Building Blocks as Experiences in Dynamic Capacitated Arc Routing Problems

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

***Abstract*—The Dynamic Capacitated Arc Routing Problem (DCARP) aims to update the service paths of vehicles in the capacitated arc routing problem when uncertain factors deteriorate the current schedule of vehicles' services. A DCARP scenario comprises a series of DCARP instances that share similarities with each other. Therefore, optimization experience gained from solving the former DCARP instance can potentially facilitate the optimization for a new DCARP instance. However, existing optimization algorithms for solving DCARP seldom consider such optimization experience and instead re-optimize the DCARP instance from scratch. This paper proposes a dynamic optimization framework with a solution building block adaptation strategy (DO-SBBA) that extracts the optimization experience from the former optimization process to facilitate the optimization of the next DCARP instance. The framework introduces the concept of building blocks for extracting the valuable experience contained in historical solutions. The building block-based constructive heuristic is proposed to handle DCARP scenarios with cost- or task-changing dynamic events, and an insertion heuristic is proposed to handle task-changing dynamic events. Experimental studies demonstrate the effectiveness of DO-SBBA for extracting and utilizing optimization experience in DCARP scenarios, significantly improving the performance of dynamic optimization compared to state-of-the-art DCARP methods.**

***Index Terms*—Experience-based optimization, uncertain environment, CARP, dynamic optimization, building blocks.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) is a classical and important combinatorial optimization problem [1] that is abstracted from a range of real-world applications, such as waste-collection [2], [3] and road gritting [4]. The problem aims to assign a fleet of vehicles with limited capacities, starting from a depot, to serve a set of edges/arcs with demands (also known as tasks) in a graph [5]. Algorithms [6] for solving CARP instances aim to find a CARP solution scheduling vehicles to serve a set of tasks in the graph with minimal total costs, subject to the following constraints:

- Each vehicle must start and return to the depot.
- Each task must be served once and only once.
- The total demand of tasks served by one vehicle cannot exceed the vehicle's capacity.

In real-world applications, the service plan for vehicles is typically obtained offline. However, unforeseen dynamic events often occur in uncertain environments when the CARP solution is in deployment and vehicles are serving their tasks [7], [8], [9], [10]. As a result, the original service plan would be impacted and could become inferior or even infeasible. For example, a road may be closed due to an accident, or new tasks may emerge during the vehicles' service. In such cases, a new Dynamic Capacitated Arc Routing Problem (DCARP) instance is formed, in which vehicles would start at different locations, referred to as "outside vehicles" with varying amounts of remaining capacities. The remaining service routes for vehicles can be improved by re-optimizing the new DCARP instance.

A DCARP scenario is composed of a series of DCARP instances. A new DCARP instance is generated based on the preceding DCARP instance and a partially executed solution. As a result, a new DCARP instance in a DCARP scenario shares similarities with its preceding DCARP instance. Intuitively, the optimization experience and solution information of the former DCARP instance could be extracted to promote the optimization for the new DCARP instance in a DCARP scenario. However, current existing works do not consider and utilize such optimization experience [11], [12], [8], [13], [9], [14], wasting valuable knowledge that could be used to enhance the optimization process for DCARP instances. Instead, they always optimize the new DCARP instance from scratch once dynamic events happen.

Additionally, in the literature, the dynamic optimization problems are prevalent in both continuous and combinatorial problem domains, and numerous effective algorithms have been developed to handle the dynamic changes in these problems [15], [16], [17], [18]. However, most algorithms leveraging the optimization experience for dynamic optimization problems were specifically designed for continuous optimization problems or different combinatorial problems, and are also not applicable for DCARP optimization.

Therefore, in this paper, we aim at answering the following three Research Questions (RQs):

- RQ1: How to extract optimization experience in DCARP? How effective is such an approach and why?
- RQ2:How to integrate optimization experience into a meta-heuristic framework for enhanced performance? How well does this framework perform compared to existing DCARP algorithms?
- RQ3: How is the effectiveness of the experience-based framework affected by DCARP instances with different severity of dynamic changes?

To answer these questions, we propose a solution building block adaptation strategy to extract useful optimization experience and incorporate it into a dynamic optimization

Hao Tong, and Xin Yao (*corresponding author*) are with the School of Data Science, Lingnan University, Hong Kong SAR, China. (email: haotong@ln.edu.hk, xinyao@ln.edu.hk)

Leandro L. Minku is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK. (email: L.L.Minku@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, bernhard.sendhoff@honda-ri.de)

framework able to transfer valuable optimization experience for the DCARP scenario, improving optimization performance. Our methodological contributions are as follows:

- We propose a new DCARP optimization framework that benefits from experience through knowledge transfer in the DCARP scenario to improve optimization of DCARP instances.
- We propose the new concept of *building block* for extracting optimization experience contained in historical solutions. Based on the building block, we propose the solution building block adaptation (SBBA) strategy to handle two different DCARP scenarios including DCARP-OC (containing only cost-changing dynamic events) and DCARP-CT (containing both cost-changing and task-changing dynamic events) scenarios.

We have evaluated our framework considering dynamic events from only cost-changing dynamic events (OC) to both cost-changing and task-changing dynamic events (CT). In particular, we consider two different types of DCARP scenarios, i.e., DCARP-OC and DCARP-CT scenarios. This evaluation led to the following new findings:

- Our computational analysis shows that the adapted historical solutions are significantly better than the solutions generated by conventional constructive heuristics when dynamic events are small and the high-quality solutions generated by SBBA for the new environment are similar to those of the previous environment.
- Our empirical results show that our SBBA-based dynamic optimization framework (DO-SBBA) significantly improves the performance of the original algorithms on these DCARP-OC and DCARP-CT scenarios, and is particularly beneficial on the DCARP-OC scenarios.
- The robustness of the proposed DO-SBBA framework was also investigated on DCARP instances with varying severity of dynamic changes. The empirical results show that our DO-SBBA framework was particularly effective in DCARP-OC scenarios with various severity of dynamic changes and DCARP-CT scenarios without a large change in the number of tasks.

The remainder of this paper is organized as follows. To clearly introduce and explain the studied DCARP in our paper, we firstly introduces the characteristics of the DCARP and provides its formulation in Section II. Section III introduces the related works about dynamic optimization and the motivations of this work. Section IV describes the proposed DO-SBBA framework and the details of SBBA strategy. Section V presents experimental studies of evaluating the performance of DO-SBBA. Conclusions and future work are provided in Section VI.

II. Problem Description

A. Characteristics of Dynamic Behavior in Studied DCARP

The DCARP studied in this paper focuses on tracking the moving optimum and updating the schedule for completing the remaining tasks when dynamic events deteriorate the current deployed solution during vehicles' services. Such dynamic events include, for instance, unexpected closure of roads or addition of new tasks occurring at random points in time during deployment. They result in the generation of a new DCARP instance to formulate the problem in the new environment, and the optimization for the new DCARP instance is applied to update the vehicle routing *during deployment*. As a result, a complete DCARP scenario is composed of a series of discrete DCARP instances corresponding to each environmental change, and optimizing these DCARP instances to update vehicle routing ensures that the vehicle routing remains efficient throughout the service period. As the optimization of a new problem instance happens during deployment, such optimization needs to be performed within a limited amount of time, given the specific problem domain being tackled.

In the literature, the uncertain CARP (UCARP) [19], [7] also deals with uncertainties in CARP scenarios. However, they are interested in finding *robust* solutions. These are solutions that are obtained offline *before deployment* and that one hopes will not deteriorate too much under different environment conditions that may be encountered during deployment. Therefore, UCARP is a considerably different problem from the one tackled in this paper. Moreover, even though the UCARP solutions are hoped not to deteriorate too much under certain environment conditions (e.g., different amounts of congestion), they are unlikely to be optimal for each individual environment condition that may be encountered during deployment. In DCARP, we are interested in updating solutions so that they remain optimal under each individual environment condition.

Dynamic optimization is also ubiquitous in various domains [15], [16]. Typically, for most dynamic optimization problems investigated in the literature, uncertain factors only affect the objective functions, causing the change of global optima. However, in the case of our studied DCARP, dynamic events influence the decision variable space as well, wherein the problem's dimension and the previous solution's feasibility is likely to change after dynamic events. In particular, in our studied DCARP, the solution has already been partially executed when a dynamic event occurs, such that some tasks in the original problem have been served and are not required to be served again. Moreover, newly emerged tasks and potential road closure are likely to make previous solutions infeasible. As a result, the decision variable space for the newly generated DCARP instance is different with the former DCARP instances causing solutions obtained for former DCARP instances to be unsuitable for the new DCARP instances. Tracking the moving optimum and updating the schedule for the remaining tasks in the DCARP scenario is a challenge due to significant changes in the decision space after the environment changes. The introduction of unexpected new tasks further complicates the rescheduling process, since there is no prior knowledge about how to efficiently allocate these new tasks within the new environment.

B. Problem Formulation

A DCARP scenario comprises a series of DCARP instances: $\mathcal{I} = [I_0, I_1, ..., I_M]$. Each DCARP instance, except I_0, contains all the information about the map and the state

of outside vehicles, which is generated from the most recent former DCARP instance and the deployed solution. The initial problem instance I_0 is a conventional static CARP instance [20]. An initial solution for a DCARP scenario is obtained by optimizing I_0 and then deployed to vehicles to serve tasks in the graph that represents the map of the problem instance. During the service process, some unpredictable dynamic events [9] may occur, leading to a change in the problem instance and potentially requiring a new, better solution. Once the current service plan is updated by a better solution obtained from optimizing the DCARP instance, vehicles continue to serve tasks according to the updated solution and start from the positions where they had previously stopped. The DCARP scenario terminates when all tasks have been served, and all vehicles have returned to the depot. In this paper, the maximum number of vehicles is not limited and we can schedule as many vehicles as desired to serve the tasks.

Suppose the map of a DCARP instance I_t, which is generated at a time point t, is given by graph $G = (V, A)$, with a set of vertices V and arcs (directed links) A. The graph contains a depot $v_0 \in V$. Each arc $u \in A$ in the graph has an associated deadheading (traversing) cost $dc(u)$, a serving cost $sc(u)$, and a demand $dm(u)$. The deadheading cost of an arc represents the cost incurred when a vehicle traverses this arc without serving, while the serving cost is the cost of serving this arc. For an arc u with $dm(u) > 0$, it represents a task that needs to be served, and all tasks in the graph form a subset $R \subseteq A$. For generated DCARP instance I_t, it will have outside vehicles with remaining capacities since they are in service when dynamic events happen. Suppose there are N_{ov} outside vehicles with remaining capacities $\{q_1, q_2, ..., q_{N_{ov}}\}$ currently located at outside positions labeled as $OV = \{v_1, v_2, ..., v_{N_{ov}}\}$. The optimization of the DCARP instance I_t aims to reschedule all the remaining tasks R with minimal cost considering both outside and depot vehicles. Suppose there are K routes in the solution $S_t = \{r_1, r_2, ...r_K\}$ of the DCARP instance I_t. The objective function of a DCARP instance I_t is given as follows:

$$\textbf{Min} \quad TC(S_t) = \sum_{k=1}^{K} RC_{r_k}, \tag{1}$$

subject to the following constraints:

$$\textbf{s.t.} \quad \sum_{k=1}^{K} l_k = |R| \tag{2a}$$

$$t_{k_1,i_1} \neq t_{k_2,i_2}, \text{for all } (k_1, i_1) \neq (k_2, i_2) \tag{2b}$$

$$\sum_{i=1}^{l_k} dm(t_{k,i}) \leq q_k, \forall k \in \{1, 2, ...N_{ov}\} \tag{2c}$$

$$\sum_{i=1}^{l_k} dm(t_{k,i}) \leq Q, \forall k \in \{N_{ov}+1, ..., K\}, \tag{2d}$$

where $t_{k,i}$, l_k and RC_{r_k} denote the i^{th} task, the number of tasks and the total costs of the k^{th} route (r_k), respectively. The RC_{r_k} is computed according to Eq. 3:

$$\begin{aligned} RC_{r_k} = {} & mdc(v_k, tail_{t_{k,1}}) + mdc(head_{t_{k,l_k}}, v_0) + \\ & \sum_{i=1}^{l_k-1} mdc(head_{t_{k,i}}, tail_{t_{k,i+1}}) + \sum_{i=1}^{l_k} sc(t_{k,i}), \end{aligned} \tag{3}$$

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 constraints (2a) and (2b) guarantee that all tasks are served once and only once. Since outside vehicles and new vehicles starting from the depot have different remaining capacities, the total demands of the route corresponding to different kinds of vehicles have to satisfy different capacity constraints as presented in the constraints (2c) and (2d). The constraint (2c) is for outside vehicles, and the constraint (2d) is for new vehicles from the depot.

III. Related Works and Motivations

A. Related Works

In the literature, the Dynamic Vehicle Routing Problem (DVRP) also targets the rescheduling of vehicles in dynamic environments, which is significantly similar to our studied DCARP. Many different algorithms and strategies have been proposed for solving the DVRP, mainly including swarm intelligence and evolutionary algorithms [21], [16], [22]. For example, Hanshar et al. [23] applied the genetic algorithm to solve DVRP and Sabar et al. [24] proposed a self-adaptive evolutionary algorithm to tackle DVRP. It is worth noting that most works for DVRP focus on the Ant Colony Optimization (ACO) algorithm [25], [26]. The ACO is naturally suitable for tackling dynamic problems since the pheromone model implements the memory strategy and inherit the past experience [27]. Mavrovouniotis et al. [27] investigated the behaviors of ACO in dynamic travelling salesperson problem (DTSP) and concluding that the performance of ACO highly relies on the pheromone update policy. Moreover, Eyckelhof et al. [28] inherited the graph's pheromone matrix and adjusted the pheromones of a few edges to increase the exploration ability for cost-changing dynamic events. Most works in the literature focus on new-task dynamic events. 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 [29]. In addition, according to the position of new tasks, η-strategy and τ-strategy are proposed to adjust the pheromone values of new tasks [30]. Recently, there are also special learning-based strategies for handling new tasks. For example, Xiang et al. [31] proposed a pairwise proximity-based ACO for DVRP, which learns the pair preference of tasks and applies such pair preferences when constructing the new solution. However, these strategies are only suitable for ACO algorithms and not for general meta-heuristic optimization algorithms such as the ones that have been obtaining state-of-the-art results in static and dynamic CARP [20], [9].

Dynamic CARP optimization focuses on the rescheduling of solutions that potentially become inferior or even infeasible

when dynamic events occur. Current approaches for solution updates mostly involve re-optimizing the newly generated DCARP instance from scratch. Liu et al. [8] proposed a memetic algorithm consisting of a random key crossover and three local search operators and with a distance-based split scheme to re-optimize the new DCARP instance. This algorithm restarts with both random solutions and solutions generated by the path-scanning heuristic. In [9], Tong et al. proposed two initialization strategies including restart and sequence transfer strategies in the generalized optimization framework for DCARP. The sequence transfer strategy concatenates all remaining routes and constructs an ordered list of tasks. Then, the tasks that have been served are removed from the list and new tasks are inserted into the list greedily. Finally, a transferred solution is generated by using the split scheme to convert the ordered list to a CARP solution.

In addition, a few works also focused on handling specific dynamic events with corresponding strategies. For example, Padungwech et al. [32] considered only new task dynamic events and applied a random insertion operator to handle them. New tasks were arranged in random order and inserted into the current solution greedily, followed by Tabu search to improve the newly generated solution. Nagy et al. [13] proposed a novel re-routing algorithm that updates at most one or two routes in the current solution. They focused on dealing with new tasks, which are greedily added to the route that results in the smallest increase in costs, ensuring that the resulting route remains feasible. Otherwise, these new tasks form a new route which is then assigned to a new vehicle. Based on this strategy, they recently proposed an artificial bee colony algorithm for dynamic CARP instances [14]. Monroy-Licht et al. [33] considered only the disruption costs for optimizing the DCARP instance with broken-down vehicles. The remaining tasks were divided into two sets: one consists of unserved tasks belonging to active vehicles, and another comprising tasks from failed vehicles. Subsequently, tasks from failed vehicles were reassigned to the closest active vehicle's task. Routes for those active vehicles that received new tasks initially assigned to failed vehicles were rescheduled, and remaining active vehicles remained fixed.

B. Motivations

As reviewed above, current literature on dynamic optimization for DCARP mostly involves re-optimizing the new DCARP instance from scratch, except for some strategies tailored to very specific dynamic events. However, for DCARP scenarios with general dynamic events, the significant knowledge contained in the solutions or search process of the optimization for the former DCARP instance has not been effectively utilized. The only previous DCARP work that attempted to benefit from experience for general dynamic events was the work that proposed a sequence transfer strategy [9]. However, it only transferred knowledge captured by the best solution (i.e., executed solution). This was beneficial for individual-based optimization algorithms that maintain a single solution over time. However, it would bring limited to no benefit on population-based optimization algorithms, where a diverse population is usually required to improve optimization. Meta-heuristic algorithms typically search for many high-quality solutions that are not selected to update the service routes. These high-quality solutions of the former DCARP instance are discarded in existing works, but they also have the potential to promote dynamic optimization for the new DCARP instance. Knowledge transfer strategies have also been applied to assist in solving static CARP [34], [35]. Nevertheless, they are also not suitable for DCARP due to either their heavy computational burden for learning the transfer model or the exclusive applicability to tree-based representations. More recently, Ardeh et al. [36], [37] and Wang et al. [38] also transferred knowledge from previously solved problems when solving uncertain CARP using genetic programming. However, they focused on the tree-based representation making their knowledge transfer strategies currently only applicable for genetic programming approaches. Moreover, evolutionary multitasking is also popular for transfer useful knowledge among different tasks to improve the performance of the algorithm on all tasks. For example, Qiao et al. [39] transferred solutions among different tasks to enhance the global and local diversity of the main task.

Therefore, in this paper, we propose a dynamic optimization framework for the DCARP scenario with a solution building block adaptation strategy to extract valuable experience contained in historical solutions generated by the previous optimization update.

IV. Dynamic optimization framework based on solution building block adaptation

A dynamic optimization framework based on a novel solution building block adaptive (SBBA) strategy is proposed in this section. Its structure is presented in Figure 1.

The dynamic optimization process starts with an initial static CARP instance. A meta-heuristic optimization algorithm can be employed to obtain the best solution for the static CARP instance, which will be used to schedule vehicles to serve the pre-defined tasks. Subsequently, a control center will detect changes in the environment [29]. Whenever any dynamic events occur while vehicles are in service, the control center will report the type of dynamic events and generate a new DCARP instance according to the occurred dynamic events. The meta-heuristic algorithm will then dynamically optimize the newly generated DCARP instance. After the dynamic optimization process for the new DCARP instance, an updated solution will be obtained and deployed to the environment to update the service plan of all vehicles. The dynamic optimization will stop only when all tasks have been served.

In our framework, the SBBA strategy adapts the historical solutions and uses them as initial solutions for the meta-heuristic algorithm. Since different types of dynamic events can result in DCARP instances with varying degrees of difficulty to solve [40], we consider two different types of DCARP instances in this paper:

1) DCARP instances with only cost-changing events (DCARP-OC). Cost-changing dynamic events include the increase and decrease of road deadheading costs.

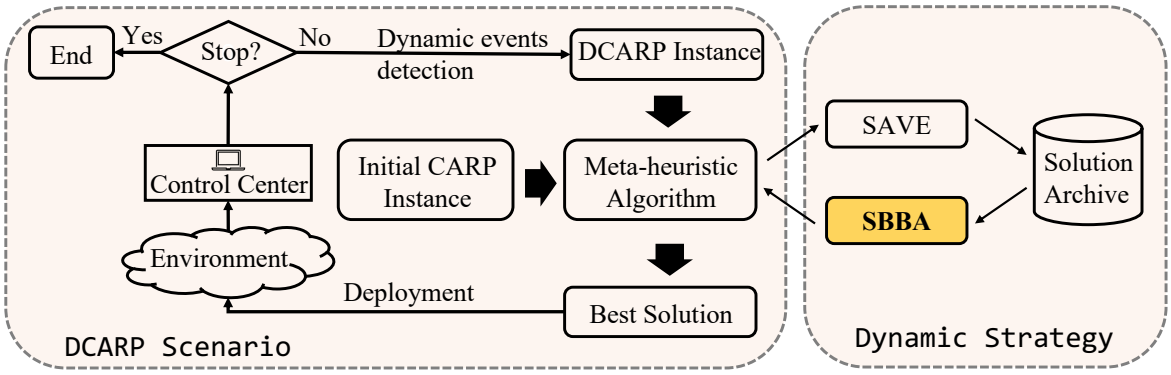


Fig. 1: The structure of the SBBA-based dynamic optimization framework for the DCARP scenario. The left part is an illustration of the DCARP scenario, where dynamic optimization process starts with an initial static CARP instance and the control center will detect changes in the environment and determine to generate DCARP instances or not. The dynamic strategy at the right is used to assist the optimization for DCARP instances.

2) DCARP instances with both cost-changing and task-changing dynamic events (DCARP-CT). Task-changing dynamic events include added and deleted tasks.

To handle these different types of DCARP instances, specific strategies in SBBA are applied to adapt the historical solutions for promoting dynamic optimization. All these strategies are based on the new concept of *building block*. Therefore, in the following subsections, we will introduce the concept of the building block first and then provide details on the specific strategies.

A. Building Blocks as Experiences

The new DCARP instance is generated based on the former DCARP instance and the deployed solution. However, except for the best solution obtained in the optimization process for the former DCARP instance, historical high-quality solutions cannot be used directly to schedule the vehicles' service plans in the new DCARP instance. Nevertheless, the information contained in these solutions can be extracted and reused. For instance, it is highly likely that two adjacent tasks in high-quality historical solutions will still be neighbors in the best solution of the new DCARP instance. Therefore, such sequence patterns contained in historical solutions can be extracted to assist in constructing feasible high-quality solutions for the new DCARP instance.

In this paper, we propose the concept of the "building block" 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 previous CARP solution is still required to be served in the new DCARP instance, we combine this sub-sequence of tasks into a building block. We assume that such a task sequence that exists in the high-quality solutions of the former DCARP instance will still remain in the high-quality solutions of the new DCARP instance.

The pseudo-code for constructing building blocks from a historical solution is presented in Algorithm 1. It receives as input a historical solution and the set of tasks to be served in the new DCARP instance. Each building block has five properties, including the block's task sequence, head node, tail node, total serving costs, and total demands. In each historical solution, consecutive sequences of tasks still required to be served in the current DCARP instance are detected and combined into building blocks as shown in Line 4 $\sim$ Line 16. These consecutive sequences of tasks are the sequence of the building block. Once a task in the historical solution is not required to be served in the current DCARP instance, the construction of the current building block is finished. The head node and tail node of this building block are the head and tail nodes of this task sequence. The cost and demand of the building block are equal to the total costs of serving this task sequence and the total demands of all tasks in this task sequence. Then, this building block is added to the set of building blocks $BLOCK$ (Line 12) and a new building block is re-initialized (Line 13). In addition, if a task in a route has both its predecessor and successor tasks served or deleted in the new DCARP instance, we also make this independent task a building task. Finally, the construction of building blocks for a historical solution will terminate until all tasks have been checked, and then, all detected building blocks form a set of building blocks for the solution construction. Additionally, no new parameter has been introduced for the construction of building blocks.

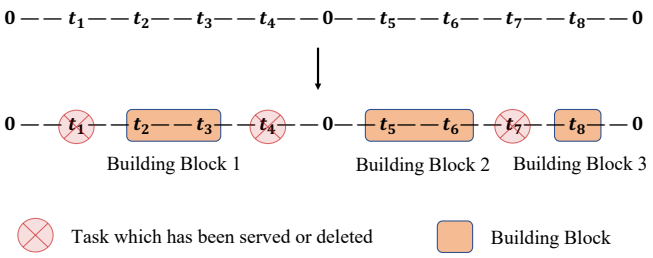


Fig. 2: An example of building block construction from an archived solution. The task sequence depicted above represents an archived solution.

To clearly illustrate how to construct building blocks from a historical solution, Figure 2 provides an example where the task sequence shown in the top is a historical solution. The

Algorithm 1: The pseudo-code of constructing building blocks.

Input: The historical solution: S; The task set E_R

1 **Function** `ConstructBuildingBlocks(`S, E_R`)`
2 Initialize an empty set for building blocks $BLOCK = \varnothing$.
3 Initialize a building block: bb, where $bb.seq = [\ ]$.
4 **for** *each route* $r_k \in S$ **do**
5 **for** i *from* 1 *to* l_k **do**
6 **if** $t_{k,i} \notin E_R$ **then**
7 **if** $bb.seq$ *is not empty* **then**
8 Set $bb.head = head_{bb.seq[0]}$.
9 Set $bb.tail = tail_{bb.seq[-1]}$.
10 Set $bb.cost$ equals to the total serving costs of $bb.seq$.
11 Set $bb.demand$ equals to the total demands of all tasks in $bb.seq$.
12 $BLOCK = BLOCK \cup bb$.
13 Re-initialize bb that $bb.seq = [\ ]$.
14 continue;
15 **else**
16 $bb.seq$.append($t_{k,i}$).
17 **return** $BLOCK$;

Output: The set of building blocks: $BLOCK$

tasks t_1, t_4, and t_7 have already been served or deleted, while t_2, t_3, t_5, t_6, and t_8 are still required to be served in the new DCARP instance. Consequently, building blocks $[t_2, t_3]$, $[t_5, t_6]$, and $[t_8]$ are formed. Taking building block $[t_2, t_3]$ as an example, its head node is the head node of t_2, and its tail node is the tail node of t_3. The cost of this building block is the sum of the total traversing costs from its head node to tail node and the total service costs of t_2 and t_3. The demand of this building block is equal to the total demands of t_2 and t_3.

B. SBBA for DCARP-OC Instances

For DCARP-OC instances, the SBBA strategy generates a new population of solutions for the new DCARP instances from an archive of historical solutions. The pseudo-code of SBBA for DCARP-OC instances is presented in Algorithm 2.

The archived solutions AS are a set of high-quality solutions saved from the optimization process for the former DCARP instance. Different definitions for "high-quality" could be adopted. In our experiments later on, we consider the top l_{AS} solutions as being high-quality solutions, where l_{AS} is a pre-defined parameter. For each historical solution S_i, we first obtain a set of building blocks using Function $ConstructBuildingBlocks(S, E_R)$ in Algorithm 1, where E_R is the task set for the new DCARP instance as shown in Line 3. Since the path-scanning constructive heuristic was effective for generating solutions for the set of tasks. Therefore, we regarded each building block as a task which has head node, tail node, cost and demand, and applied the path-scanning to the obtained building blocks to generate a new solution NS_i as shown in Line 4. If the current adaptive solution set does not contain NS_i, then it is added to the adaptive solution set as one of the adaptive solutions as shown in Line 6. Finally, the new adaptive solution set generated from archived solutions which belong to the previous DCARP instance is returned, and used to help the solution initialization for new DCARP instance. For our SBBA strategy for DCARP-OC instances, only a single parameter needs to be predetermined: specifically, the size of the set of archived solution l_{AS}.

Algorithm 2: SBBA for DCARP-OC instances

Input: The set of archived solutions: AS; The task set: E_R.

1 Initialize an empty solution set $P = \varnothing$.
2 **for** *each solution* $S_i \in AS$ **do**
3 $BLOCK$ = `ConstructBuildingBlocks(`S_i, E_R`)`.
4 Apply Path-Scanning algorithm to $BLOCK$ and obtain a new solution NS_i.
5 **if** $NS_i \notin P$ **then**
6 $P = P \cup NS_i$.

Output: Adaptive solution set P

C. SBBA for DCARP-CT Instances

Different from the DCARP-OC instances, DCARP-CT instances require considering changing tasks, especially newly added tasks and how to assign them to existing or new routes. Therefore, a building block based constructive strategy and an insertion strategy are proposed to handle changing tasks in SBBA. The pseudo-codes of SBBA for DCARP-CT instances are presented in Algorithm 3.

First, the newly added tasks are filtered out from all tasks E_R of the current DCARP instance to form a new task set $NE_R \subset E_R$ as shown in Line 1. Then, the building block based constructive (Line 3 $\sim$ Line 17) and insertion strategies (Line 19 $\sim$ Line 37) are used to adapt historical solutions to the new environment from different perspectives.

The building block based constructive strategy (Line 3 $\sim$ Line 17) is similar to SBBA for DCARP-OC instances, which considers building blocks in historical solutions. However, SBBA for DCARP-CT instances also considers each new task as an independent building block as shown in Line 5 $\sim$ Line 11. Therefore, both the sequence pattern in historical solutions and the new tasks are considered for generating new solutions as shown in Line 14. Finally, the path-scanning algorithm is used to generate new solutions as shown in Line 15.

The insertion strategy considers the complete sequence pattern of the historical solution and inserts the new task into the historical solution to obtain the new solution (Line 19 $\sim$ Line 37). For each historical solution, tasks that have been served or deleted are first deleted from the solution's task sequence as shown in Line 23. This results in an incomplete solution that contains information about the historical solution. Since the minimal number of vehicles k_0 can be determined based on the instance's information (i.e., $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$ where

Algorithm 3: SBBA for DCARP-CT instances

Input: The set of archived solutions: AS; The task set: E_R; The capacity Q.

1 Filter the new task set $NE_R \subset E_R$.
2 /* Building block based construction */
3 Initialize an empty set of building blocks $BLOCK = \varnothing$.
4 Initialize a building block: bb, where $bb.seq = [\,]$.
5 **for** *each task* $tk \in NE_R$ **do**
6 Set $bb.seq.append(tk)$.
7 Set $bb.head = head_{tk}$.
8 Set $bb.tail = tail_{tk}$.
9 Set $bb.cost = sc(tk)$.
10 Set $bb.demand = dm(tk)$.
11 $BLOCK = BLOCK \cup bb$.
12 Initialize an empty set $P = \varnothing$.
13 **for** *each solution* $S_i \in AS$ **do**
14 $BLOCK' = BLOCK \cup$ `ConstructBuildingBlocks(`S_i, E_R`)`.
15 Apply Path-Scanning algorithm to $BLOCK'$ and obtain a new solution NS_i.
16 **if** $NS_i \notin P$ **then**
17 $P = P \cup NS_i$.
18 /* Insertion */
19 Set W_T as the total demands of all tasks in E_R.
20 Set $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$.
21 **for** *each solution* $S_i \in AS$ **do**
22 Set a new solution $NS_i = S_i$.
23 Remove all served tasks and deleted tasks in NS_i.
24 Set K as the number of routes in NS_i.
25 **if** $K < k_0$ **then**
26 Add $k_0 - K$ empty routes into NS_i.
27 Set a copy of new task set $CNE_R = NE_R$.
28 **while** $|CNE_R| > 0$ **do**
29 $t, r =$ task and route with the smallest increasing cost for the insertion.
30 **if** $t, r = NULL$ **then**
31 // No route is feasible for any tasks in CNE_R
32 Add a new empty route into NS_i.
33 **else**
34 Insert t into r in NS_i.
35 Remove t from CNE_R.
36 **if** $NS_i \notin P$ **then**
37 $P = P \cup NS_i$.

Output: Adaptive solution set P

W_T is the total demands of all tasks), we add several empty routes to the current incomplete solution if the number of routes in the current incomplete solution is smaller than the minimum required number of vehicles as shown in Line 24 $\sim$ Line 26. Then, the new tasks are inserted into the current incomplete solution to obtain a new feasible solution in a greedy way, such that the task resulting in the smallest increase in solution cost is added first as shown in Line 28 $\sim$ Line 35. If no route is feasible for any of the remaining tasks to be inserted, a new empty route is inserted into the current solution, and the greedy insertion process continues as shown in Line 30 $\sim$ Line 32.

After using the two different strategies, a new adaptive solution set is obtained to assist the solution initialization for the new DCARP instance. Similarly, our SBBA strategy for DCARP-CT instances also only needs to predetermine the size of the set of archived solution l_{AS}.

D. Discussion

Our work shows that a relatively simple building block strategy can be very effective to update solutions to dynamic events under limited optimization time, as will be shown in Section V. In particular, it is able to explicitly and efficiently extract valuable information from historical solutions in the former environment, particularly in DCARP-OC scenarios where no new tasks are inserted. However, for DCARP-CT scenarios involving the insertion of new tasks, our approach currently has no optimization experience based mechanism for assisting insert these new tasks to proper positions. As a result, the proposed strategies might be less effective if there are a large number of new tasks, which was demonstrated in Section V-E. In addition, the SBBA strategy proposed in this study is based on the building blocks which are subsequences of tasks derived from the historical solution. Naturally, when remaining tasks are dispersed throughout a historical solution, the resulting building blocks tend to be shorter and contain only a few tasks. Consequently, the SBBA strategy may not benefit from such less informative building blocks. Therefore, a valuable future direction is to investigate how to extract useful implicit knowledge from past optimization to enhance the insertion of new tasks. The efficacy of proposed strategies will be demonstrated and analyzed in the following experimental section.

V. Experimental Studies

In this section, the effectiveness of proposed SBBA strategy are evaluated in a series of different DCARP scenarios. Then, we embed two state-of-the-art meta-heuristic algorithms into our proposed DO-SBBA framework and evaluate their performance in a series of different DCARP scenarios as well. The empirical results are presented and analyzed in this section.

A. Experimental Settings

All experiments were conducted on a series of DCARP scenarios generated by the simulation system proposed in [9] based on a static CARP benchmark, namely the *EGL* set[1] [42]. The set consists of 22 different static CARP instances, including 10 *EGL-G* instances with 200 initial tasks and 12 *EGL-S* instances with 100 initial tasks.

[1]The dimensionality of other existing benchmarks, such as GDB or VAL, is too small to effectively assess the performance of our proposed strategy and framework.

TABLE I: Results of generated solutions' quality for SBBA, PS[41], GRD[20] and RR1[13] strategies on DCARP-OC (left part) and DCARP-CT (right part) scenarios from the EGL dataset. The value in each cell represents "MEAN $\pm$ STD" of NC for all DCARP instances in 10 generated DCARP scenarios for each map. The highlighted values denote the best results among strategies under the Friedman test with a significance level of 0.05.

MapName	DCARP-OC Scenarios			DCARP-CT Scenarios			
	SBBA	PS	GRD	SBBA	PS	GRD	RR1
egl-g1-A	***0.2259 ± 0.1001***	0.5797 ± 0.0935	0.5075 ± 0.0883	***0.3502 ± 0.0723***	0.5369 ± 0.0907	0.4729 ± 0.0732	0.4838 ± 0.1498
egl-g1-B	***0.2588 ± 0.1347***	0.6445 ± 0.0942	0.5637 ± 0.1060	***0.3989 ± 0.0926***	0.5551 ± 0.0964	0.4678 ± 0.0758	0.5499 ± 0.1844
egl-g1-C	***0.2213 ± 0.1254***	0.6213 ± 0.0979	0.5504 ± 0.0973	***0.4288 ± 0.1067***	0.5369 ± 0.0972	0.4995 ± 0.0729	0.6239 ± 0.1787
egl-g1-D	***0.2330 ± 0.1159***	0.6184 ± 0.1062	0.5431 ± 0.1057	***0.4728 ± 0.1023***	0.5826 ± 0.0900	0.4981 ± 0.0805	0.6664 ± 0.2042
egl-g1-E	***0.2664 ± 0.1331***	0.6432 ± 0.1166	0.5801 ± 0.0984	***0.4750 ± 0.1143***	0.5898 ± 0.1067	0.4904 ± 0.0815	0.6364 ± 0.1699
egl-g2-A	***0.2257 ± 0.1249***	0.5861 ± 0.1128	0.5222 ± 0.1005	***0.3698 ± 0.0907***	0.5140 ± 0.1039	0.4166 ± 0.0879	0.5103 ± 0.1307
egl-g2-B	***0.2560 ± 0.1212***	0.6065 ± 0.1014	0.5569 ± 0.0968	***0.4093 ± 0.0944***	0.5634 ± 0.0964	0.4869 ± 0.0785	0.5605 ± 0.1688
egl-g2-C	***0.2248 ± 0.1133***	0.6027 ± 0.0982	0.5434 ± 0.0903	***0.4201 ± 0.1069***	0.5728 ± 0.1129	0.5017 ± 0.0802	0.5866 ± 0.1877
egl-g2-D	***0.2296 ± 0.1102***	0.6038 ± 0.1137	0.5296 ± 0.1146	***0.4620 ± 0.1076***	0.5708 ± 0.0929	0.5190 ± 0.0838	0.6781 ± 0.1768
egl-g2-E	***0.2498 ± 0.1356***	0.6574 ± 0.1159	0.5820 ± 0.1266	***0.4586 ± 0.1039***	0.6288 ± 0.0926	0.5305 ± 0.0852	0.6254 ± 0.1632
egl-s1-A	***0.1876 ± 0.1350***	0.6020 ± 0.1979	0.5710 ± 0.1940	***0.3935 ± 0.1028***	0.5755 ± 0.1401	0.5164 ± 0.1372	0.7152 ± 0.2875
egl-s1-B	***0.1396 ± 0.0892***	0.6955 ± 0.1437	0.6202 ± 0.1565	***0.4228 ± 0.1111***	0.6352 ± 0.1431	0.5743 ± 0.1405	0.6686 ± 0.2345
egl-s1-C	***0.0921 ± 0.0759***	0.6481 ± 0.2181	0.5618 ± 0.2241	***0.4655 ± 0.1347***	0.6461 ± 0.1355	0.5924 ± 0.1440	0.7597 ± 0.2716
egl-s2-A	***0.1966 ± 0.1380***	0.6167 ± 0.1843	0.5646 ± 0.1790	***0.3654 ± 0.1100***	0.5616 ± 0.1372	0.5008 ± 0.1132	0.6234 ± 0.2657
egl-s2-B	***0.1487 ± 0.1321***	0.6189 ± 0.2048	0.5579 ± 0.2036	***0.4060 ± 0.1278***	0.5935 ± 0.1428	0.5301 ± 0.1454	0.6694 ± 0.2506
egl-s2-C	***0.1601 ± 0.1022***	0.6343 ± 0.1593	0.5651 ± 0.1835	***0.4640 ± 0.1375***	0.6672 ± 0.1520	0.6081 ± 0.1705	0.7804 ± 0.2813
egl-s3-A	***0.2017 ± 0.1371***	0.6428 ± 0.1519	0.5684 ± 0.1482	***0.3589 ± 0.1214***	0.5513 ± 0.1353	0.5015 ± 0.1304	0.5623 ± 0.2471
egl-s3-B	***0.1258 ± 0.0913***	0.6505 ± 0.1458	0.6086 ± 0.1483	***0.4109 ± 0.0976***	0.6257 ± 0.1278	0.5732 ± 0.1303	0.6997 ± 0.2102
egl-s3-C	***0.1436 ± 0.1086***	0.6656 ± 0.1634	0.5180 ± 0.1762	***0.4311 ± 0.1289***	0.6248 ± 0.1520	0.5546 ± 0.1680	0.7405 ± 0.2511
egl-s4-A	***0.1530 ± 0.1169***	0.5596 ± 0.1678	0.4838 ± 0.1606	***0.4026 ± 0.1514***	0.5943 ± 0.1513	0.5179 ± 0.1665	0.6546 ± 0.2232
egl-s4-B	***0.1592 ± 0.1081***	0.5946 ± 0.1903	0.5573 ± 0.1841	***0.4126 ± 0.1370***	0.6030 ± 0.1286	0.5199 ± 0.1317	0.6731 ± 0.2580
egl-s4-C	***0.1468 ± 0.1089***	0.6061 ± 0.1597	0.5444 ± 0.1613	***0.4320 ± 0.1157***	0.6205 ± 0.1341	0.5411 ± 0.1133	0.7639 ± 0.2971

To simulate cost-changing dynamic events, a base cost ($base\_cost$) is assigned to each edge in the map to indicate the minimal cost of the edge. When simulating the cost-changing dynamic events, the deadheading cost (dc) of an edge is determined by the following equation:

$$dc = \begin{cases} dc, & p < 0.25 \\ base\_cost \times r, & 0.25 \leq p \leq 0.75 \\ base\_cost, & p > 0.75 \end{cases} \tag{4}$$

where $p \in U(0,1)$ is a random number drawn from a standard uniform distribution. For each edge in the map, if $p < 0.25$, its cost remains the same, indicating there is no dynamic event on the edge. If $p > 0.75$, the edge's cost becomes $base\_cost$. Otherwise, the edge's cost becomes r times the base cost, where $r \in U(1, C)$ is also a random number drawn from a uniform distribution with an upper bound value of C. In our experiments, C is set to 5.

Then, to simulate task-changing dynamic events, a vanishing probability of $p_v = 0.2$ is assigned to each remaining task when generating new DCARP instances. To simulate newly added tasks, a predefined set of available arcs is used, and N_{NE_R} arcs are randomly selected from the set to become the new tasks for the new environment. The value of N_{NE_R} is determined by the following equation:

$$N_{NE_R} = p_t \times N_{RT}, \tag{5}$$

where N_{RT} denotes the number of remaining tasks when dynamic events occur, and p_t is a factor used to control the size of the new task dynamic events. In our experiments, p_t is set to 0.2.

We generated 10 different DCARP scenarios independently for each static CARP instance. In our simulation, after optimization for a DCARP instance with limited time (1 minute in our experiments), the best-obtained solution will be used to update the schedule of vehicles. Then, the next DCARP instance will be generated after the current solution has been executed for a random duration. We generated up to 5 DCARP instances in each DCARP scenario. If the number of tasks in a new DCARP instance is less than 20, we also stopped generating new instances.

Besides, to investigate the generalization ability of the proposed SBBA, we also generated a series of DCARP scenarios with varying severity of dynamic changes. Specifically, in the DCARP-OC scenarios, the severity of cost changes is controlled by the parameter C, and we investigate six different settings with $C \in \{1, 3, 5, 10, 15, 20\}$. For the DCARP-CT scenarios, we focused on investigating different numbers of added tasks in our analysis. Thus, we generate six different severity of added tasks with $p_t \in \{0.1, 0.2, 0.4, 0.5, 0.6, 0.8\}$. For simplicity, we only used the static CARP instance $egl-g1-A$ for this analysis. In the DCARP-CT scenarios, the parameter C for cost changes is set to 5.

In our experiment, two optimization algorithms are embed-

ded into the proposed DO-SBBA framework, including the memetic algorithm with extended neighbor search (MAENS) [20] and the global repair operator-based Tabu search algorithm (RTS) [43] embedded with the virtual task strategy [9], which are state-of-the-art optimization algorithms for solving DCARP. MAENS is a population-based meta-heuristic algorithm and its population size is set as $psize = 30$ which is same as its original paper [20]. The number of archived top solutions l_{AS} is also set as 30 in our experiments. For MAENS, if the number of adaptive solutions is higher than the predefined population size $psize$, the best $psize$ solutions are selected as the initial solutions. Otherwise, if the number of adaptive solutions is smaller than the predefined population size due to removing repeated solutions, the necessary number of additional solutions is generated by the optimization algorithm's initialization scheme to form a complete population. Conversely, RTS is an individual-based meta-heuristic algorithm and only the best among all adaptive solutions is used, as $psize = 1$. All parameters for these two algorithms are set as specified in their original papers [20], [43].

To make a fair comparison in our experiments, for each DCARP instance optimized by all meta-heuristic algorithms, the maximum optimization time was set to 1 minute [14], as it is desirable to update the DCARP solution quickly in real world DCARP problems when dynamic events influence the current solution ensuring efficient service continuity. Additionally, we executed 25 independent runs in our experiments. All programs were implemented in C language and ran on a Linux server with an AMD Ryzen Threadripper PRO 3995WX 64-Cores 2.7GHZ processor. All static CARP instances, generated DCARP scenarios, and algorithms' source codes are available online[1].

Upon completion of the optimization process, the best-obtained solution will be employed to update the vehicle schedules. In our simulation, a new DCARP instance is generated after the current solution has been executed for a specified duration.

B. Performance Measurement

Since the generated DCARP instances in a DCARP scenario can vary significantly in scale, we defined a normalized cost (NC) for each DCARP instance using the following equation:

$$NC = \frac{TC - TC_{min}}{TC_{max} - TC_{min}} \tag{6}$$

where TC is the cost of the solution that needs to be normalized, and TC_{max} and TC_{min} are the maximum and minimum costs, respectively, used for normalization in the corresponding DCARP instance. The specific values for TC_{max} and TC_{min} depend on the set of experiments, which will be discussed in Section V-C.

Furthermore, since a DCARP scenario comprises a series of DCARP instances, we have employed an average normalized cost (ANC) for dynamic optimization [15] to compare the performance of different algorithms/strategies for a DCARP scenario. The ANC of an algorithm/strategy for a DCARP scenario is calculated using the following equation:

$$ANC = \frac{1}{M}\sum_{m=1}^{M} NC_m \tag{7}$$

where M is the number of instances, and NC_m is the normalized cost of the corresponding algorithm/strategy in the m^{th} instance of the DCARP scenario. In our experiments, smaller values of NC and ANC indicate better performance.

C. Effectiveness of the SBBA Strategy

The SBBA strategy was proposed to adapt historical solutions for extracting and leveraging the optimization experience in DCARP, addressing the first part of RQ1. In this section, we compare the quality of solutions obtained by the SBBA strategy with several constructive heuristics to answer the second part of RQ1, investigating how effective is the proposed SBBA and the reason for its effectiveness. The path-scanning (PS) [44], greedy generation (GRD) [20], and one recent strategy: rerouting algorithm (RR1) [13] are employed for comparison. We use the same parameters as given in the original papers [20], [43] for these strategies. The former two conventional strategies are still the most popular and widely-adopted initialization mechanisms in the meta-heuristic algorithms for optimizing (D)CARP in the literature [9]. The RR1 algorithm only incorporates a strategy to address task-changing dynamic events, without any mechanisms for dynamic changes in costs. Nevertheless, it is still applicable to DCARP-CT scenarios in our work as cost-changing dynamic events only affect the cost values in the DCARP instance, such that the algorithm can still evaluate the quality of a solution. Therefore, in our experiments, our SBBA strategy is compared with RR1 algorithm only in DCARP-CT scenarios. The MAENS algorithm was used to optimize each DCARP instance, and the l_{AS} best solutions of the former DCARP instance were saved for the new DCARP instance. Thus, for a new DCARP instance, we could obtain at most l_{AS} solutions from the SBBA strategy, one solution from the PS strategy, and $psize$ solutions from the GRD strategy. To make a fair comparison, we set l_{AS} equal to $psize = 30$ in our experiments. The performance of the three strategies was evaluated by comparing their best-generated solutions.

NC performance measurement was used to compare the performance of three strategies. For each DCARP instance, in Eq.(6), TC represents the cost of the best solution obtained by the corresponding strategy, TC_{min} represents the cost of the best solution obtained by MAENS after optimization, and TC_{max} represents the cost of the worst solution among all solutions obtained by all strategies. We generated 20 different DCARP scenarios for each map, including 10 DCARP-OC scenarios and 10 DCARP-CT scenarios. For each type of DCARP scenario for each map, the mean and standard deviation (mean $\pm$ std) of all NC values for all DCARP instances of the these strategies are presented in Table I. For each map, we applied the Friedman test with a significance level of 0.05 to all NC values corresponding to the obtained solutions of the three strategies in all DCARP instances of the 10 generated

[1]https://github.com/HawkTom/DCARP-SBBA

DCARP scenarios. The best results under the hypothesis test are highlighted in bold values with a gray background. From Table I, it is clear that the quality of solutions obtained by SBBA is significantly better than the other constructive strategies, both in DCARP-OC and DCARP-CT scenarios.

We also calculate the difference in solution's quality between the SBBA and each of the other strategies in all DCARP instances for DCARP-OC and DCARP-CT scenarios (Figure 3). A higher difference implies a higher improvement in NC using the SBBA strategy. As we can see, the improvements were present in both scenarios, but were of larger magnitude for the DCARP-OC scenarios. In other words, the building blocks were particularly helpful when coping with cost-related dynamic events. The reason for the difference of algorithm's performance in these two types of DCARP scenarios is probably that the building block strategy extracts the optimization experience through utilizing task sequences belonging to former DCARP instances. As new tasks do not exist in these building blocks, the building block strategy becomes less useful when there are new tasks, as is the case in the DCARP-CT instances.

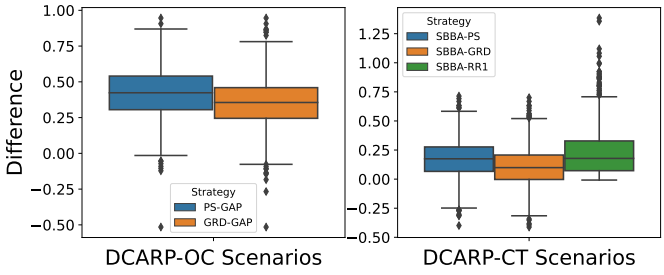


Fig. 3: The difference of the generated solution's quality between the SBBA and other strategies in all DCARP instances for DCARP-OC and DCARP-CT scenarios.

We also hypothesized that SBBA outperforms the other strategies because SBBA managed to obtain solutions that are more similar to the best solution for a DCARP instance. To test this hypothesis, a type of "similarity" (Sim) between one solution and the "best" solution was defined. The best solution is set as the best among all solutions obtained after applying MAENS to optimize the DCARP instance. The similarity was calculated using the following equation:

$$Sim = \frac{\#SameLinks}{\#TotalLinks} \tag{8}$$

where $\#TotalLinks$ is the number of links between two consecutive tasks in the best solution, and $\#SameLinks$ is the number of links that exist in both the given solution and the best solution. If more links in the best solution also exist in the given solution, the Sim value will be higher, indicating that the given solution is more similar to the best solution.

We calculated the NC and Sim values for all solutions generated by the three strategies in all DCARP scenarios and maps, and the results are presented in Figure 4. The grey points in the figure represent the data of all solutions, while the white contours represent the estimated distribution. From the figure, it is evident that the value of NC decreases as the Sim value increases. Therefore, a strategy that can generate solutions that are more similar to the best solution will have higher-quality solutions.

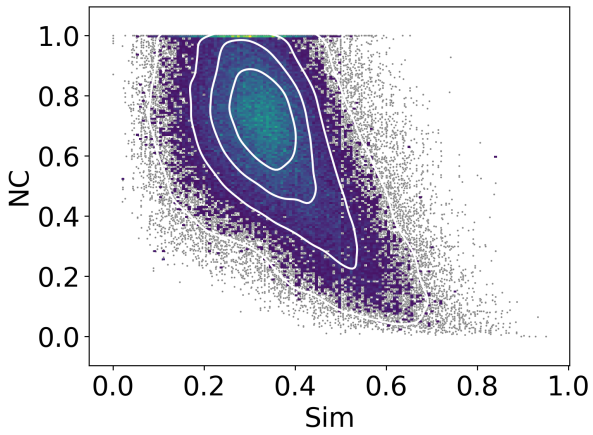
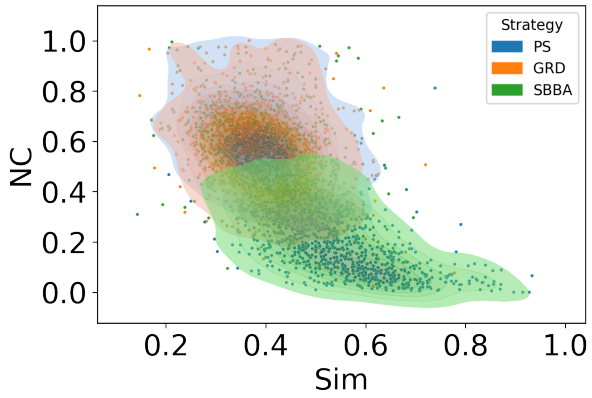
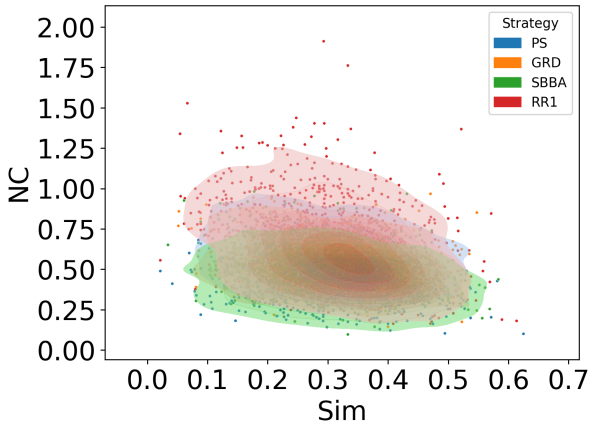


Fig. 4: Results of the NC and Sim for all solutions generated by three strategies in all DCARP scenarios of all maps. White contours represent the estimated distribution for all data.



(a) Solutions' distributions in DCARP-OC scenarios



(b) Solutions' distributions in DCARP-CT scenarios

Fig. 5: The distribution of solutions' Sim and NC for path-scanning (PS), greedy generation (GRD), SBBA and rerouting (RR1) strategies.

Then, we calculated the NC and Sim values of the best solutions obtained by each strategy in different types of DCARP scenarios (Figure 5). Figure 5a and Figure 5b show the distributions of solutions for DCARP-OC and DCARP-CT scenarios, respectively. It is evident that the solutions obtained by SBBA are located in the bottom right part for both types of DCARP scenarios, and especially for DCARP-OC. This indicates that the solutions generated by SBBA are overall more similar to the best solution and have a higher quality than the other strategies in both DCARP-OC and DCARP-CT scenarios, and especially for DCARP-OC. This is in line with the improvements in NC, which were present for both DCARP-OC and DCARP-CT, but were relatively larger for DCARP-OC (Figure 3) as discussed earlier in this section.

Additionally, our motivation for the SBBA is to retain the high-quality solutions' characteristics of the former DCARP instance by the *building block* so that the new DCARP instance can benefit from such characteristics and generate better solutions. The above results between the solution's similarity to the best solution and the solution's quality also demonstrated that our proposed SBBA indeed retains the valuable characteristics of the high-quality solutions of the former DCARP instance. SBBA is effective for generating high-quality solutions, especially in DCARP-OC scenarios (Figure 3).

D. Effectiveness of the SBBA-based Dynamic Optimization Framework

The SBBA-based dynamic optimization framework employed the SBBA strategy as the initialization strategy to promote the dynamic optimization, addressing the first part of RQ2. In this section, we applied the generated solutions by SBBA strategy to be used as the initial solutions for the meta-heuristic algorithm. Through comparing the quality of solutions obtained using these adapted meta-heuristic algorithms, we are going to answer the second part of RQ2, investigating the effectiveness of the SBBA-based dynamic optimization framework. As mentioned in Section V-A, two meta-heuristic algorithms, i.e., MAENS (population-based) [20] and RTS (individual-based) [43] with the state-of-the-art virtual task strategy for DCARP [9], are employed to evaluate the performance of proposed DO-SBBA framework.

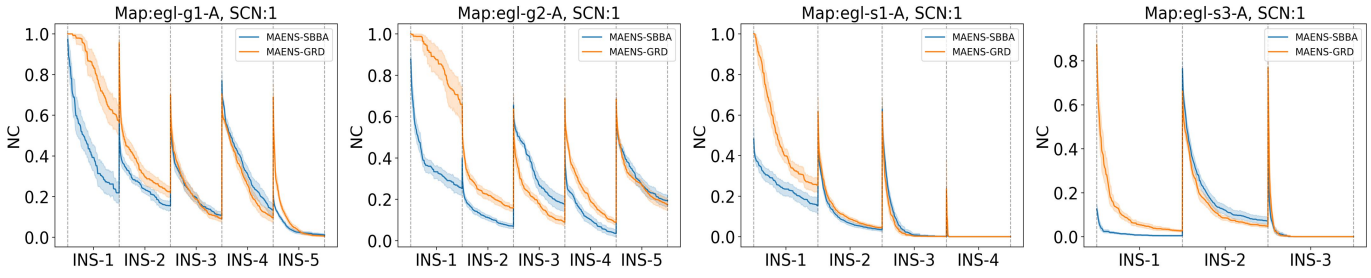
Given that the RR1 algorithm only focuses on adapting the best solution from the previous DCARP instance to the new one, we only integrated SBBA and GRD strategies into the meta-heuristic algorithm, employing them as the initialization strategy. The meta-heuristic algorithm with SBBA strategy was compared to the same algorithm with GRD strategy in both DCARP-OC and DCARP-CT scenarios. For each type of DCARP scenario, we generated 10 different DCARP scenarios for each map, and each optimization algorithm optimized each DCARP instance for 25 independent runs. The ANC was used for the performance measurement. We calculated the ANC value in each independent run for meta-heuristic algorithms with different strategies. Then, for each DCARP scenario, the Wilcoxon signed-rank test with a significance level of 0.05 was applied to all ANC values from 25 independent runs for two strategies. The results of MAENS and RTS algorithms with the SBBA and GRD strategies in DCARP-OC and DCARP-CT scenarios are presented in Table II[2]. The values in each cell represent the number of "win-draw-lose" of the SBBA strategy versus the GRD strategy under the hypothesis test over 10 different DCARP scenarios for each map. Besides, we also plotted the convergence curves of meta-heuristic algorithms with different strategies in DCARP scenarios, and some of them are presented in Figure 6. Each plot represents a complete DCARP scenario including all DCARP instances. Each segment in the plot represent the algorithm's convergence curve within the maximum optimization time in the corresponding DCARP instance.

TABLE II: Comparison of MAENS and RTS meta-heuristic algorithms using SBBA against GRD in DCARP-OC and DCARP-CT scenarios. The values in each cell represent the number of "win-draw-lose" for the SBBA strategy based on Wilcoxon signed-rank tests with a significance level of 0.05 in 10 DCARP scenarios of each map.

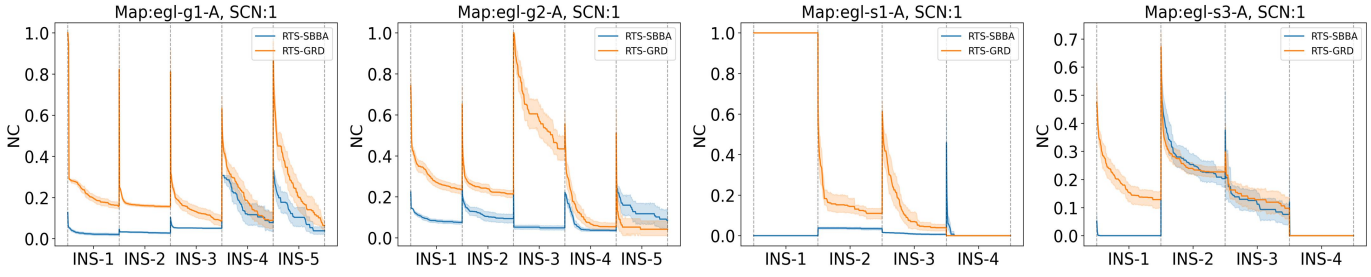
MapName	DCARP-OC Scenarios		DCARP-CT Scenarios	
	MAENS	RTS	MAENS	RTS
egl-g1-A	10-0-0	8-2-0	6-4-0	7-1-2
egl-g1-B	10-0-0	7-1-2	3-7-0	6-2-2
egl-g1-C	10-0-0	8-0-2	2-6-2	4-3-3
egl-g1-D	10-0-0	8-2-0	4-5-1	2-3-5
egl-g1-E	10-0-0	8-2-0	4-5-1	1-5-4
egl-g2-A	10-0-0	7-1-2	3-5-2	8-1-1
egl-g2-B	10-0-0	6-2-2	4-6-0	4-3-3
egl-g2-C	10-0-0	10-0-0	5-5-0	5-2-3
egl-g2-D	10-0-0	7-0-3	5-5-0	6-1-3
egl-g2-E	10-0-0	6-3-1	6-4-0	4-3-3
egl-s1-A	4-4-2	9-1-0	4-5-1	5-1-4
egl-s1-B	3-7-0	10-0-0	4-4-2	7-0-3
egl-s1-C	8-1-1	10-0-0	2-7-1	3-4-3
egl-s2-A	8-2-0	3-5-2	1-8-1	2-7-1
egl-s2-B	5-5-0	4-5-1	1-7-2	2-7-1
egl-s2-C	3-6-1	2-8-0	2-6-2	2-8-0
egl-s3-A	6-4-0	7-3-0	3-5-2	2-5-3
egl-s3-B	7-3-0	4-6-0	3-6-1	1-8-1
egl-s3-C	5-4-1	5-5-0	2-7-1	1-9-0
egl-s4-A	5-5-0	6-4-0	2-8-0	1-6-3
egl-s4-B	5-5-0	2-5-3	0-10-0	2-7-1
egl-s4-C	4-5-1	4-6-0	4-6-0	1-7-2

Based on the results presented in Table II and Figure 6, it can be observed that the SBBA strategy improves the optimization performance of both MAENS and RTS algorithms in most DCARP scenarios, especially in DCARP-OC scenarios. However, in most DCARP-CT scenarios, the MAENS/RTS with the SBBA strategy performs similarly to MAENS/RTS with the GRD strategy. This is understandable, since the SBBA analysis shown in Section V-C showed that this strategy usually leads to larger improvements in the initial solutions for

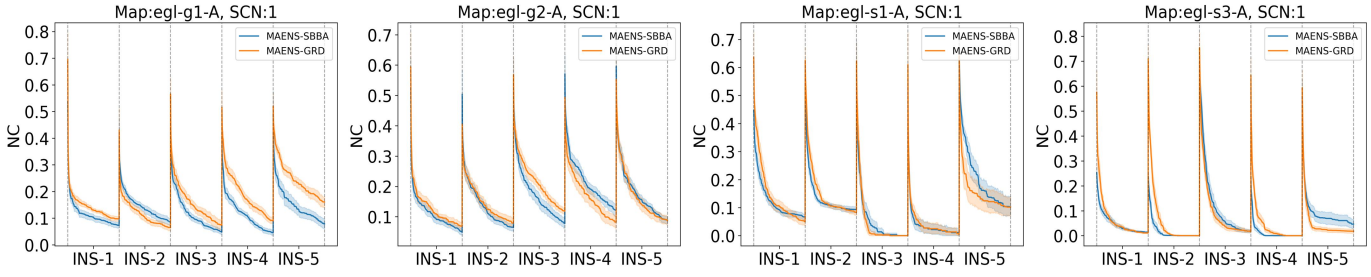
[2]Due to the page limitation, the "Mean $\pm$ Std" of ANC values for each DCARP scenario of each map are presented in the supplementary material.



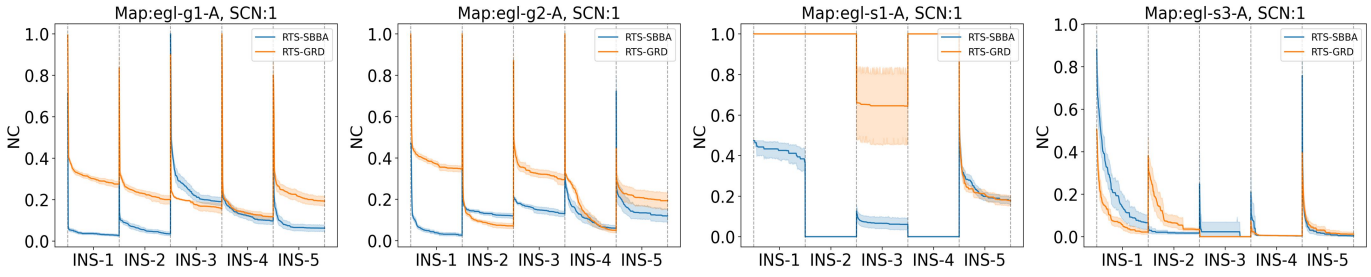
(a) MAENS for DCARP-OC scenario



(b) RTS for DCARP-OC scenario



(c) MAENS for DCARP-CT scenario



(d) RTS for DCARP-CT scenario

Fig. 6: The convergence curves of MAENS [20] and RTS [43] with SBBA (*blue lines*) and GRD (*orange lines*) strategies in some DCARP-OC and DCARP-CT scenarios.

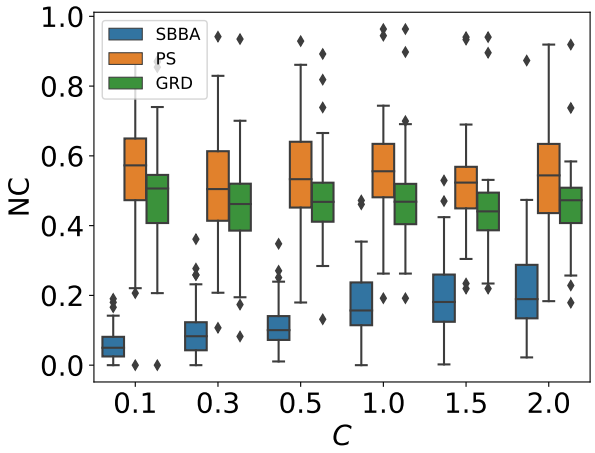
DCARP-OC than for DCARP-CT. The convergence speed is also faster with the SBBA strategy, as shown in Figure 6. This is due to the higher-quality and higher-similarity initial solutions generated by the SBBA strategy (Figure 5). The higher-quality solutions used for optimization can lead to faster convergence, and solutions with higher similarity indicate they are closer to the best solution, which can also lead to faster convergence of meta-heuristic algorithms.

The SBBA strategy is shown to be effective in promoting the performance of meta-heuristic algorithms in DCARP scenarios, especially in DCARP-OC scenarios. This indicates that the SBBA is much more effective for DCARP-OC scenarios without task changing. Additionally, in Table II, MAENS with SBBA strategy outperforms the GRD strategy in all DCARP-OC scenarios of *EGL-G* maps, which has larger number of tasks. This indicates that the MAENS with SBBA strategy significantly outperforms the GRD strategy in DCARP scenarios with more tasks because scenarios with a smaller number of tasks are too easy for MAENS so that the effectiveness of SBBA is not obvious. For RTS, since it only uses one adaptive solution for optimization, the effectiveness of SBBA is also not obvious. However, it still provides a performance boost for RTS if the adaptive solution has a higher quality according to Figure 6b and 6d.

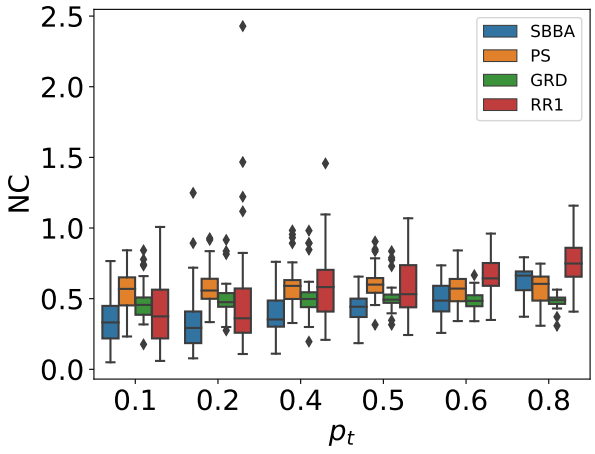
Therefore, we can conclude from our experimental results that DO-SBBA can promote the dynamic optimization performance of meta-heuristic algorithms within a limited optimization time. Specifically, the DO-SBBA is much more effective in DCARP-OC scenarios and DCARP scenarios that are harder for the original meta-heuristic algorithms to handle.

E. SBBA Strategy's Performance Under Different Severities of Dynamic Changes

To investigate the generalization ability of the proposed SBBA for answering the RQ3, we applied it to the DCARP scenarios with varying severity of dynamic changes listed in Section V-A. We employed the MAENS algorithm to optimize each DCARP instance in the dynamic optimization framework.



(a) Algorithms' performance comparison in DCARP-OC scenarios



(b) Algorithms' performance comparison in DCARP-CT scenarios

Fig. 7: The performance comparison between path-scanning (PS), greedy generation (GRD), rerouting (RR1) and SBBA strategies on DCARP scenarios with different severity of dynamic changes.

For each setting of C and p_t, which control the severity of dynamic changes, we generated 10 different DCARP scenarios independently. Smaller C and p_t indicate that smaller changes in edges' deadheading costs and fewer number of added tasks, respectively. We calculated the NC of solutions generated by different strategies in all DCARP instances of the 10 scenarios. The results are presented in the boxplots in Figure 7. For the DCARP-OC scenarios in Figure 7a, our SBBA strategy outperforms the other two algorithms, even when the severity of dynamic changes is larger. However, the performance gap between SBBA and the other two algorithms decreases as the severity of dynamic changes increases. For the DCARP-CT scenarios in Figure 7b, our SBBA strategy is slightly better than the other three constructive heuristic algorithms when the number of added tasks is small. However, when the number of added tasks becomes larger, the performance of SBBA decreases substantially and is even worse than the two conventional constructive strategies when $p_t > 0.6$, but it is still better than the RR1 algorithm.

Therefore, based on the experimental results, our proposed SBBA strategy is more effective when the severity of dynamic changes is small. This is because small dynamic changes make the new DCARP instance more similar to the previously optimized DCARP instance, causing high-quality solutions for both DCARP instances to also be similar. The SBBA strategy utilizes these similarities to generate solutions for new DCARP instances. Therefore, small dynamic changes are more suitable for our SBBA strategy. When the severity of dynamic changes becomes large, our SBBA strategy is more robust in DCARP-OC scenarios than in DCARP-CT scenarios. Furthermore, the SBBA strategy performs even worse than two conventional constructive strategies when the number of added tasks is large in DCARP-CT scenarios. This is because in good solutions of new DCARP instances, the newly added tasks and remaining tasks might form a new building block, and so the building blocks for the former DCARP instances are likely to cause a bad solution.

VI. Conclusion

In this paper, we focused on dynamic optimization for the DCARP scenarios, which considers unforeseen dynamic events occurring during the deployment of a CARP solution. We leveraged the optimization experience from the previous instance to facilitate dynamic optimization for the new instance. The *building blocks* adaptive strategy (SBBA) was proposed to extract such valuable experience from historical solutions. Then, the solution building block adaptive strategy based dynamic optimization framework (DO-SBBA) was proposed to assist meta-heuristic algorithms in solving two different DCARP scenarios: the DCARP-OC scenario containing instances generated only from cost-changing dynamic events, and the DCARP-CT scenario containing instances generated from both cost-changing and task-changing dynamic events.

To evaluate the efficacy of the proposed SBBA strategy and the optimization framework, we applied them to a series of DCARP-OC and DCARP-CT scenarios. The experimental results demonstrated that the SBBA benefits a lot from the experience contained in historical solutions and significantly outperforms conventional constructive strategies in terms of solution quality. Empirical analysis indicated that solutions

constructed by the SBBA are more similar to the best solutions with the help of the extracted experience. Furthermore, we analyzed the performance of the DO-SBBA framework for dynamic optimization by embedding two meta-heuristic algorithms, MAENS [20] and RTS [43]. The experimental results showed that the DO-SBBA framework led to fast convergence and helped the original algorithm obtain better solutions for both algorithms in both DCARP-OC and DCARP-CT scenarios. The proposed framework was particularly helpful for DCARP-OC scenarios, with improvements in performance being observed across different severity of change in cost. For DCARP-CT scenarios, improvements in performance were larger for smaller changes in the number of added tasks.

In the future, more effective strategies could be proposed for DCARP-CT instances. Since SBBA directly inherits task sequences from historical solutions, it would be valuable to extract more indirect experience from historical solutions to help generate higher-quality solutions for DCARP scenarios in the future. Furthermore, considering the observed performance degradation of our algorithm on complex DCARP scenarios, it would also be worthwhile to propose novel approaches for complex and large-scale DCARP.

ACKNOWLEDGEMENTS

This work was supported by the Honda Research Institute Europe (HRI-EU), the National Natural Science Foundation of China (Grant No. 62250710682), an internal grant from Lingnan University, the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386).

REFERENCES

[1] Y. Zhang, Y. Mei, S. Huang, X. Zheng, and C. Zhang, "A route clustering and search heuristic for large-scale multidepot-capacitated arc routing problem," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8286–8299, 2021.

[2] 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.

[3] X. Jin, H. Qin, Z. Zhang, M. Zhou, and J. Wang, "Planning of garbage collection service: An arc-routing problem with time-dependent penalty cost," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2692–2705, 2020.

[4] 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.

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

[6] Á. Corberán, R. Eglese, G. Hasle, I. Plana, and J. M. Sanchis, "Arc routing problems: A review of the past, present, and future," *Networks*, vol. 77, pp. 88–115, jun 2020.

[7] 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.

[8] 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.

[9] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "A novel generalized metaheuristic framework for dynamic capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1486–1500, 2022.

[10] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "Benchmarking dynamic capacitated arc routing algorithms using real-world traffic simulation," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, jul 2022.

[11] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "A hybrid local search framework for the dynamic capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, jul 2021.

[12] 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.

[13] Z. Nagy, Á. Werner-Stark, and T. Dulai, "A data-driven solution for the dynamic capacitated arc routing problem," *Proceedings of IAC in Budapest 2021*, pp. 64–83, 2021.

[14] Z. Nagy, Á. Werner-Stark, and T. Dulai, "An artificial bee colony algorithm for static and dynamic capacitated arc routing problems," *Mathematics*, vol. 10, no. 13, p. 2205, 2022.

[15] 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.

[16] 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.

[17] S. Yang, Y. Jiang, and T. T. Nguyen, "Metaheuristics for dynamic combinatorial optimization problems," *IMA Journal of Management Mathematics*, vol. 24, no. 4, pp. 451–480, 2013.

[18] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades—part a," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, 2021.

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

[20] 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.

[21] 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.

[22] B. H. O. Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos, "Recent dynamic vehicle routing problems: A survey," *Computers & Industrial Engineering*, vol. 160, p. 107604, Oct. 2021.

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

[24] N. R. Sabar, A. Bhaskar, E. Chung, A. Turky, and A. Song, "A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion," *Swarm and evolutionary computation*, vol. 44, pp. 1018–1027, 2019.

[25] Y.-H. Jia, Y. Mei, and M. Zhang, "A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10855–10868, 2021.

[26] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[27] M. Mavrovouniotis, S. Yang, M. Van, C. Li, and M. Polycarpou, "Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 15, no. 1, pp. 52–63, 2020.

[28] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic tsp," in *International workshop on ant algorithms*, pp. 88–99, Springer, 2002.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between cvrp and carp," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 644–658, 2014.
[35] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems," *Memetic Computing*, vol. 7, pp. 159–180, 2015.
[36] M. A. Ardeh, Y. Mei, and M. Zhang, "Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 765–779, 2021.
[37] M. A. Ardeh, Y. Mei, M. Zhang, and X. Yao, "Knowledge transfer genetic programming with auxiliary population for solving uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 311–325, 2022.
[38] S. Wang, Y. Mei, and M. Zhang, "Explaining genetic programming-evolved routing policies for uncertain capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, 2023.
[39] K. Qiao, J. Liang, Z. Liu, K. Yu, C. Yue, and B. Qu, "Evolutionary multitasking with global and local auxiliary tasks for constrained multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 10, pp. 1951–1964, 2023.
[40] H. Tong, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "What makes the dynamic capacitated arc routing problem hard to solve," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 305–313, ACM, 2022.
[41] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.
[42] 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.
[43] 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.
[44] 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.