Apply the meta-heuristic algorithm to optimize the converted 'static' CARP instance
4) Convert the obtained solution with virtual tasks to an executable solution

For a DCARP instance, the framework will first construct the virtual tasks to convert the dynamic instance to a 'static' instance. Then, one of the two initialization strategies explained above can be adopted to assist the optimisation of the DCARP instances. In the computational studies in Section

V, we will compare the effectiveness of these two strategies. Then, meta-heuristic algorithms with the initialisation strategy are applied to optimize the DCARP instance as a 'static' instance with virtual tasks. Finally, the solution obtained for the 'static' instance is converted into an executable solution, in which the routes with virtual tasks are assigned to the corresponding outside vehicles and the routes without virtual tasks are assigned to vehicles located at the depot.

V. Computational Studies

In order to evaluate the efficiency of our proposed framework (GOFVT), three sets of experimental studies have been conducted by embedding a selection of meta-heuristic algorithms in the GOFVT in this section. In the first experiment, the virtual-task strategy is compared with a simple rescheduling strategy. After that, the virtual-task strategy's efficiency is investigated by comparing it with an existing algorithm for DCARP. Finally, GOFVT is combined with several classical meta-heuristic algorithms originally designed for static CARP, and its performance is analysed by running the newly generated algorithms in DCARP scenarios.

A. Experimental Settings

All experiments are conducted on a series of DCARP instances or scenarios generated by the simulation system presented in Section III-B, based on a static CARP benchmark, namely the *egl* set [35]. The *egl* set contains 24 CARP instances. In each experiment, the DCARP instances or scenarios are generated independently from static CARP instances based on our simulator. For our first and second set of experiments, 3 DCARP instances are generated corresponding to each CARP instance. These DCARP instances are not used to compose a DCARP scenario, as the algorithms just optimise the current instance and never benefit from a history instance in these two experiments. For the third, one DCARP scenario including 5 DCARP instances is generated based on each CARP instance. When the simulator executes a solution according to the deployment policy, the time for serving all tasks will be calculated first and the simulator will uniform randomly select a stop point within the longest time. The parameters in the simulator are set as $n = 0$, $p_{event} = 0.5$, $p_{road} = 0.1$, $p_{bdrr} = 0.5$, $p_{crr} = 0.3$, $p_{crbb} = 0.6$, $p_{cid} = 0.35$, $p_{add} = 0.35$. Eight different types of dynamic changes (Table II) are simulated in the simulator excluding the case of "vehicles broken down" because its formulation is the same as the event of added tasks.

Because a key optimization requirement when dynamic changes happen in the real world is to obtain a new solution quickly, we limit the maximum optimisation time to 60s for the small problems (E1 $\sim$ E4) and 180s for larger maps (S1 $\sim$ S4) in *egl* for all algorithms. The source code of our experiments is available on github [3].

B. Is It Necessary to Reschedule for DCARP?

Although algorithms have been proposed to solve DCARP in the literature, a simple baseline strategy, named the return-first strategy, has been ignored. The return-first strategy schedules all outside vehicles back to the depot first in order to convert a DCARP instance to a static one, and then reschedules all vehicles in a new static instance after all vehicles are located at the depot. If the return-first strategy is efficient enough, the direct optimisation of a DCARP instance would not be necessary any more. However, there was no such demonstration in the literature. Therefore, in this subsection, we use the proposed virtual-task strategy to solve DCARP and compare it with the return-first strategy to show the importance of optimizing DCARP instances directly instead of ignoring the outside vehicles and assigning new vehicles to all remaining tasks.

In our experiment, the simulation system generates three different DCARP instances for each test map with a different set of remaining capacities, i.e. $[0, 0.33Q]$, $[0.34Q, 0.66Q]$ and $[0.67Q, Q]$. Then, an optimisation algorithm, Memetic Algorithm with Extended Neighborhood Search (MAENS) [6], assisted with the return-first strategy and virtual-task strategy is applied to optimise each DCARP instance. In terms of mean and standard deviation over 25 independent runs (mean$\pm$std), the comparison results of the return-first (RF) strategy and virtual-task (VT) strategy on DCARP instances with different remaining capacities are presented in Table III. The bold values with grey background for each DCARP instance are the better results between return-first strategy and virtual-task strategy based on the Wilcoxon signed-rank test with the level of significance of 0.05. The second last row of Table III summarises the number of win-draw-lose of RF strategy versus VT strategy. Besides, we calculated the Wilcoxon signed-rank test with the level of significance 0.05 for the mean total cost of RF strategy and VT strategy on the instances with the same range of remaining capacities, and the p-values are listed in the last row of Table III.

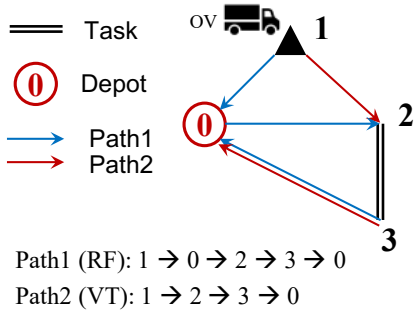

Fig. 3. An example of demonstrating why the RF strategy is not efficient enough when outside vehicles have enough remaining capacities.

Table III shows that the RF strategy and VT strategy are significantly different for the instances where the remaining capacities are in the range of $[0.34Q, Q]$ (instances 2 and 3 in Table III), with the VT strategy outperforming the RF strategy on all DCARP instances. In contrast, for the scenarios with remaining capacities in $[0, 0.33Q]$, there are 12 out of 24 DCARP instances where the RF strategy outperforms the VT strategy, while VT strategy is still better than RF strategy in 9 out of 24 instances, and there are also three instances where

[3]Github link will be added after review.

TABLE III
RESULTS OF VIRTUAL-TASK STRATEGY (VT) AND RETURN-FIRST STRATEGY (RF) ON DCARP INSTANCES WITH DIFFERENT SETTINGS OF REMAINING CAPACITIES FROM THE *egl* DATASET. THE VALUE IN EACH CELL REPRESENTS "MEAN ± STD" OVER 25 INDEPENDENT RUNS AND THE BOLD ONES DENOTE THE BETTER RESULT ON THE DCARP INSTANCE BASED ON THE WILCOXON SIGNED-RANK TEST WITH THE LEVEL OF SIGNIFICANCE 0.05. THE PENULTIMATE ROW SUMMARIES THE NUMBER OF WIN-DRAW-LOSE OF RF STRATEGY VERSUS VT STRATEGY AND THE LAST ROW PROVIDES THE P-VALUES OF THE WILCOXON SIGNED-RANK TEST WITH THE LEVEL OF SIGNIFICANCE 0.05 ON INSTANCES WITH THE SAME SETTINGS OF ALL MAPS,

Map	Instance 1 ($q \in [0, 0.33Q]$)		Instance 2 ($q \in [0.34Q, 0.66Q]$)		Instance 3 ($q \in [0.67Q, Q]$)	
	RF	VT	RF	VT	RF	VT
E1-A	8386±22	***8095±31***	10397±34	***9925±22***	8630±8	***8566±10***
E1-B	***9830±29***	***9839±39***	9718±22	***8437±28***	9675±16	***9208±29***
E1-C	***12181±52***	***12109±52***	14726±31	***14106±16***	11286±11	***10565±25***
E2-A	***10255±6***	10480±6	13740±27	***12278±29***	11692±51	***10231±34***
E2-B	***13162±15***	13317±23	16435±29	***15207±18***	15667±35	***14321±55***
E2-C	***17425±41***	17566±35	17444±32	***16881±25***	14186±16	***13871±46***
E3-A	11343±42	***11167±17***	12877±21	***12030±26***	10369±19	***8869±12***
E3-B	***15517±84***	15776±31	15150±31	***12797±68***	20743±59	***20081±39***
E3-C	***20549±69***	20880±63	20012±144	***18078±88***	27521±81	***25289±82***
E4-A	***11216±31***	11387±27	12489±26	***12152±32***	16213±29	***14858±135***
E4-B	15522±40	***15242±62***	14933±65	***13791±63***	17477±71	***15869±29***
E4-C	20928±63	***20691±41***	21998±28	***20174±86***	17869±79	***15798±59***
S1-A	16097±49	***16012±40***	15643±31	***14917±40***	13684±53	***13159±18***
S1-B	***20462±69***	21167±58	16184±35	***15697±35***	15151±19	***13428±82***
S1-C	***27944±71***	28624±116	21040±36	***18616±87***	26470±103	***25789±76***
S2-A	19610±85	***19479±107***	22314±40	***20016±48***	22499±67	***19212±82***
S2-B	***27970±75***	28408±59	26334±130	***25911±118***	25723±67	***25024±94***
S2-C	***31859±108***	32101±125	40170±132	***38575±186***	35053±143	***31815±133***
S3-A	19489±64	***19211±107***	23630±118	***21329±98***	28237±76	***24494±71***
S3-B	***27518±72***	***27521±98***	25651±64	***24311±93***	32286±50	***30646±96***
S3-C	32468±93	***32049±107***	35435±67	***33927±68***	43151±98	***40801±97***
S4-A	***27256±87***	28082±142	23780±74	***22788±95***	29895±102	***27576±105***
S4-B	34872±122	***34709±149***	28283±121	***28023±93***	31770±87	***30652±125***
S4-C	***43746±168***	44165±157	36808±117	***36706±123***	44006±145	***42827±178***
#of 'w-d-l'	12-3-9		0-0-24		0-0-24	
p-value	0.22		1.82e-5		1.82e-5	

the two strategies obtain comparable results. When comparing the results using the Wilcoxon test across maps, we confirm that none of these strategies is a winner when analysed across maps where the remaining capacities are smaller than $0.33Q$. This is understandable because when the vehicles are mostly full, i.e., when the remaining capacities are smaller than 0.33Q, there is limited room for serving more tasks no matter what strategy is used.

Overall, we can conclude that it is necessary and much more effective to optimise the DCARP instance directly rather than using the RF strategy when outside vehicles have enough remaining capacities.

The reason why the RF strategy is not always helpful when outside vehicles have enough remaining capacities can be explained using a simple example following Fiure 3, where an outside vehicle stops at vertex 1. If its remaining capacity is sufficient to serve task t_{23}, it can directly traverse from vertex 1 to vertex 2, presented as 'Path 2' in Figure 3, and the final total cost will be $d_{vt} = d_{12} + d_{23} + d_{30}$. But if we apply the RF strategy, the total cost will change to $d_{rf} = d_{10}+d_{02}+d_{23}+d_{30}$, presented as 'Path 1' in Figure 3. It is obvious that $d_{rf} \geq d_{vt}$ because of $d_{10}+d_{02} \geq d_{12}$. The RF strategy increases the final cost because vehicles take a detour in such scenarios. Therefore, when outside vehicles have large remaining capacities after dynamic events, assigning them to continue to serve other tasks is much more efficient in terms of the total cost.

C. Analysis of the Effects of the Virtual-Task Strategy

Memetic Algorithm with new Split scheme for DCARP (MASDC) [20] is the only in the literature that considers a general DCARP scenario including several common dynamic events, such as road closure and added tasks. It comprises a distance-based split scheme to assist the DCARP solution being used in the crossover and local search. Our virtual-task strategy can convert a DCARP instance to a 'static'

TABLE IV
RESULTS OF MASDC AND VT-MASDC ON DCARP INSTANCES FROM THE *egl* DATASET. THE VALUE IN EACH CELL REPRESENTS "MEAN ± STD" OVER 25 INDEPENDENT RUNS AND THE BOLD ONES DENOTE THE BETTER RESULT ON THE DCARP INSTANCE BASED ON THE WILCOXON SIGNED-RANK TEST WITH THE LEVEL OF SIGNIFICANCE 0.05. THE LAST ROW SUMMARIES THE NUMBER OF WIN-DRAW-LOSE OF MASDC VERSUS VT-MASDC.

Map	Ins	MASDC	VT-MASDC
E1-A	1	19354±748	***16180±1716***
	2	15074±529	***12551±1584***
	3	18528±724	***15927±1946***
E1-B	1	20644±783	***17229±1644***
	2	18052±716	***15889±1394***
	3	18947±564	***15691±1664***
E1-C	1	22591±754	***19104±1379***
	2	18778±556	***16192±1444***
	3	23001±1039	***18119±1755***
E2-A	1	23673±917	***20981±1881***
	2	19463±762	***16235±2155***
	3	18319±697	***15843±1800***
E2-B	1	25745±869	***21650±2151***
	2	23281±816	***20146±1744***
	3	23517±941	***19071±2264***
E2-C	1	29112±1014	***24493±2035***
	2	23363±682	***20451±1624***
	3	27457±896	***24353±1883***
E3-A	1	24838±1030	***20851±1867***
	2	25274±1011	***21138±2332***
	3	26588±1254	***22012±2441***
E3-B	1	27026±893	***22399±2364***
	2	27106±812	***21950±2649***
	3	23861±833	***20398±2059***
E3-C	1	31574±942	***27110±2354***
	2	32775±1018	***28566±1542***
	3	31589±679	***28282±1855***
E4-A	1	20835±854	***17411±1440***
	2	23469±939	***20619±1781***
	3	25219±896	***22005±2083***
E4-B	1	23648±946	***20050±1708***
	2	26950±779	***23664±2491***
	3	25755±844	***21767±2165***
E4-C	1	28338±812	***25212±1430***
	2	30586±708	***26925±1560***
	3	28152±762	***24482±1979***

Map	Ins	MASDC	VT-MASDC
S1-A	1	35122±1318	***28122±2707***
	2	37189±1362	***32033±2467***
	3	40972±1185	***34348±2529***
S1-B	1	38612±1196	***33192±2775***
	2	36846±1292	***32335±2240***
	3	37521±1153	***32815±2734***
S1-C	1	42269±1540	***37935±2671***
	2	36263±971	***31978±2606***
	3	48458±1389	***43726±2405***
S2-A	1	57452±1384	***51503±4217***
	2	58765±1622	***52707±3584***
	3	45582±1390	***40030±2801***
S2-B	1	64514±2001	***54667±3991***
	2	58803±1833	***53834±2675***
	3	56169±1253	***48964±2281***
S2-C	1	65181±1799	***58049±2561***
	2	72010±1674	***64127±3558***
	3	75819±2091	***69322±2828***
S3-A	1	49937±1061	***43826±3168***
	2	45992±1571	***39608±3019***
	3	43552±981	***36280±2977***
S3-B	1	56644±1736	***51214±2533***
	2	61856±1440	***53962±3334***
	3	66870±1877	***57878±3772***
S3-C	1	62193±1778	***56232±3060***
	2	69501±1601	***62797±3424***
	3	79699±1891	***71613±3912***
S4-A	1	58456±1621	***51512±3298***
	2	55394±1746	***49422±3517***
	3	64795±1546	***58676±3517***
S4-B	1	64343±1886	***57017±3622***
	2	65027±1712	***57413±3033***
	3	60143±1612	***53342±3323***
S4-C	1	69224±1475	***62760±2929***
	2	68070±1424	***62304±2465***
	3	78134±1893	***69348±3451***
# of w-d-l			0-0-72
p-value			1.67e-13

CARP instance so that the operator used in the static CARP can be used in the DCARP instance directly without any specific operator. In this subsection, we analyse the effects of VT strategy by embedding it to MASDC, referred to as VT-MASDC, and comparing it to the original MASDC. The advantage of embedding the strategy into MASDC is that this enables us to isolate and analyse the effect of the virtual tasks compared to a state-of-the-art DCARP algorithm. In particular, all components in MASDC and VT-MASDC are the same except for the use of virtual tasks and the distance-based split scheme, which needs to be replaced in VT-MASDC as the instances in VT-MASDC become 'static'. Therefore, the distance-based split scheme is replaced by Ulusoy's split scheme in VT-MASDC. The use of virtual tasks is thus inherently linked to this split scheme, and any advantages provided by the virtual tasks are also linked to the fact that they enable this split scheme to be adopted.

In our experiment, we generate three independent DCARP instances for each map, ensuring that outside vehicles had enough remaining capacities, i.e. $q \geq 0.5Q$, in all generated instances. However, we need to generate DCARP instances rather than DCARP scenarios because the aim of DCARP is to minimise the total cost for each DCARP instance separately (Eq. 2). Therefore, the performance comparison can be based

on the DARP instances. The optimisation results are presented in Table IV, in which the values in each cell represent the mean and standard deviation over 25 independent runs (mean±std). For each DCARP instance, the better result, based on the Wilcoxon signed-rank test with the level of significance 0.05, is highlighted using a bold font and grey background in Table IV. The summary of win-draw-lose of MASDC versus VT-MASDC which is presented in the last row, shows that VT-MASDC outperforms MASDC on all generated DCARP instances. The Wilcoxon signed-rank test with the level of significance 0.05 was conducted for the average total cost of VT-MASDC and MASDC in all instances, and the p-value was 1.67e-13, which is lower than 0.05, indicating that VT-MASDC is significantly different from MASDC. Overall, we can conclude that VT-MASDC performs much better than the original MASDC.

MASDC uses a distance-based split scheme (DSS) to evaluate DCARP solution's fitness because the split scheme designed for static CARP is not suitable for DCARP [25]. The DSS operator randomly splits the sequence of tasks to an executable CARP solution with explicit routes, and the random splitting process is repeated three times to obtain a better schedule. After embedding with the VT strategy, the DSS operator is replaced by Ulusoy's split [3] as explained in the beginning of this section, and the evaluation of a sequence of tasks only requires to apply the Ulusoy's split once. As a result, the randomness brought by the DSS's split scheme is removed.

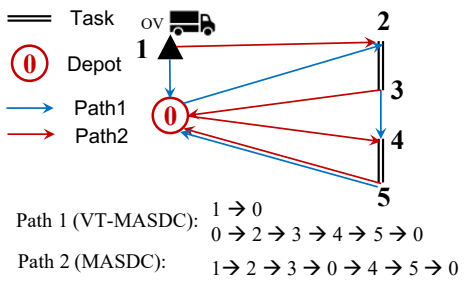

Fig. 4. An example of demonstrating the advantages of considering new vehicles.

Moreover, the DSS operator never considers new vehicles starting from the depot during the optimisation, whereas our VT strategy enables both outside and new vehicles to be used. We provide an example in Figure 4 to show the advantages of considering new vehicles during the optimisation for DCARP. An outside vehicle is located at vertex 1, and its remaining capacity can only serve task t_{23}. We assume that $dd_{10}+dd_{02} \approx dd_{12}$ and $dd_{03} + dd_{04} \gg dd_{34}$ and the total cost of 'Path1' and 'Path2' can be calculated as:

$$TC_1 = \underline{dd_{01} + dd_{02}} + sc_{23} + \underline{dd_{34}} + sc_{45} + dd_{50},$$
$$TC_2 = \underline{dd_{12}} + sc_{23} + \underline{dd_{30} + dd_{04}} + sc_{45} + dd_{50}.$$

For a sequence of tasks $[0, t_{23}, t_{45}, 0]$, if applied with the DSS operator, the only obtained path will be 'Path 2' as shown in Figure 4 . In contrast, if we use the VT strategy, a better path, i.e. 'Path 1' in Figure 4, can be obtained, which avoids traversing the longer returning path.

D. Analysis of the Effects of GOFVT

The proposed optimisation framework GOFVT is capable of generalising all algorithms for static CARPs to optimise DCARP. To demonstrate its efficiency, we have selected three classical meta-heuristic algorithms for static CARPs, namely RTS [36], ILMA [37] and MAENS [6], and embedded them into the GOFVT in our experiments to evaluate whether it is advantageous to make use of existing static CARP algorithms within the GOFVT framework.

A brief description of each algorithm is presented below.

- ***RTS*** [36]: A global repair operator which is embedded in a tabu search algorithm (TSA [5]). The global repair operator is designed to repair the low-cost infeasible solutions to high-quality solutions. Source code is available on the website[4].
- ***ILMA*** [37]: An improved version of Lacomme's memetic algorithm (LMA) [1]. LMA is a memetic algorithm, combining the genetic algorithms and local search operator. ILMA considers the violation of capacity constraints during the local search, thus, improving the original LMA. For our experiments, we implemented ILMA[5] ourselves according to the details given in [37] because we are unable to obtain the source code.
- ***MAENS*** [6]: A memetic algorithm with a merge-split operator that helps local search to use a large step size and avoid being trapped in a local optimum. Source code is available on the website[6].

The newly generated algorithm instances for optimising DCARP are denoted by VT-RTS, VT-ILMA and VT-MAENS. We use these acronyms to denote GOFVT with the restart strategy. When using the sequence transfer strategy, they are denoted by VTtr-RTS, VTtr-ILMA, VTtr-MAENS.

The parameter settings as given in all algorithms follow the settings in their original papers [36], [6], [37]. In order to analyse the efficiency of the restart and sequence transfer initial strategies, a DCARP scenario consisting of 5 DCARP instances has been generated for each map in our experiments. A new DCARP instance is generated by executing the obtained best solution of the previous DCARP instance in our simulation system.

We have also embedded MASDC into GOFVT and generated two algorithm instances labelled as VT-MASDC and VTtr-MASDC for using the restart and sequence transfer strategies. The results of the above eight algorithm instances are presented in Table V. The values in each cell represent the 'mean±std' and the average ranking (in the brackets) in terms of the average total cost over 25 independent runs. The bold values represent the better result between restart strategy and sequence transfer strategy for each algorithm on a DCARP instance under the Wilcoxon signed-rank test with the level of significance 0.05. Rows where both restart strategy and sequence transfer strategy are in bold represent that these two strategies are not significantly different under the hypothesis test. The summaries of win-draw-lose of the restart strategy

[4]https://meiyi1986.github.io/publication/mei-2009-global/code.zip
[5]Github link will be added after review.
[6]https://meiyi1986.github.io/publication/tang-2009-memetic/code.zip

TABLE V

RESULTS OF VT-RTS, VTTR-RTS, VT-ILMA, VTTR-ILMA, VT-MAENS, VTTR-MAENS, VT-MASDC AND VTTR-MASDC ON THE *egl* DATASET. THE VALUES IN EACH CELL REPRESENT THE "MEAN±STD" WITH THE AVERAGE RANKING (IN THE BRACKETS) W.R.T THE AVERAGE TOTAL COST OVER 25 INDEPENDENT RUNS. THE BOLD VALUES ARE THE BETTER RESULTS UNDER THE WILCOXON SIGNED-RANK TEST WITH THE LEVEL OF SIGNIFICANCE 0.05 BETWEEN RESTART AND INHERITING STRATEGIES IN AN ALGORITHM FOR A DCARP INSTANCE. THE LAST TWO ROWS SUMMARISE THE NUMBER OF WIN-DRAW-LOSE OF RESTART STRATEGY VERSUS INHERITING STRATEGY IN EACH ALGORITHM AND THE P-VALUES OF THE WILCOXON SIGNED-RANK TEST WITH THE LEVEL OF SIGNIFICANCE 0.05 ON ALL INSTANCES.

Map	Ins	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
	1	10465±35(5.9)	***9608±427(5.1)***	***8882±88(3.3)***	***8886±96(3.1)***	***8738±54(1.8)***	***8744±60(1.8)***	***17398±1359(7.5)***	***17398±1359(7.5)***
	2	10801±0(6)	***9307±93(3.6)***	***9369±110(4.0)***	***9391±98(4.3)***	***9167±41(1.6)***	***9156±38(1.5)***	***18667±1820(7.5)***	***18667±1820(7.5)***
E1-A	3	11518±466(5.7)	***10491±169(4.9)***	***10309±121(3.7)***	***10292±107(3.7)***	***9920±79(1.4)***	***9917±95(1.6)***	***19505±2078(7.5)***	***19505±2078(7.5)***
	4	16543±0(6)	***12685±143(3.5)***	***12911±236(4.1)***	***12933±271(4.4)***	***12217±63(1.4)***	***12221±75(1.6)***	***29176±2612(7.5)***	***29176±2612(7.5)***
	5	15214±91(6)	***12856±162(4.4)***	***12782±198(4.0)***	***12731±204(3.6)***	***12055±89(1.5)***	***12056±77(1.5)***	***29298±2879(7.5)***	***29298±2879(7.5)***
	1	***11847±719(4.9)***	14106±186(6)	***10123±176(3.8)***	***10050±137(3.3)***	***9589±34(1.5)***	***9582±44(1.5)***	***19356±2043(7.5)***	***19356±2043(7.5)***
	2	***12769±10(5.5)***	***13398±1779(5.5)***	***10627±181(3.6)***	***10578±156(3.4)***	***10063±28(1.5)***	***10061±25(1.5)***	***19599±1581(7.5)***	***19599±1581(7.5)***
E1-B	3	18244±25(6)	***16828±607(5)***	***14229±212(3.6)***	***14179±235(3.4)***	***13659±50(1.4)***	***13667±46(1.6)***	***26484±2919(7.5)***	***26484±2919(7.5)***
	4	***16846±49(5.3)***	***17849±2528(5.4)***	***13928±214(3.6)***	***13949±193(3.8)***	***13334±26(1.7)***	***13321±23(1.3)***	***24979±1879(7.5)***	***24979±1879(7.5)***
	5	***17444±79(5.4)***	***17231±905(5.6)***	***14055±153(3.5)***	***14022±131(3.5)***	***13526±56(1.5)***	***13535±56(1.5)***	***27082±2155(7.5)***	***27082±2155(7.5)***
	1	***12899±652(5.2)***	13220±682(5.7)	***9983±171(3.6)***	***9953±152(3.6)***	***9604±46(1.5)***	***9594±36(1.5)***	***16584±1760(7.5)***	***16584±1760(7.5)***
	2	***17580±21(5.0)***	19788±1143(6.1)	***14403±85(3.6)***	***14431±164(3.5)***	***13908±63(1.6)***	***13880±55(1.4)***	***22521±1437(7.4)***	***22521±1437(7.4)***
E1-C	3	***18672±88(5)***	20128±138(6)	***15851±200(3.4)***	***15846±124(3.6)***	***15147±56(1.5)***	***15143±68(1.5)***	***26301±2000(7.5)***	***26301±2000(7.5)***
	4	***18317±477(5.3)***	18630±197(5.7)	***14414±217(3.6)***	***14394±221(3.4)***	***13735±88(1.5)***	***13748±101(1.5)***	***27743±2191(7.5)***	***27743±2191(7.5)***
	5	***20998±138(5)***	22605±331(6)	***15932±189(3.6)***	***15894±230(3.4)***	***15333±9(1.4)***	***15336±11(1.6)***	***31064±2866(7.5)***	***31064±2866(7.5)***
	1	12049±72(6)	***10679±258(4.9)***	***10121±127(3.6)***	***10070±114(3.4)***	***9714±60(1.5)***	***9720±55(1.5)***	***18699±1742(7.5)***	***18699±1742(7.5)***
	2	14173±246(5.7)	***12743±1499(4.9)***	***11042±117(3.4)***	***11071±128(4.0)***	***10712±76(1.4)***	***10719±81(1.6)***	***21735±1707(7.5)***	***21735±1707(7.5)***
E2-A	3	14358±46(6.0)	***12196±1489(5.0)***	***10688±186(3.6)***	***10654±150(3.4)***	***10287±23(1.4)***	***10294±30(1.6)***	***23440±2822(7.5)***	***23440±2822(7.5)***
	4	10803±793(5.3)	***9898±122(3.2)***	***10132±131(4.7)***	***10165±150(4.8)***	***9471±36(1.4)***	***9482±63(1.6)***	***20649±1990(7.5)***	***20649±1990(7.5)***
	5	***11814±443(4.4)***	13003±1148(5.9)	***11623±166(3.8)***	***11621±101(3.9)***	***11127±86(1.4)***	***11129±74(1.6)***	***23731±2454(7.5)***	***23731±2454(7.5)***
	1	***14822±24(5.3)***	15626±1340(5.7)	***12624±157(3.6)***	***12617±150(3.4)***	***12082±76(1.5)***	***12092±66(1.5)***	***22829±1803(7.5)***	***22829±1803(7.5)***
	2	***24926±39(5)***	31035±138(6.2)	***21828±208(3.6)***	***21860±218(3.4)***	***20966±27(1.5)***	***20973±32(1.5)***	***34624±2271(7.4)***	***34624±2271(7.4)***
E2-B	3	***22818±236(5.5)***	***23448±4504(5.4)***	***18715±212(3.6)***	***18681±179(3.5)***	***18055±80(1.6)***	***18064±75(1.4)***	***32244±2307(7.5)***	***32244±2307(7.5)***
	4	20693±23(6.0)	***18843±1502(4.2)***	***18533±282(3.9)***	***18491±199(3.9)***	***17754±9(1.6)***	***17748±17(1.4)***	***31178±1906(7.5)***	***31178±1906(7.5)***
	5	***23918±206(5)***	27676±276(6)	***21215±173(3.5)***	***21272±286(3.5)***	***20502±76(1.7)***	***20477±82(1.3)***	***35184±2828(7.5)***	***35184±2828(7.5)***
	1	***19158±39(5.1)***	19760±339(6.1)	***15655±128(3.3)***	15756±185(3.7)	***15163±66(1.6)***	***15164±57(1.4)***	***24052±2346(7.4)***	***24052±2346(7.4)***
	2	***19099±14(5)***	22019±140(6.1)	***16496±112(3.6)***	***16469±165(3.4)***	***15855±48(1.7)***	***15831±41(1.3)***	***25457±1845(7.5)***	***25457±1845(7.5)***
E2-C	3	21042±10(6)	***20168±16(5)***	***17479±155(3.4)***	***17521±199(3.6)***	***17225±32(1.6)***	***17213±35(1.4)***	***27856±2381(7.5)***	***27856±2381(7.5)***
	4	27475±278(6)	***24960±0(5)***	***21322±171(3.4)***	***21369±183(3.6)***	***20714±80(1.5)***	***20707±66(1.5)***	***32173±2287(7.5)***	***32173±2287(7.5)***
	5	29733±15(6)	***27727±521(5)***	***24182±288(3.6)***	***24168±258(3.4)***	***23416±68(1.5)***	***23398±61(1.5)***	***36975±2467(7.5)***	***36975±2467(7.5)***
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	1	***22989±156(3.6)***	***22971±190(3.4)***	***24100±307(5.5)***	***24073±289(5.5)***	***22411±61(1.5)***	***22401±70(1.5)***	***52327±3849(7.5)***	***52327±3849(7.5)***
	2	***25249±348(2.5)***	26012±192(4.0)	***27134±338(5.3)***	***27234±405(5.6)***	***25013±105(1.8)***	***25017±103(1.8)***	***57870±3931(7.5)***	***57870±3931(7.5)***
S3-A	3	***25907±350(3.3)***	26117±160(3.7)	***27011±250(5.5)***	***27042±320(5.5)***	***25285±145(1.4)***	***25349±113(1.6)***	***59652±3792(7.5)***	***59652±3792(7.5)***
	4	41928±177(6)	***33347±287(3)***	***34800±398(4.5)***	***34840±420(4.5)***	***32374±119(1.4)***	***32423±100(1.6)***	***72252±4788(7.5)***	***72250±4790(7.5)***
	5	31319±4(6)	***24329±316(2.7)***	***25614±322(4.6)***	***25632±334(4.4)***	***23989±74(1.7)***	***24006±85(1.6)***	***62362±4014(7.5)***	***62362±4014(7.5)***
	1	30945±70(6)	***29429±141(5)***	***26486±282(3.4)***	***26493±230(3.6)***	***25161±123(1.5)***	***25165±106(1.5)***	***46491±3247(7.5)***	***46491±3247(7.5)***
	2	***28494±1307(3.5)***	29198±1867(4.1)	***29318±356(5.2)***	***29374±347(5.2)***	27445±121(1.8)	***27356±140(1.2)***	***55642±2518(7.5)***	***55642±2518(7.5)***
S3-B	3	***36665±2093(5.3)***	***35529±3448(3.8)***	***34378±330(4.2)***	***34515±347(4.7)***	***32584±78(1.5)***	***32613±112(1.5)***	***65559±3265(7.5)***	***65559±3265(7.5)***
	4	36424±10(6)	***29498±240(3.0)***	***30262±280(4.6)***	***30306±339(4.4)***	***28983±133(1.6)***	***28930±131(1.4)***	***59740±4565(7.5)***	***59740±4565(7.5)***
	5	***32715±2377(4.0)***	***31524±348(3.7)***	***32271±345(5.2)***	***32185±417(5.1)***	***30444±167(1.4)***	***30437±160(1.6)***	***61165±3506(7.5)***	***61165±3506(7.5)***
	1	***45906±0(5.0)***	47919±1147(6.0)	***40932±333(3.4)***	***40930±425(3.6)***	***39026±146(1.4)***	***39049±154(1.6)***	***69270±3741(7.5)***	***69270±3741(7.5)***
	2	***58082±263(5.0)***	58770±195(6.0)	***51538±346(3.6)***	***51445±366(3.4)***	***49131±169(1.5)***	***49157±207(1.5)***	***78938±4233(7.5)***	***78938±4233(7.5)***
S3-C	3	***45802±74(5.2)***	45868±21(5.8)	***38727±341(3.4)***	38925±241(3.6)	***36791±118(1.6)***	***36730±94(1.4)***	***67602±3362(7.5)***	***67602±3362(7.5)***
	4	41011±1(6)	***40879±115(5)***	***35457±319(3.5)***	***35339±338(3.5)***	***33663±89(1.7)***	***33653±91(1.3)***	***61607±2949(7.5)***	***61607±2949(7.5)***
	5	***49309±114(5.5)***	***46793±3817(4.8)***	***42843±488(3.9)***	***42764±568(3.7)***	***40153±107(1.6)***	***40143±132(1.4)***	***72621±3858(7.5)***	***72621±3858(7.5)***
	1	24849±1470(6)	***21105±170(2.9)***	***22336±255(4.5)***	***22385±332(4.5)***	***20730±131(1.6)***	***20699±85(1.5)***	***53513±3603(7.5)***	***53513±3603(7.5)***
	2	***24693±1073(3.1)***	24947±322(4.0)	***26177±477(5.4)***	***26175±352(5.5)***	***23967±134(1.6)***	***23920±96(1.4)***	***59023±3019(7.5)***	***59020±3023(7.5)***
S4-A	3	28262±12(6)	***22203±286(3)***	***23181±356(4.4)***	***23091±319(4.5)***	***21661±67(1.6)***	***21638±68(1.4)***	***49301±3606(7.5)***	***49325±3575(7.5)***
	4	28294±261(4.6)	***27786±653(3.1)***	***28551±420(5.0)***	***28636±379(5.4)***	***26885±82(1.7)***	***26856±103(1.3)***	***62334±3972(7.5)***	***62334±3972(7.5)***
	5	***28727±249(3.0)***	28989±389(3.6)	***30571±479(5.5)***	***30693±441(5.5)***	***28421±139(1.6)***	***28440±125(1.8)***	***64233±5147(7.5)***	***64242±5138(7.5)***
	1	47887±173(6)	***40827±2573(3.3)***	***41301±476(4.4)***	***41242±464(4.3)***	***38564±117(1.6)***	***38549±139(1.4)***	***70139±4037(7.5)***	***70144±4029(7.5)***
	2	43677±0(5.6)	***43024±2961(4.7)***	***40667±448(3.7)***	***40804±492(4.0)***	***38055±128(1.5)***	***38061±172(1.5)***	***70219±3947(7.5)***	***70219±3947(7.5)***
S4-B	3	***50458±193(3.4)***	***50440±239(3.2)***	***52701±547(5.6)***	***52632±519(5.4)***	***50091±210(1.6)***	***50103±151(1.8)***	***83768±3444(7.5)***	***83768±3444(7.5)***
	4	34946±288(5.6)	***33974±1347(3.2)***	***34526±317(4.4)***	***34478±318(4.7)***	***32820±95(1.5)***	***32864±148(1.5)***	***62765±3949(7.5)***	***62765±3949(7.5)***
	5	***38698±4308(4.5)***	***38901±3107(4.8)***	***36166±349(4.4)***	***36195±289(4.4)***	***34049±163(1.6)***	***34016±143(1.4)***	***65365±4152(7.5)***	***65369±4147(7.5)***
	1	42365±10(5.7)	***42128±314(5.3)***	***35903±393(3.5)***	***35922±456(3.5)***	***33866±168(1.5)***	***33892±143(1.5)***	***61633±3417(7.5)***	***61633±3417(7.5)***
	2	***49614±26(5)***	49797±74(6)	***44052±394(3.4)***	***44087±340(3.6)***	***41842±170(1.4)***	***41878±204(1.6)***	***71240±3779(7.5)***	***71240±3779(7.5)***
S4-C	3	50103±0(6)	***49263±282(5)***	***41426±369(3.6)***	***41299±442(3.4)***	***39367±173(1.4)***	***39386±133(1.6)***	***71625±2776(7.5)***	***71625±2776(7.5)***
	4	51311±0(6)	***50781±48(5)***	***44940±369(3.4)***	***45099±410(3.6)***	***42895±168(1.4)***	***42939±179(1.6)***	***72347±3396(7.5)***	***72347±3396(7.5)***
	5	46067±54(6)	***45495±105(5)***	***40747±390(3.6)***	***40751±340(3.4)***	***38984±120(1.4)***	***38994±79(1.6)***	***66381±3802(7.5)***	***66381±3802(7.5)***
Average Ranking		5.25	4.78	3.95	4.00	1.51	1.51	7.50	7.5
#of 'w-d-l'		43-18-59		4-116-0		1-118-1		0-120-0	
p-value		0.01		0.15		0.93		0.95	

The data of E3-A $\sim$ S2-C are omitted in the table due to the page limitation. The complete data has been attached as supplementary material.

versus sequence transfer strategy on all DCARP instances are listed in the penultimate row. We also have conducted the Wilcoxon signed-rank test with the level of significance 0.05 for the average total cost of the restart strategy and sequence transfer strategy of each meta-heuristic algorithm in all instances, and the p-value associated to each meta-heuristic algorithm are listed in the last row of Table V.

We conclude that the restart strategy is significantly different from the sequence transfer strategy when embedded in RTS ($0.01 < 0.05$), but both strategies obtain a similar performance when embedded in ILMA ($0.15 > 0.05$), MAENS ($0.93 > 0.05$) and MASDC ($0.95 > 0.05$). This is mainly because RTS is an individual-based meta-heuristic algorithm while ILMA, MAENS and MASDC are population-based algorithms. An individual-based algorithm only uses one solution for the optimisation. The solution generated by the sequence transfer strategy will be the only initial solution in the individual-based algorithm and therefore significantly influences the optimisation result. In contrast, the population-based algorithm contains a population during the optimisation so that it depends much less on a single inherited solution.

From the statistical test result and the number of 'win-draw-lose' of all DCARP instances for the individual-based algorithm, i.e. RTS, we conclude that the performance of the restart strategy seems slightly better than the sequence transfer strategy. The efficiency depends on how much information is inherited from the previous best solution. For a CARP solution, the most critical information is the sequence of tasks of each route. Therefore, if each route has many tasks left, and these remaining tasks can also maintain the tasks' sequence of the best solution in the previous instance, the inherited solution will be of high quality. The sequence transfer strategy fixes the remaining tasks' position to maintain the sequence information. Then, new tasks are inserted into the sequence of remaining tasks to construct a new initial solution. If an outside vehicle has only a few remaining tasks, most tasks in the new schedule will be the new tasks so that the new schedule is unlikely to benefit from the previous best schedule.

In contrast, if there are many remaining tasks for an outside vehicle, the order of remaining tasks will be maintained in the new sequence of tasks, and the Ulusoy's split will still assign them to an outside vehicle. Consider the following two remaining task sequences as an example:

$$S_1 : \{v_0, VT_1, t_1, v_0, VT_2, t_2, VT_3, t_3, v_0\},$$
$$S_2 : \{v_0, VT_1, t_1, t_2, t_3, t_4, t_5, v_0, VT_2, t_6, t_7, t_8, t_9, t_{10}, t_{11}, v_0\}.$$

S_1 has three outside vehicles with one remaining task for each vehicle, and S_2 has two outside vehicles with 5 and 6 remaining tasks, respectively. The task sequences in both remaining task sequences are the same as those of the previous instance's best schedule. After greedy insertion of a set of new tasks, the final task sequence served by outside vehicles in S_2 will be more similar to the schedule in the previous instance's best solution. As a result, the second scenario is more likely to obtain a high-quality inherited solution.

We have calculated the average remaining tasks for outside vehicles on all DCARP instances in our experiment. The result is presented in Figure 5. We conclude, since the average remaining tasks for each outside vehicle increases, the sequence transfer strategy benefits more from the optimisation experience in the previous instance and outperforms the restart strategy with a higher probability. However, it is not always the case that sequence transfer strategy performs better than restart strategy when the average number of remaining tasks is larger.

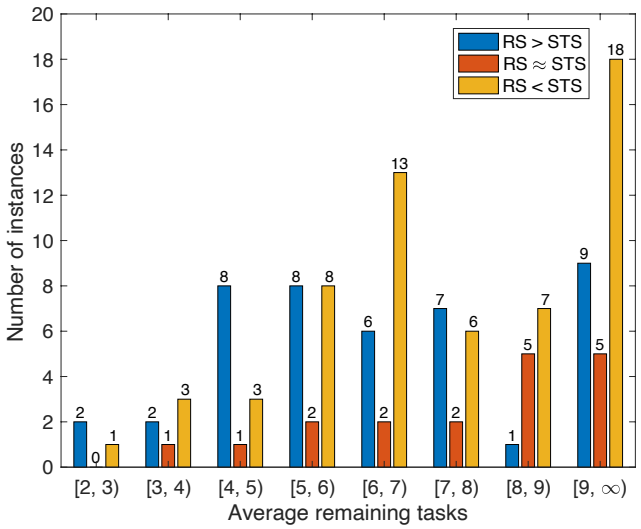

Fig. 5. The number of instances for the restart strategy (RS) wins over ($\succ$), draws against ($\approx$) and loses to ($\prec$) the sequence transfer strategy (STS) for different ranges of average remaining tasks.

We have also compared the rankings of each algorithm over 25 independent runs on each DCARP instance, which is presented in each cell's brackets in Table V. The overall average rankings of each algorithm instance over 120 DCARP instances are summarised in the row of 'Average Ranking' in Table V. The Friedman test with the level of significance 0.05 was carried out to compare the ranking of eight algorithms across problem instances, leading to a p-value of 8.69e-159. This indicates that at least one pair of algorithms are not equivalent to each other. We then perform Nemenyi posthoc tests to identify which algorithms perform significantly different. The critical difference diagram is presented in Figure 6 where the value of critical difference is 0.96 [38]. We conclude from this test that the restart and sequence transfer strategies obtain almost the same performance for population-based algorithms (MAENS and ILMA) while the sequence transfer strategy slightly outperforms the restart strategy in the individual-based algorithm (RTS) in our experiments.

Furthermore, MEANS obtains the best result, and RTS is the worst algorithm among the three meta-heuristic algorithms which are originally designed for static CARP. However, all of them significantly outperform the existing dynamic algorithm embedded with GOFVT, i.e. VT-MASDC and VTtr-MASDC. Recall that VT-MASDC was shown to outperform MASDC in Section V-C. Therefore, we conclude that not only the proposed framework generalises the existing algorithms for static CARPs to solve DCARPs but that also the constructed algorithm maintains its powerful optimisation ability when optimising a DCARP instance.

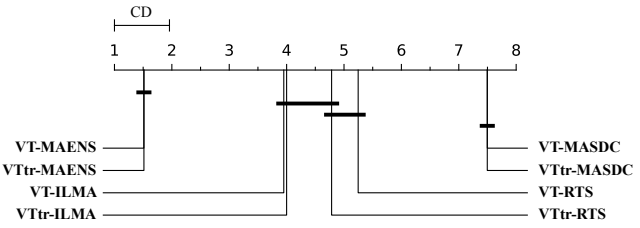

Fig. 6. Critical difference diagram for the comparison of 8 algorithms against each other on *egl* with Friedman test and Nemenyi test. Groups of algorithms which are not significantly different at the level of 0.05 are connected.

VI. Conclusion and future work

In this paper, we focused on the dynamic capacitated arc routing problem (DCARP), in which dynamic events, such as road closure, added tasks, etc., occur during the vehicles' service. First, a mathematical formulation of DCARP was provided for the first time in the literature. Then, we designed a simulation system as the research platform for DCARP. Unlike the existing benchmark generator, our simulation system generates DCARP instances from a given map in a way that both, the existing benchmark maps in static CARP and self-designed maps can be used in our simulation system. The events simulated in the system enable existing benchmark maps from static CARP scenarios to be adopted in scenarios much closer to the real world, where dynamic events happen while a solution is being deployed i.e., while vehicles are serving the map. For example, road congestion, as well as its recovering, can be considered in our simulation system. Given the mathematical model and the simulation system, we proposed a generalised optimisation framework which can generalise all algorithms for static CARPs to optimise DCARPs. In our framework, we propose a virtual-task strategy that constructs a virtual task between the stop location and the depot, to make all outside vehicles virtually return to the depot, which tremendously simplified the optimisation for DCARP. As a result, the DCARP instance was converted into a virtually 'static' instance so that algorithms for static CARP can directly solve it.

Two strategies were applied in the generalised optimisation framework: the restart and sequence transfer strategies. The restart strategy completely restarts the optimisation algorithm when there is a new DCARP instance. The sequence transfer strategy maintains the sequence of remaining tasks to new DCARP instances and greedily inserted the new tasks into the remaining sequence, to transfer the information of task sequence and benefit from the previous optimisation experience.

In our computational study, the necessity of directly optimising DCARP together with outside vehicles was first demonstrated by comparing the virtual-task strategy with a return-first strategy. The results indicated that it would be more efficient to optimise the DCARP instance by using the virtual-task strategy when the outside vehicles' remaining capacities were sufficiently large to serve more tasks. Then, the efficiency of the virtual-task strategy was demonstrated by embedding it into an existing algorithm and comparing it to the original version of the existing algorithm. Finally, the proposed generalised optimisation framework's efficiency was analysed by integrating a set of optimization algorithms that were designed for static CARPs, and the constructed algorithm instances performed significantly better than existing algorithms for DCARP. The performance of restart and sequence transfer strategies were compared and analysed. Our results indicate that the sequence transfer strategy is much more powerful for individual-based meta-heuristic algorithms when each outside vehicle has many remaining tasks.

We have demonstrated that the optimisation for DCARP by using the virtual-task strategy is suitable for the scenario where the outside vehicles' remaining capacities are sufficiently large to serve the new tasks. In future, it is valuable to investigate the influence of remaining capacities from a theoretical perspective. In addition, the sequence transfer strategy only uses the optimisation experience belonging to the instance of the previous optimisation. We assume that it is valuable to utilize all search experience taken from the whole DCAPR scenario. Finally, it is desirable to test the proposed framework with large scale instances and real world applications in the future.

Acknowledgements

Hao Tong gratefully acknowledges the financial support from Honda Research Institute Europe (HRI-EU).

References

[1] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Evolutionary algorithms for periodic arc routing problems," *European Journal of Operational Research*, vol. 165, no. 2, pp. 535–553, 2005.

[2] H. Handa, L. Chapman, and X. Yao, "Robust route optimization for gritting/salting trucks: A cercia experience," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 6–9, 2006.

[3] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *European Journal of Operational Research*, vol. 22, no. 3, pp. 329–337, 1985.

[4] B. L. Golden, J. S. DeArmon, and E. K. Baker, "Computational experiments with algorithms for a class of routing problems," *Computers & Operations Research*, vol. 10, no. 1, pp. 47–59, 1983.

[5] J. Brandão and R. Eglese, "A deterministic tabu search algorithm for the capacitated arc routing problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1112–1126, 2008.

[6] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[7] K. Tang, J. Wang, X. Li, and X. Yao, "A scalable approach to capacitated arc routing problems based on hierarchical decomposition," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3928–3940, 2016.

[8] S. Wøhlk and G. Laporte, "A fast heuristic for large-scale capacitated arc routing problems," *Journal of the Operational Research Society*, vol. 69, no. 12, pp. 1877–1887, Jan. 2018.

[9] M. C. Mourão and L. S. Pinto, "An updated annotated bibliography on arc routing problems," *Networks*, vol. 70, no. 3, pp. 144–194, Aug. 2017.

[10] Á. Corberán, R. Eglese, G. Hasle, I. Plana, and J. M. Sanchis, "Arc routing problems: A review of the past, present, and future," *Networks*, Jun. 2020.

[11] Y. Mei, K. Tang, and X. Yao, "Evolutionary computation for dynamic capacitated arc routing problem," in *Evolutionary Computation for Dynamic Optimization Problems*. Springer, 2013, pp. 377–401.

[12] J. Liu, K. Tang, and X. Yao, "Robust optimization in uncertain capacitated arc routing problems: Progresses and perspectives," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 63–82, 2021.

[13] L. Xing, P. Rohlfshagen, Y. Chen, and X. Yao, "An evolutionary approach to the multidepot capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 356–374, 2009.

[14] R. K. Arakaki and F. L. Usberti, "Hybrid genetic algorithm for the open capacitated arc routing problem," *Computers & Operations Research*, vol. 90, pp. 221–231, Feb 2018.

[15] J.-M. Belenguer, E. Benavent, N. Labadi, C. Prins, and M. Reghioui, "Split-delivery capacitated arc-routing problem: Lower bound and metaheuristic," *Transportation Science*, vol. 44, no. 2, pp. 206–220, May 2010.

[16] Y. Chen and J.-K. Hao, "Two phased hybrid local search for the periodic capacitated arc routing problem," *European Journal of Operational Research*, vol. 264, no. 1, pp. 55–65, Jan. 2018.

[17] J. de Armas, P. Keenan, A. A. Juan, and S. McGarraghy, "Solving large-scale time capacitated arc routing problems: from real-time heuristics to metaheuristics," *Annals of Operations Research*, vol. 273, no. 1-2, pp. 135–162, Feb. 2018.

[18] H. Handa, L. Chapman, and X. Yao, "Dynamic salting route optimisation using evolutionary computation," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2005, pp. 158–165.

[19] M. Tagmouti, M. Gendreau, and J.-Y. Potvin, "A dynamic capacitated arc routing problem with time-dependent service costs," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 1, pp. 20–28, 2011.

[20] M. Liu, H. K. Singh, and T. Ray, "A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 595–602.

[21] ——, "A benchmark generator for dynamic capacitated arc routing problems," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 579–586.

[22] M. Monroy-Licht, C. A. Amaya, A. Langevin, and L.-M. Rousseau, "The rescheduling arc routing problem," *International Transactions in Operational Research*, vol. 24, no. 6, pp. 1325–1346, 2017.

[23] W. Padungwech, J. Thompson, and R. Lewis, "Effects of update frequencies in a dynamic capacitated arc routing problem," *Networks*, vol. 76, no. 4, pp. 522–538, 2020.

[24] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multi-robot sensor-based coverage considering energy constraints," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5930–5935.

[25] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "Towards novel meta-heuristic algorithms for dynamic capacitated arc routing problems," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2020, pp. 428–440.

[26] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[27] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 327–343, 2005.

[28] F. T. Hanshar and B. M. Ombuki-Berman, "Dynamic vehicle routing using genetic algorithms," *Applied Intelligence*, vol. 27, no. 1, pp. 89–99, 2007.

[29] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and B. W. Thomas, "On modeling stochastic dynamic vehicle routing problems," *EURO Journal on Transportation and Logistics*, vol. 9, no. 2, p. 100008, 2020.

[30] H. Zou and M. M. Dessouky, "A look-ahead partial routing framework for the stochastic and dynamic vehicle routing problem," *Journal on Vehicle Routing Algorithms*, vol. 1, no. 2, pp. 73–88, 2018.

[31] L. Foulds, H. Longo, and J. Martins, "A compact transformation of arc routing problems into node routing problems," *Annals of Operations Research*, vol. 226, no. 1, pp. 177–200, Sep. 2014.

[32] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, Apr. 2017.

[33] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, Oct. 2012.

[34] M. Mavrovouniotis and S. Yang, "Adapting the pheromone evaporation rate in dynamic routing problems," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2013, pp. 606–615.

[35] L. Y. Li and R. W. Eglese, "An interactive algorithm for vehicle routeing for winter—gritting," *Journal of the Operational Research Society*, vol. 47, no. 2, pp. 217–228, 1996.

[36] Y. Mei, K. Tang, and X. Yao, "A global repair operator for capacitated arc routing problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 3, pp. 723–734, 2009.

[37] ——, "Improved memetic algorithm for capacitated arc routing problem," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 1699–1706.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.