Data-driven Meta-heuristic Algorithm for Dynamic Capacitated Arc Route Problem

Hao Tong*†
*The School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK.
†The Department of Computer Science and Engineering, SUSTech, Shenzhen 518055, China.

***Abstract*—The Dynamic Capacitated Arc Routing Problem (DCARP) aims at re-routing the service paths of vehicles for the capacitated arc routing problem when one or more dynamic events happen during vehicles' services and influence the current schedule. For example, a road may become congested or even not accessible anymore because of a traffic accident, which is likely to deteriorate the quality of the current schedule. Since a new DCARP instance in a DCARP scenario is similar to the previous DCARP instances, knowledge of the old optimisation process for the old DCARP instance is capable of promoting the dynamic optimisation for the new DCARP instance. However, the existing optimisation algorithms for solving DCARP never considered such knowledge but re-optimised the DCARP instance directly once dynamic events happen. Therefore, in this project, we targeted to make use of the knowledge of the old optimisation process and proposed a dynamic optimisation framework with historical solutions adaptation heuristic (DO-HSAH) for speeding up the DCARP's dynamic optimisation. The DO-HSAH includes two different strategies to cope with two kinds of DCARP instances: instances caused only by cost-changing events and instances caused by both cost-changing and task-changing events. In the empirical studies, a state-of-the-art DCARP optimisation algorithm was embedded into the proposed DO-HSAH framework, and the new algorithm was evaluated in a set of testing DCARP scenarios. The experimental results demonstrated that the DO-HASH could significantly improve the performance of dynamic optimisation for the DCARP.**

I. Introduction

The Capacitated Arc Routing Problems (CARP) is a classical and important combinatorial optimisation problem with a range of applications in the real world [1], [2]. It aims at assigning a fleet of vehicles with limited capacities starting from a depot to serve a set of edges with demands in a graph. Algorithms for solving CARP are targeted to schedule vehicles with limited capacities for serving a set of tasks on a map with minimal travelling costs.

However, in real-world applications, dynamic events are common when vehicles are in service, i.e., when a solution is partially executed, thus influencing the vehicles' follow-on service. For example, a road may be closed due to an accident, or new tasks may emerge during the vehicles' service. When that happens, a new graph, i.e. a new DCARP instance, is formed, in which vehicles would stop at different locations, labelled as outside vehicles, with various amounts of remaining capacities. As a result, the current schedule may become inferior or even feasible.

Therefore, DCARP optimisation aims to update the service schedule when the quality of the original schedule deteriorates due to some dynamic events in an uncertain environment. When dynamic events occur, the remaining tasks, the map's information, and the status of all service vehicles at that time compose a DCARP instance. DCARP optimisation is mainly targeted to re-optimise the DCARP instance and obtain an updated schedule for the new environment. When dynamic events happen, vehicles are usually located at different points and have served different tasks resulting in different remaining capacities. Therefore, besides CARP requiring the total demand being served by a vehicle not to exceed a vehicle's capacity, a DCARP instance has two additional constraints, i.e., (1) some vehicles are outside and have to start from these outside positions and (2) the remaining capacities of these outside vehicles are different from those of vehicles in the depot. These two constraints cause the algorithms for solving static CARP are not applicable to dynamic CARP directly.

The dynamic CARP was seldom considered in the literature, and they almost all focused on specific dynamic events in the literature. DCARP, to the best of our knowledge, was first investigated in [3] when considering the salting route optimisation problem. Liu et al. [4] defined some possible changes in DCARP and proposed a benchmark generator for DCARP [5]. Furthermore, Marcela et al. [6] deal with the rescheduling for DCARP, which considered the failure of vehicles, and Wasin et al. [7] considered new tasks in DCARP. In our previous work, we proposed a novel generalised optimisation framework with a virtual-task strategy for solving DCARP [8]. The framework deals with the outside vehicle as a virtual task so that the meta-heuristic algorithms designed for static CARP are also capable of solving DCARP instances.

However, as far as we know, current works always optimise the DCARP instance from scratch once a new DCARP instance is generated. Since the new DCARP instance is generated based on the old DCARP instance and the executable solution, the new DCARP instance is similar to the old DCARP instance. Therefore, the solutions obtained for the old DCARP instance can be used in the new environment, potentially promoting the optimisation for the new DCARP instance. Therefore, this project mainly considers such knowledge transfer in the DCARP optimisation, and our contributions are as follows:

- To the best of our knowledge, it is the first time to systematically make use of the historical solutions obtained in the old environment to promote the optimisation of DCARP.
- The dynamic events are considered progressively from

only cost-related dynamic events (OC) to both cost-related and task-related dynamic events (CT). A dynamic optimisation framework based on the proposed historical solution adaptation heuristic (Do-HASH) is proposed.
- The HASH includes two different strategies, i.e., HASH-OC and HASH-CT, for the DCARP instances with the above two types of dynamic events. In HASH, the historical solutions are re-constructed based on the concept of the building block. For DCARP instances with new tasks, HASH contains an insertion strategy to adapt the historical solutions to the new DCARP instances.
- A state-of-the-art meta-heuristic algorithm was embedded into the proposed DO-HASH framework, and the new algorithm was evaluated on a set of generated DCARP scenarios. The empirical results demonstrated that our DO-HASH framework significantly improves the performance of the original algorithm.

The remainder of this paper is organised as follows. Section II introduces the problem definition as well as some related work in dynamic vehicle routing problems. Section III introduces the proposed DO-HASH framework and the details of HASH strategy. Section IV presents experimental results for evaluating the performance of DO-HASH. Conclusions and future work are provided in Section V.

II. Background

A. Problem description

The 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 having the same full capacities. We can obtain an initial solution in I_0 and execute this solution in the graph. During the execution, some dynamics [4] happen at random points in time when vehicles are in service, thus changing the problem instance and potentially requiring a new better solution. Vehicles then continue to serve tasks from the positions they had stopped (stop points). 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 should be as low as possible for each DCARP instance. Let's first focus on the mathematical formulation for one DCARP instance.

The map for any DCARP instance I_m is provided 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 and 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 given by

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

where for each arc u, i.e. $< v_i, v_j > \in A$, 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 (traversing) cost $dc(u)$, a serving cost $sc(u)$ and a demand $dm(u)$. The deadheading cost of an arc means the cost that the vehicle just traverse this arc without serving while the serving cost is the cost when vehicles serve this arc. A subset $R \subseteq A$ contains all arcs required to be served in the graph. The arc $u \in R$ is named as 'task' and has a positive demand $dm(u) > 0$. For convenience, we use t to represent a task, and use an arc ID for identification.

The DCARP instance I_0 only contains vehicles at the depot. As for DCARP instances $I_m (m > 0)$, in addition to vehicles that are currently at the depot, there may also be outside vehicles with remaining capacities. These are vehicles that had already started to serve tasks when a dynamic event occurs. Suppose there are N_{veh} vehicles in total with a maximum capacity Q at the depot and N_{ov} ($N_{ov} \leq N_{veh}$) 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 remaining tasks with the 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 locations that outside vehicles located while routes $r_{N_{ov}+1}$ to r_K start from the depot. Each route can be represented by three components: starting vertex, an ordered list of tasks (arc IDs) and the final depot. Therefore, 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 location v_k and returns to the depot v_0, whereas l_k denotes the number of tasks served by route r_k. For route r_k, where $k > N_{ov}$, v_k equals to v_0. This representation is very easy to be converted to an explicit route by connecting two subsequent tasks using Dijkstra's algorithm so that the route cost can be calculated. 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.

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 and the constraints for DCARP are given as follows:

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

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:

$$RC_{r_k} = mdc(v_k, tail_{t_{k,1}}) + mdc(head_{t_{k,l_k}}, v_0) + \sum_{i=1}^{l_k-1} mdc(head_{t_{k,i}}, tail_{t_{k,i+1}}) + \sum_{i=1}^{l_k} sc(t_{k,i}) \quad (2)$$

where $head_t$, $tail_t$ denotes the head and tail vertices of the task, $mdc(v_i, v_j)$ denotes the minimal total deadheading cost traversing from node v_i to node v_j, and $sc(t_{k,i})$ denotes the serving cost of task $t_{k,i}$. 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. Related work

In the literature, there are two related but different research topics which target the (re)scheduling of vehicles in dynamic environments: Dynamic CARP (DCARP) and dynamic vehicle routing problem (DVRP). DCARP focuses on serving tasks which are the arcs in the graph, while DVRP focuses on serving vertices. For dynamic optimisation for VRP, many different algorithms and strategies have been proposed in the literature.

In DVRP, the dynamic strategies for speeding up the optimisation process when it has new DVRP instances are also very important. Many works have focused on this topic in the literature [9]. There are three main dynamic strategies in DVRP as follows:

1) Increasing diversity after a change: complete restart and partial restart. The main approach for increasing diversity after a change is to insert new solutions/individuals into the population. The complete restart means that all solutions are re-generated for the new optimisation, and parameter restart will maintain some old solutions. The repair operator should be used first if the old solution is not infeasible after the change.
2) Maintaining diversity during execution: introduce immigrant ants randomly or use the best from the previous environment in the new environment.
3) Memory schemes: promising solutions for the old environment are stored in the memory and are reused when dynamic events happen.

It is worth mentioning that the optimisation algorithm in most works for solving DVRP is the Ant Colony Optimisation (ACO) algorithm. These three strategies are applied to ACO to help with dynamic optimisation.

The ACO was originally proposed to solve the TSP inspired by the foraging behaviours of some ant species [10]. The ants will deposit pheromones on the ground to mark favourable paths. When dynamic events happen, the pheromone in edges remains for dynamic problems and can also be used for new environments. The pheromone matrix contains encrypted information about the characteristics of good solutions for these problems. In particular, pairs of customers which are visited in sequence in good solutions will have high values in the corresponding entries of the pheromone matrix [11].

Many strategies for dealing with DVRP have been proposed to handle dynamic events. For example, for cost-related dynamic events, Eyckelhof et al. [12] inherited the graph's pheromone matrix and adjusted a few edges' pheromones to increase the exploration ability. Most works focused on new-task dynamic events in the literature. For settings of pheromone when adding new tasks in the DVRP, the restart strategy is the most intuitive approach, which assigns an initial pheromone value to the new task [11]. In addition, according to the position of new tasks, η-strategy and τ-strategy are proposed to adjust the pheromone values of new tasks [13]. Recently, it also has special learning-based strategies for handling new tasks. For example, Xiang et al. [14] proposed a pairwise proximity-based ant colony algorithm for DVRP, which learns the pair preference of tasks and applies such pair preference in ACO when constructing the new solution.

III. Historical Solutions Adaptation Heuristic based Dynamic Optimisation Framework

As stated above, we are targeted to make use of the historical solutions in the dynamic optimisation to promote the optimisation for new DCARP instances. Thus, a HASH-based dynamic optimisation framework is proposed, and its structure is presented in Figure 1.

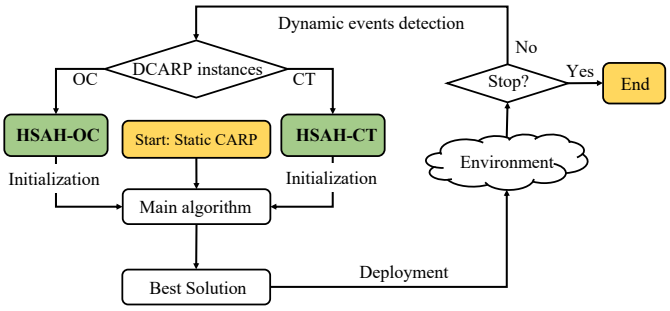


Fig. 1: The structure of HSAH-based dynamic CARP optimisation

The dynamic optimisation starts from the static CARP instance. The meta-heuristic optimisation algorithm can be used to obtain the best solution for the static CARP instance, and the best solution is deployed to serve the pre-defined tasks. Then, once some dynamic events happen, the control centre can detect the type of dynamic events and generate a new DCARP instance according to the occurred dynamic events. If the dynamic events only contain the cost-changing events (OC), such as road congestion, the HASH-OC strategy will be used to help the dynamic optimisation. On the other hand, if dynamic events also contain the added-task or deleted task events (CT), the HSAH-CT strategy will be applied to promote dynamic optimisation. As a result, a new solution will be obtained after the dynamic optimisation, and it will be deployed to the environment to update the service plan of all vehicles. Finally, the dynamic optimisation will stop until all tasks have been served.

In our framework, the HSAH strategy adapts the historical solutions and makes these new solutions as the initial solutions for the meta-heuristic algorithm. For OC and CT events, HSAH contains two different strategies, as HASH-OC and

HASH-CT, and both these two strategies are based on a new concept of *building block*. Therefore, in the following subsections, we will introduce the concept of the building block first, and the details of HASH-OC and HASH-CT are followed.

A. Building Block

The new DCARP instance is generated based on the old DCARP instance and the executable solution. Moreover, the solutions except the best solutions obtained in the optimisation process for old DCARP instances, i.e., the historical solutions, can not be directly used for the new DCARP instance. However, the information contained in these solutions can be extracted and reused. For example, it is highly probable that two adjacent tasks in the high-quality historical solutions still are adjacent in the best solutions for the new DACRP instance. Therefore, such sequence patterns can be extracted to construct the feasible solution for the new DCARP instance.

In this project, the *building block* is used to describe such sequence patterns. Since the CARP solution is a set of routes and the route is composed of a sequence of tasks, if a sub-sequence of tasks in the old CARP solution still remains in the new DCARP instances, i.e., the tasks in a sub-sequence in the route are still the tasks required to be served, we combine this sub-sequence of tasks into a *building block*. We assume the sequence of these tasks will still remain in the high-quality solutions of the new DCARP instances.

The pseudo-code of constructing building blocks from a historical solution is presented in Algorithm 1. Each building block has five properties, including the block's task sequence, the block's head node and tail node, and the block's cost and demand. In each historical solution, we detect each consecutive sub-sequence of tasks and make them a building block. The head node and tail node of this building block are the head and tail nodes of this task sub-sequence. The cost and demand of the building block are equal to the total costs of serving this task sub-sequence and the total demands of all tasks in this sub-sequence. If one task in the route that both its predecessor task and successor task have been served or deleted in the new DACRP instance, we also make this independent task a building task. Finally, all detected building blocks formed into a set of building blocks for the solution construction.

B. HSAH-OC

For DCARP instances only cost-changing dynamic events (DCARP-OC), the HSAH-OC strategy would be used to generate a new population of solutions for the new DCARP instances from an archive of historical solutions. The pseudo-code of HSAH-OC is presented in Algorithm 2.

The archived solutions AS, which are the final population of the meta-heuristic algorithm, were saved from the optimisation for the old DCARP instance. For each historical solution S_i, we first get a set of building blocks using Function 1 and then, the path-scanning constructive heuristic is applied to the obtained building blocks to generate a new solution NS_i. If it has no repetition of NS_i in the current population, the NS_i is added to the current population as one of the initial solutions for the meta-heuristic algorithm. Finally, if the size of new population is smaller than the pre-defined population size, generate corresponding number of solutions by optimisation algorithm's initialisation scheme to form a full population.

Function 1: `ConstructBuildingBlocks(`S, E_R`)`

Input: The historical solution: S; The task set E_R
Initial an empty set for building blocks $BLOCK = \varnothing$.
Initial a building block: bb, that $bb.seq = [\,]$.
for *each route* $r_k \in S$ **do**
  for i *from* 1 *to* l_k **do**
    if $t_{k,i} \notin E_R$ **then**
      if $bb.seq$ *is not empty* **then**
        Set $bb.head = bb.seq[0].head$.
        Set $bb.tail = bb.seq[-1].tail$.
        Set $bb.cost$ equals to the total serving costs of $bb.seq$.
        Set $bb.demand$ equals to the total demands of all tasks in $bb.seq$.
        $BLOCK = BLOCK \cup bb$.
        Re-initialize bb that $bb.seq = [\,]$.
      continue;
    else
      $bb.seq$.append($t_{k,i}$).
Output: The set of building blocks: $BLOCK$

Algorithm 2: HSAH-OC for DCARP-OC instances

Input: The set of archived solutions: AS; The new task set: E_R; The population size pop_size.
Initialise an empty population $POP = \varnothing$.
Set $l_{pop} = 0$.
for *each solution* $S_i \in AS$ **do**
  $BLOCK$ = `ConstructBuildingBlocks(`S_i, E_R`)`.
  Apply Path-Scanning algorithm to $BLOCK$ and obtain a new solution NS_i.
  if $NS_i \notin POP$ **then**
    $POP = POP \cup NS_i$.
    $l_{pop} += 1$.
if $l_{pop} > pop\_size$ **then**
  Remove the worst $pop\_size - l_{pop}$ solutions from POP.
if $l_{pop} < pop\_size$ **then**
  Generate $l_{pop} - pop\_size$ new solutions by optimisation algorithm's initialisation scheme and add them into POP.
Output: Population POP

C. HSAH-CT

Different from the DCARP-OC, DCARP instances with both cost-changing dynamic events and task-changing dynamic events (DCARP-CT) should consider the influence of

deleted tasks and added tasks, especially the added tasks. The insertion of new tasks has to be considered for the new DCARP instance. Therefore, the HASH-CT is proposed to handle the historical solutions, deleted tasks and added tasks for the DCARP-CT instances. The pseudo-code of HSAH-OC is presented in Algorithm 3.

Since we have a set of historical solutions in the archive, we need to filter the tasks that exist in the historical solutions and the newly added tasks. The newly added tasks are selected to form a new task set $NE_R \subset E_R$. Then, for generating new solutions from the historical solutions, HSAH-OC contains two different strategies, i.e., building block-based constructive and insertion strategies, from different perspectives.

The building block-based constructive strategy is similar to the HSAH-OC that considers the building blocks in the historical solutions. Additionally, HSAH-CT considers each new task as an independent building block. Thus, both the sequence pattern in the historical solutions and the new tasks are considered for generating the new solution. Then, the path-scanning algorithm is also used to generate new solutions.

On the other hand, the insertion strategy considers the complete sequence pattern of the historical solution and inserts the new task into the historical solutions to obtain new solutions. For each historical solution, tasks that have been served or deleted should be deleted first so that we can obtain an incomplete solution containing information about the historical solution. Since the minimal number of vehicles can be obtained according to the instance's information and the best solution of an instance usually only contains the minimal number of required vehicles, we add several empty routes to the current incomplete solution if the number of routes in the current incomplete solution is smaller than the minimal number of required vehicles. Then, the new tasks are inserted into the current incomplete solution to obtain a new feasible solution. If no route is feasible for all remaining tasks when insertion, a new empty route is inserted into the current solution and then continues to perform the greedy insertion.

After using two different strategies, a new population is obtained. If its size is greater than the pre-defined population size pop_size, the best pop_size solutions are selected as the initial solutions for the meta-heuristic algorithm. On the contrary, if the new population's size is smaller than the pre-defined population size due to removing repeated solutions, the corresponding number of solutions is also required to be generated by the optimisation algorithm's initialisation scheme to form a complete population.

Algorithm 3: HSAH for DCARP instances with both cost and task related events

Input: The set of archived solutions: AS; The new task set: E_R; The population size pop_size.

Filter the new task set $NE_R \subset E_R$.
/* Building blcok */
Initial an empty building block $BLOCK0 = \varnothing$.
Initial a building block: bb, that $bb.seq = [\,]$.
for *each task* $tk \in NE_R$ **do**
  Set $bb.head = tk.head$.
  Set $bb.tail = tk.tail$.
  Set $bb.cost$ equals to the service costs of t_k.
  Set $bb.demand$ equals to the demand of tk.
  $BLOCK0 = BLOCK0 \cup bb$.
Set $l_{pop} = 0$.
Initialise an empty population $POP = \varnothing$.
for *each solution* $S_i \in AS$ **do**
  $BLOCK$ = `ConstructBuildingBlocks(`S_i, E_R`)`.
  $BLOCK = BLOCK \cup BLOCK0$.
  Apply Path-Scanning algorithm to $BLOCK$ and obtain a new solution NS_i.
  if $NS_i \notin POP$ **then**
    $POP = POP \cup NS_i$.
    $l_{pop} += 1$.
/* Insertion */
Set W_T as the total demands of all tasks in E_R.
Set $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$.
for *each solution* $S_i \in AS$ **do**
  Delete all served tasks and deleted tasks in S_i.
  Set k as the number of route in S_i.
  if $k < k0$ **then**
    Add $k0 - k$ empty routes into S_i.
  while *All new tasks are inserted* **do**
    if *No route is feasible for all remaining tasks* **then**
      Add a new empty route into S_i.
    Find a task and a route with the smallest increasing cost for the insertion.
    Operate the insertion.
  Obtain a new solution NS_i.
  if $NS_i \notin POP$ **then**
    $POP = POP \cup NS_i$.
    $l_{pop} += 1$.
if $l_{pop} > pop\_size$ **then**
  Remove the worst $pop\_size - l_{pop}$ solutions from POP.
if $l_{pop} < pop\_size$ **then**
  Generate $l_{pop} - pop\_size$ new solutions from scratch and add them into POP.

Output: Population POP

IV. Experiments

We embedded a state-of-the-art meta-heuristic algorithm into our proposed DO-HSAH framework in this section. The new algorithm was evaluated in a series of DACRP scenarios. The empirical results are presented and analysed in this section.

A. Experimental settings

All experiments are conducted on a series of DCARP scenarios generated by the simulation system proposed in [8] based on a static CARP benchmark, namely the *egl* set [15]. 22

TABLE I: Results of MAENS with HASH and restart strategy in 3 DCARP scenarios only with cost-changing dynamic events. 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 a significance level of 0.05. The values in the last column summarise the number of win-draw-lose of HASH strategy versus the restart strategy over 10 different DCARP scenarios.

Mapname	SCN1		SCN2		SCN3		W-D-L
	HASH	RST	HASH	RST	HASH	RST	
egl-g1-A	$\mathbf{0.1244 \pm 0.0370}$	0.1943 ± 0.0461	$\mathbf{0.1100 \pm 0.0223}$	0.1472 ± 0.0343	$\mathbf{0.2034 \pm 0.0517}$	0.3037 ± 0.0654	10-0-0
egl-g1-B	$\mathbf{0.1358 \pm 0.0383}$	0.1989 ± 0.0540	$\mathbf{0.0907 \pm 0.0149}$	0.1272 ± 0.0272	$\mathbf{0.0894 \pm 0.0213}$	0.1209 ± 0.0258	10-0-0
egl-g1-C	$\mathbf{0.1561 \pm 0.0451}$	0.3010 ± 0.0483	$\mathbf{0.1050 \pm 0.0266}$	0.1912 ± 0.0380	$\mathbf{0.1020 \pm 0.0163}$	0.2037 ± 0.0423	10-0-0
egl-g1-D	$\mathbf{0.1047 \pm 0.0142}$	0.1971 ± 0.0310	$\mathbf{0.0986 \pm 0.0281}$	0.2148 ± 0.0360	$\mathbf{0.1264 \pm 0.0308}$	0.1837 ± 0.0382	10-0-0
egl-g1-E	$\mathbf{0.0638 \pm 0.0180}$	0.1986 ± 0.0417	$\mathbf{0.0854 \pm 0.0178}$	0.1236 ± 0.0277	$\mathbf{0.0707 \pm 0.0231}$	0.2147 ± 0.0422	10-0-0
egl-g2-A	$\mathbf{0.1455 \pm 0.0364}$	0.2338 ± 0.0527	$\mathbf{0.0577 \pm 0.0225}$	0.1206 ± 0.0386	$\mathbf{0.0823 \pm 0.0186}$	0.1785 ± 0.0302	10-0-0
egl-g2-B	$\mathbf{0.1380 \pm 0.0352}$	0.2407 ± 0.0449	$\mathbf{0.1007 \pm 0.0375}$	0.2330 ± 0.0248	$\mathbf{0.1362 \pm 0.0235}$	0.1657 ± 0.0309	10-0-0
egl-g2-C	$\mathbf{0.0803 \pm 0.0211}$	0.2238 ± 0.0475	$\mathbf{0.0878 \pm 0.0191}$	0.1631 ± 0.0300	$\mathbf{0.1014 \pm 0.0318}$	0.2237 ± 0.0424	10-0-0
egl-g2-D	$\mathbf{0.1624 \pm 0.0276}$	0.3028 ± 0.0444	$\mathbf{0.0779 \pm 0.0165}$	0.1405 ± 0.0221	$\mathbf{0.0724 \pm 0.0212}$	0.2463 ± 0.0279	10-0-0
egl-g2-E	$\mathbf{0.0832 \pm 0.0167}$	0.1587 ± 0.0316	$\mathbf{0.0822 \pm 0.0245}$	0.1530 ± 0.0350	$\mathbf{0.0938 \pm 0.0250}$	0.1979 ± 0.0471	10-0-0
egl-s1-A	$\mathbf{0.0474 \pm 0.0230}$	0.0744 ± 0.0220	$\mathbf{0.0347 \pm 0.0284}$	0.1536 ± 0.0440	$\mathbf{0.1313 \pm 0.0350}$	$\mathbf{0.1494 \pm 0.0330}$	4-4-2
egl-s1-B	$\mathbf{0.0001 \pm 0.0005}$	0.0042 ± 0.0080	$\mathbf{0.1112 \pm 0.0365}$	$\mathbf{0.1036 \pm 0.0347}$	$\mathbf{0.0075 \pm 0.0043}$	0.0102 ± 0.0038	3-7-0
egl-s1-C	0.1487 ± 0.0608	$\mathbf{0.0674 \pm 0.0433}$	$\mathbf{0.0201 \pm 0.0303}$	0.0411 ± 0.0261	$\mathbf{0.1237 \pm 0.0397}$	0.1794 ± 0.0723	8-1-1
egl-s2-A	$\mathbf{0.0584 \pm 0.0261}$	0.1063 ± 0.0553	$\mathbf{0.0450 \pm 0.0162}$	$\mathbf{0.0456 \pm 0.0181}$	$\mathbf{0.0278 \pm 0.0100}$	0.0386 ± 0.0130	8-2-0
egl-s2-B	$\mathbf{0.0182 \pm 0.0067}$	0.1057 ± 0.1132	$\mathbf{0.0086 \pm 0.0068}$	0.0217 ± 0.0133	$\mathbf{0.1094 \pm 0.0592}$	$\mathbf{0.1024 \pm 0.0890}$	5-5-0
egl-s2-C	$\mathbf{0.0220 \pm 0.0301}$	0.0449 ± 0.0362	$\mathbf{0.0104 \pm 0.0139}$	$\mathbf{0.0168 \pm 0.0173}$	$\mathbf{0.0183 \pm 0.0105}$	$\mathbf{0.0229 \pm 0.0089}$	3-6-1
egl-s3-A	$\mathbf{0.0254 \pm 0.0151}$	$\mathbf{0.0249 \pm 0.0087}$	$\mathbf{0.1170 \pm 0.0266}$	0.1423 ± 0.0444	$\mathbf{0.0827 \pm 0.0305}$	$\mathbf{0.0822 \pm 0.0348}$	6-4-0
egl-s3-B	$\mathbf{0.0625 \pm 0.0378}$	0.1046 ± 0.0532	$\mathbf{0.0651 \pm 0.0133}$	$\mathbf{0.0477 \pm 0.0406}$	$\mathbf{0.0012 \pm 0.0005}$	$\mathbf{0.0010 \pm 0.0009}$	7-3-0
egl-s3-C	$\mathbf{0.0416 \pm 0.0115}$	0.0791 ± 0.0252	$\mathbf{0.0045 \pm 0.0119}$	0.0231 ± 0.0245	$\mathbf{0.0641 \pm 0.0208}$	$\mathbf{0.0759 \pm 0.0223}$	5-4-1
egl-s4-A	$\mathbf{0.0178 \pm 0.0215}$	$\mathbf{0.0194 \pm 0.0182}$	$\mathbf{0.0063 \pm 0.0085}$	$\mathbf{0.0060 \pm 0.0069}$	$\mathbf{0.0728 \pm 0.0267}$	0.1082 ± 0.0401	5-5-0
egl-s4-B	$\mathbf{0.0608 \pm 0.0171}$	0.1022 ± 0.0479	$\mathbf{0.0012 \pm 0.0061}$	0.0083 ± 0.0150	$\mathbf{0.0260 \pm 0.0125}$	0.0738 ± 0.0332	5-5-0
egl-s4-C	$\mathbf{0.1049 \pm 0.0105}$	0.1628 ± 0.0618	0.0947 ± 0.0306	$\mathbf{0.0648 \pm 0.0398}$	$\mathbf{0.0863 \pm 0.0136}$	$\mathbf{0.1072 \pm 0.0523}$	4-5-1

different static CARP instances including 10 *egl-g* instances with 200 initial tasks and 12 *egl-e* instances with 100 initial tasks.

To generate different DCARP scenarios with cost-changing dynamic events, assign a base cost to each edge in the map indicating the minimal cost of an edge. Then, when simulating the cost-changing dynamic events, the deadheading cost of an edge is determined by the following equation:

$$deadheading\_cost = \begin{cases} base\_cost * r, & p < 0.5 \\ base\_cost, & otherwise \end{cases} \quad (3)$$

where $p \in U(0,1)$ is a random number from a standard uniform distribution, and $r \in U(1,C)$ is also a random from the uniform distribution with an upper bound value as C. The C is set as 5 in our experiments. The equation means that for each edge in the map, its deadheading cost will become r times base costs with a probability of 0.5 and its cost also a probability of 0.5 to become the base costs.

Then, for DCARP instances with task-changing dynamic events, each remaining task will vanish with a probability of $p_v = 0.2$ when generating new DCARP instances. The newly added tasks in each new DCARP instance are pre-defined in each static CARP instance.

We generate 10 different DCARP scenarios independently for each static CARP instance, and in each DCRAP scenario, we generate 5 DCARP instances at most. If the number of tasks in a new DCARP instance is smaller than 20, we also stop to continue to generate new DCARP instances.

In our experiment, the state-of-the-art DCARP optimisation algorithm, i.e., the memetic algorithm with extended neighbour search (MAENS) [16] with virtual task strategy, was used to embed into the proposed DO-HSAH framework. Its performance was evaluated by comparing the new algorithm's performance to the original optimisation algorithm with the restart strategy. For each DCARP instance, 25 independent runs are executed. The maximum optimisation time was set to 60s for all problems in our experiments. All programs are implemented in C language and run on Linux server with AMD Ryzen Threadripper PRO 3995WX 64-Cores 2.7GHZ.

B. Performance measurement

An average normalised cost for dynamic optimisation [17] is employed to measure the dynamic optimisation algorithm's performance for a DCARP scenario. The normalised cost was calculated by the following equation:

$$\overline{cost\_norm} = \sum_{m=1}^{M} \frac{TC_m - TC_{min}}{TC_{max} - TC_{min}} \quad (4)$$

where M is the number of instances in a DCARP scenario, TC_m is the cost of the best solution of m^{th} instance in the DCARP scenario. TC_{max} and TC_{min} are the maximum and minimal costs of all obtained best solutions' costs of m^{th} instances in the DCARP scenario over 25 independent runs. A smaller normalised cost indicates better performance in our experiments according to Equation 4.

C. Results and Analysis

1) DCARP-OC: The results of MAENS with HASH and restart strategy in DCARP scenarios only with cost-changing dynamic events are presented in Table I. The values in each cell represent "Mean ± Std" over 25 independent runs. For each DCARP scenario, the Wilcoxon signed-rank test with a significance level of 0.05 was applied. The significantly

TABLE II: Results of MAENS with HASH and restart strategy in 3 DCARP scenarios with both cost-changing and task-changing dynamic events. 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 a significance level of 0.05. The values in the last column summarise the number of win-draw-lose of the HASH strategy versus the restart strategy over 10 different DCARP scenarios.

Mapname	SCN1		SCN2		SCN3		W-D-L
	HASH	RST	HASH	RST	HASH	RST	
egl-g1-A	$\mathbf{0.0583 \pm 0.0168}$	0.0833 ± 0.0158	$\mathbf{0.0768 \pm 0.0200}$	$\mathbf{0.0852 \pm 0.0162}$	$\mathbf{0.1032 \pm 0.0193}$	0.1379 ± 0.0214	4-5-1
egl-g1-B	$\mathbf{0.0655 \pm 0.0153}$	0.0792 ± 0.0153	$\mathbf{0.0543 \pm 0.0083}$	0.0679 ± 0.0141	$\mathbf{0.0571 \pm 0.0106}$	$\mathbf{0.0535 \pm 0.0121}$	4-6-0
egl-g1-C	$\mathbf{0.0669 \pm 0.0192}$	$\mathbf{0.0733 \pm 0.0156}$	$\mathbf{0.0712 \pm 0.0113}$	0.0793 ± 0.0153	$\mathbf{0.0685 \pm 0.0180}$	0.0903 ± 0.0238	4-4-2
egl-g1-D	$\mathbf{0.0536 \pm 0.0106}$	0.0788 ± 0.0150	$\mathbf{0.0843 \pm 0.0230}$	0.0985 ± 0.0212	$\mathbf{0.0794 \pm 0.0083}$	$\mathbf{0.0730 \pm 0.0149}$	4-4-2
egl-g1-E	$\mathbf{0.0343 \pm 0.0097}$	$\mathbf{0.0370 \pm 0.0088}$	$\mathbf{0.0490 \pm 0.0089}$	0.1054 ± 0.0304	$\mathbf{0.0661 \pm 0.0313}$	$\mathbf{0.0587 \pm 0.0225}$	5-5-0
egl-g2-A	$\mathbf{0.0545 \pm 0.0120}$	0.0866 ± 0.0167	$\mathbf{0.1166 \pm 0.0220}$	$\mathbf{0.1228 \pm 0.0198}$	$\mathbf{0.0638 \pm 0.0116}$	0.0783 ± 0.0124	6-4-0
egl-g2-B	$\mathbf{0.0519 \pm 0.0127}$	0.0908 ± 0.0131	$\mathbf{0.0599 \pm 0.0103}$	0.0785 ± 0.0179	$\mathbf{0.0911 \pm 0.0179}$	$\mathbf{0.0935 \pm 0.0172}$	8-1-1
egl-g2-C	$\mathbf{0.0729 \pm 0.0136}$	$\mathbf{0.0709 \pm 0.0148}$	$\mathbf{0.0472 \pm 0.0096}$	0.0876 ± 0.0142	$\mathbf{0.0679 \pm 0.0127}$	0.0856 ± 0.0138	5-4-1
egl-g2-D	$\mathbf{0.0477 \pm 0.0250}$	$\mathbf{0.0518 \pm 0.0148}$	$\mathbf{0.0909 \pm 0.0200}$	$\mathbf{0.0916 \pm 0.0134}$	$\mathbf{0.0473 \pm 0.0140}$	0.0788 ± 0.0138	6-4-0
egl-g2-E	$\mathbf{0.0932 \pm 0.0156}$	0.1088 ± 0.0213	$\mathbf{0.0775 \pm 0.0175}$	0.1081 ± 0.0146	0.1003 ± 0.0178	$\mathbf{0.0859 \pm 0.0162}$	8-1-1
egl-s1-A	$\mathbf{0.0430 \pm 0.0215}$	0.0581 ± 0.0198	$\mathbf{0.0275 \pm 0.0129}$	$\mathbf{0.0288 \pm 0.0120}$	$\mathbf{0.0226 \pm 0.0055}$	$\mathbf{0.0243 \pm 0.0045}$	2-6-2
egl-s1-B	$\mathbf{0.0260 \pm 0.0105}$	0.0351 ± 0.0104	$\mathbf{0.0333 \pm 0.0140}$	$\mathbf{0.0368 \pm 0.0110}$	$\mathbf{0.0711 \pm 0.0149}$	$\mathbf{0.0763 \pm 0.0198}$	2-6-2
egl-s1-C	$\mathbf{0.0222 \pm 0.0074}$	0.0446 ± 0.0280	$\mathbf{0.0570 \pm 0.0279}$	0.1119 ± 0.0265	$\mathbf{0.0424 \pm 0.0130}$	0.0578 ± 0.0177	4-6-0
egl-s2-A	$\mathbf{0.0151 \pm 0.0112}$	$\mathbf{0.0182 \pm 0.0125}$	$\mathbf{0.0694 \pm 0.0181}$	0.0900 ± 0.0177	$\mathbf{0.0290 \pm 0.0146}$	0.0485 ± 0.0181	5-5-0
egl-s2-B	0.1382 ± 0.0366	$\mathbf{0.1020 \pm 0.0547}$	$\mathbf{0.0291 \pm 0.0091}$	$\mathbf{0.0266 \pm 0.0097}$	$\mathbf{0.0410 \pm 0.0122}$	$\mathbf{0.0470 \pm 0.0165}$	4-4-2
egl-s2-C	0.0821 ± 0.0208	$\mathbf{0.0641 \pm 0.0200}$	$\mathbf{0.0257 \pm 0.0187}$	0.0433 ± 0.0284	$\mathbf{0.0198 \pm 0.0078}$	$\mathbf{0.0182 \pm 0.0074}$	2-5-3
egl-s3-A	$\mathbf{0.0349 \pm 0.0159}$	0.0499 ± 0.0242	$\mathbf{0.0872 \pm 0.0311}$	$\mathbf{0.0890 \pm 0.0354}$	$\mathbf{0.0566 \pm 0.0142}$	$\mathbf{0.0597 \pm 0.0310}$	5-4-1
egl-s3-B	$\mathbf{0.0689 \pm 0.0246}$	$\mathbf{0.0753 \pm 0.0204}$	$\mathbf{0.0661 \pm 0.0177}$	$\mathbf{0.0663 \pm 0.0149}$	$\mathbf{0.0310 \pm 0.0165}$	0.0688 ± 0.0152	3-7-0
egl-s3-C	$\mathbf{0.0563 \pm 0.0282}$	$\mathbf{0.0424 \pm 0.0168}$	$\mathbf{0.0389 \pm 0.0136}$	0.0461 ± 0.0117	$\mathbf{0.0351 \pm 0.0151}$	$\mathbf{0.0318 \pm 0.0156}$	3-7-0
egl-s4-A	$\mathbf{0.0395 \pm 0.0119}$	0.0522 ± 0.0180	$\mathbf{0.0628 \pm 0.0161}$	0.0860 ± 0.0160	$\mathbf{0.0472 \pm 0.0144}$	$\mathbf{0.0569 \pm 0.0175}$	5-4-1
egl-s4-B	$\mathbf{0.0433 \pm 0.0111}$	$\mathbf{0.0447 \pm 0.0113}$	0.0952 ± 0.0169	$\mathbf{0.0848 \pm 0.0175}$	$\mathbf{0.0326 \pm 0.0093}$	$\mathbf{0.0359 \pm 0.0143}$	2-7-1
egl-s4-C	$\mathbf{0.0553 \pm 0.0105}$	$\mathbf{0.0572 \pm 0.0135}$	$\mathbf{0.0446 \pm 0.0176}$	$\mathbf{0.0623 \pm 0.0393}$	$\mathbf{0.0644 \pm 0.0164}$	$\mathbf{0.0661 \pm 0.0297}$	3-7-0

better results are highlighted with bold fonts in the table. Due to the page limitation here, only 3 DCARP scenarios are presented in Table I. However, the number of win-draw-lose of HASH strategy versus restart strategy over 10 different DCARP scenarios are presented in the last column of Table I.

From the results of Table I, we can find that the HASH strategy outperforms the restart strategy in all DCARP scenarios of *egl-g* maps which has more number of tasks. In the DCARP scenarios generated from *egl-e* maps, the HASH strategy outperforms the restart strategy or has a similar performance compared with the restart strategy in most scenarios. The restart strategy only outperforms the HASH in very few DCARP scenarios.

The difference in performance between HSAH and restart strategy is probably because of the number of tasks in the DCARP instance since *egl-g* maps have much more tasks than *egl-e* maps. The HSAH could perform significantly better in the DCARP instance with more tasks. To validate such a hypothesis, we define a performance gap as follows:

$$gap = cost\_norm^{RST} - cost\_norm^{HSAH} \tag{5}$$

where $cost\_norm^{RST}$ and $cost\_norm^{HSAH}$ represent the normalised cost of the restart strategy and HSAH strategy, respectively in each DCARP instance. Then, we calculate the average performance gap over 25 independent runs and the number of tasks for all DCARP instances, which was presented in Figure 2.

From 2, it is clear that when the number of task increases, the performance gap between the HSAH strategy and restart strategy become large, indicating that the HSAH strategy promotes the dynamic optimisation much more on instances

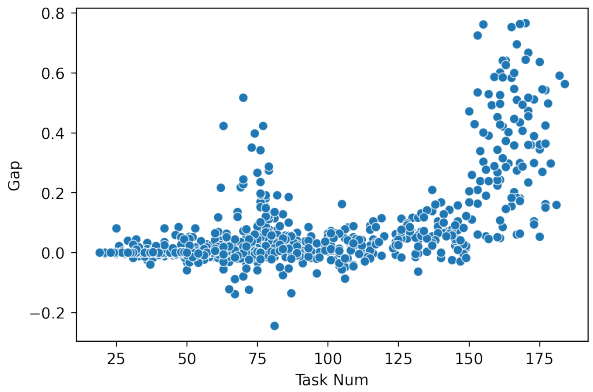
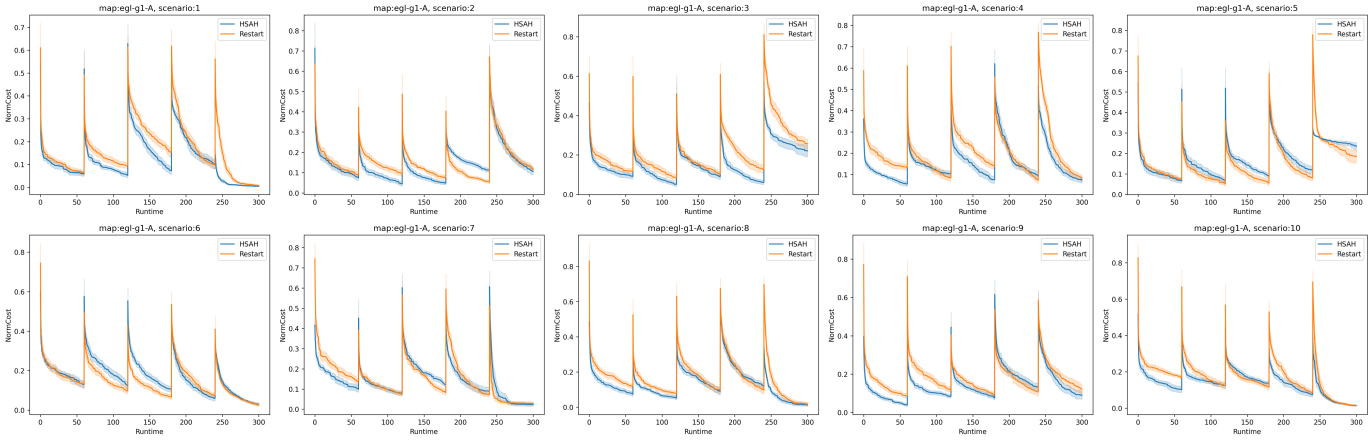


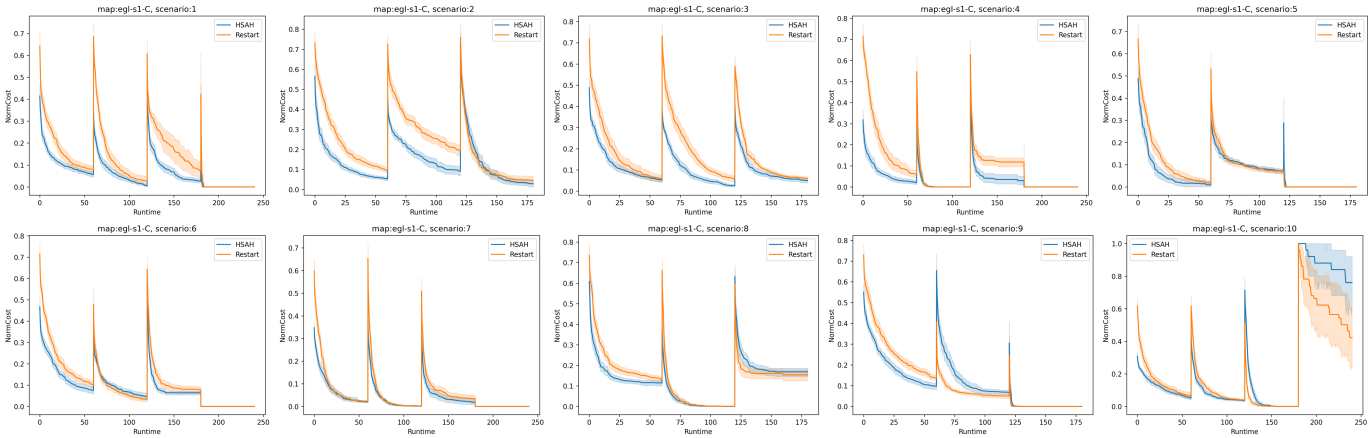
Fig. 2: The relation between the performance gap and the number of tasks for all DCARP instances.

with more number of tasks. For instances with a smaller number of tasks, all instances are too easy for the algorithm, such that dynamic optimisation is hard to benefit a lot from the adapted solutions. On the other hand, for instances with a bigger number of tasks, all instances are hard for the algorithm, and the adapted solutions can help the algorithm locate a better search space and obtain a better solution within a limited time.

However, it still has some outliers in Figure 2, and these outliers almost concentrate in the range of $[60, 80]$ in the figure. Only a few points are outliers. For outliers with large gap values, the adapted solutions are much closer to the final best solutions, which helps a lot in finding the final best solution. Moreover, for instances with small values, it is probably because of the search diversity. The adapted solutions



(a) egl-g1-A



(b) egl-s1-C

Fig. 3: The evolved curves of MAENS with HASH (*blue lines*) and restart (*orange lines*) strategies in 10 DCARP scenarios only with cost-changing dynamic events in maps of egl-g1-A (a) and egl-s1-C (b).

are similar to each other, which decreases the diversity and is easy to make the algorithm trap in the local optima in some instances.

Besides, we also plotted the convergence curves for MAENS with HSAH and restart strategies to investigate the influence of the HSAH strategy on the algorithm's convergence performance. 20 convergence curves are provided in Figure 3 including 10 scenarios of *egl-g1-A* (Figure 3a) and 10 scenarios of *egl-s1-C* (Figure 3b).

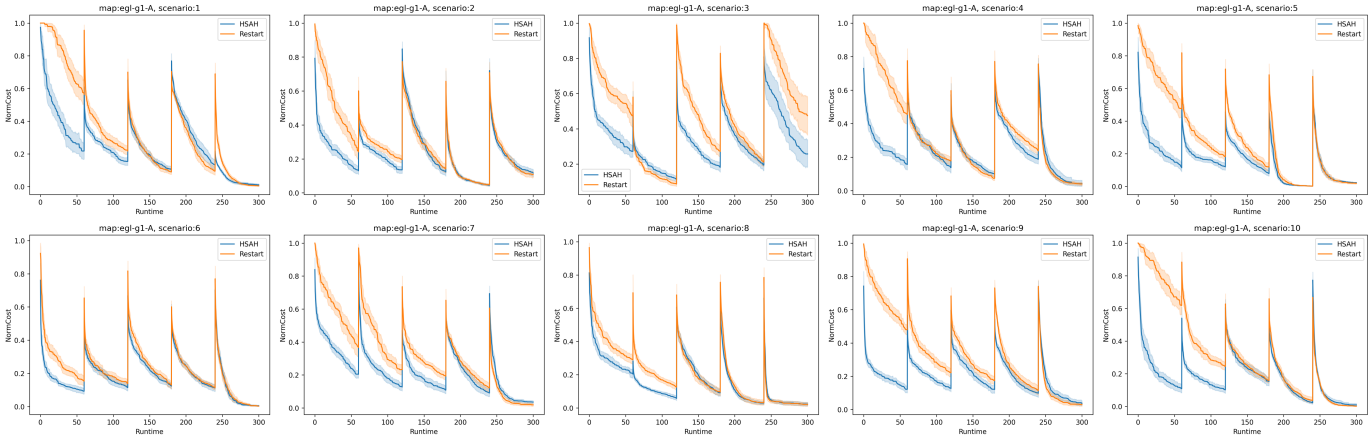
From 3, it is clear that the HSAH strategy can also promote the convergence speed of the dynamic optimisation for DCARP instances with both cost-changing and task-changing dynamic events.

2) DCARP-CT: The results of MAENS with HASH and restart strategy in DCARP scenarios with both cost-changing and task-changing dynamic events are presented in Table II. The values in each cell represent "Mean $\pm$ Std" over 25 independent runs. For each DCARP scenario, the Wilcoxon signed-rank test with a significance level of 0.05 was applied. The significantly better results are highlighted with bold fonts in the table. Due to the page limitation here, only 3 DCARP scenarios are presented in Table II. However, the number of win-draw-lose of HASH strategy versus restart strategy over 10 different DCARP scenarios are presented in the last column of Table II.

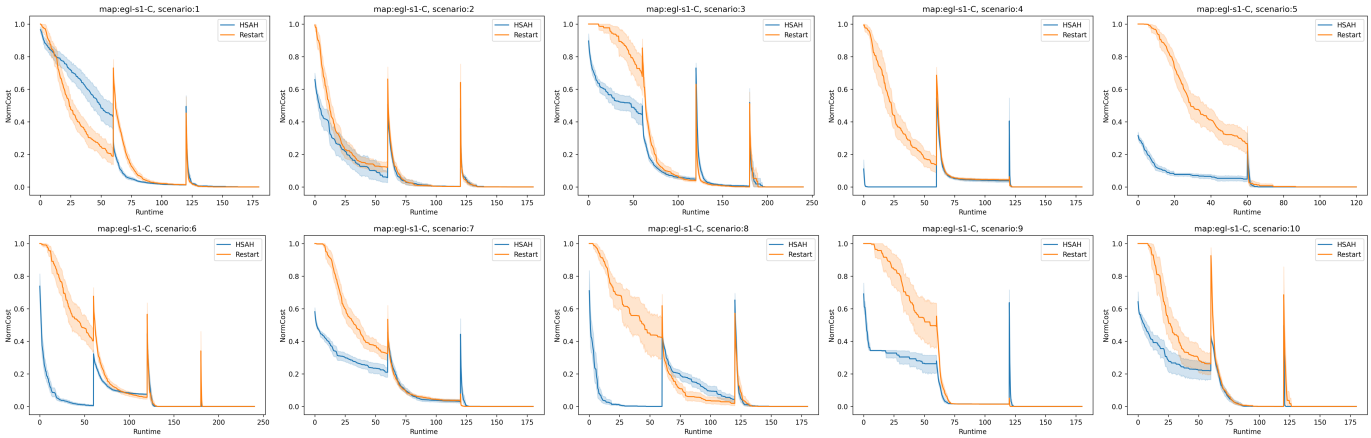
From the results of Table II, we can find HASH strategy outperforms the restart strategy or has a similar performance compared with the restart strategy in most scenarios. The restart strategy only outperforms the HASH in very few DCARP scenarios.

Besides, we also plotted the convergence curves for MAENS with HSAH and restart strategies to investigate the influence of the HSAH strategy on the algorithm's convergence performance. 20 convergence curves are provided in Figure 4 including 10 scenarios of *egl-g1-A* (Figure 4a) and 10 scenarios of *egl-s1-C* (Figure 4b).

From Figure 4, it is very clear that the large gap only exists in the first DCARP instance of a DCARP scenario. In the remaining DCARP instances, the HSAH's performance is always similar to the restart strategy. The first reason is probably that the historical solutions from the static CARP are obtained with sufficient computational resources, while the historical solutions for the remaining DCARP instances are from the optimisation process of the old DCARP instance, which does not have computational resources. On the other hand, in the latter DCARP instances of a DCARP scenario,



(a) egl-g1-A



(b) egl-s1-C

Fig. 4: The evolved curves of MAENS with HASH (*blue lines*) and restart (*orange lines*) strategy in 10 DCARP scenarios only both cost changing and task-changing dynamic events in maps of egl-g1-A (a) and egl-s1-C (b).

the number of tasks usually decreases a lot because many tasks have been served. It makes the DCARP instance easy to solve, and the initialisation solutions do not greatly impact the optimisation's performance.

V. Conclusion

In this project, we focused on the optimisation for the DCARP scenario, which considers the dynamic events during the deployment of the CARP solution. In the existing work for DCARP, the re-optimisation from scratch is usually applied to solve the DCARP scenario. However, since the DCARP instances in a DCARP scenario are generated based on the last DCARP instances, and many similarities exist between them, the information on optimisation for the old DCARP instances can be used to promote the dynamic optimisation for new DCARP instances. Therefore, in this work, we proposed a historical solution adaptation heuristic-based dynamic optimisation framework (DO-HSAH) to help optimise the DCARP scenario. The proposed DO-HSAH includes two strategies, i.e., HSAH-OC and HSAH-CT, to handle the DCARP instances generated from the only cost-changing dynamic events or both with cost-changing and task-changing dynamic events separately.

In the empirical studies, we embedded a state-of-the-art dynamic optimisation algorithm, i.e., MAENS, into the DO-HSAH framework and evaluated it on a series of DCARP scenarios. The performance of DO-HSHA was compared with the restart strategy in our experiment. The experimental results demonstrated that the adapted historical solutions indeed promote the dynamic optimisation for CARP and also lead to a fast convergence.

Currently, we directly adapted the historical solutions to the new DCARP instances to promote dynamic CARP optimisation. Its performance is still similar to or even worse than the restart strategy, especially for DCARP instances with task-changing dynamic events. Therefore, in the future, we will focus on extracting more indirect experience, such as constructive rules, from historical solutions to generate higher-quality solutions for new DCARP instances.

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] H. Handa, L. Chapman, and X. Yao, "Dynamic salting route optimisation using evolutionary computation," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 158–165, IEEE, 2005.
[4] 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)*, pp. 595–602, IEEE, 2014.
[5] M. Liu, H. K. Singh, and T. Ray, "A benchmark generator for dynamic capacitated arc routing problems," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 579–586, IEEE, 2014.
[6] 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.
[7] 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.
[8] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "A novel generalised meta-heuristic framework for dynamic capacitated arc routing problems," 2021.
[9] 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.
[10] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.
[11] 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.
[12] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic tsp," in *International workshop on ant algorithms*, pp. 88–99, Springer, 2002.
[13] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic tsp," in *Workshops on applications of evolutionary computation*, pp. 213–222, Springer, 2001.
[14] X. Xiang, Y. Tian, X. Zhang, J. Xiao, and Y. Jin, "A pairwise proximity learning-based ant colony algorithm for dynamic vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 5275–5286, jun 2022.
[15] 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.
[16] 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.
[17] 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.