Ant Colony Optimization with Heuristic Repair for the Dynamic Vehicle Routing Problem

Iaê S. Bonilha†, Michalis Mavrovouniotis*, Felipe M. Müller‡, Georgios Ellinas*, Marios Polycarpou*
*KIOS Research and Innovation Center of Excellence,
Department of Electrical and Computer Engineering, University of Cyprus, 2109 Nicosia, Cyprus.
email: {mavrovouniotis.michalis,gellinas,mpolycar}@ucy.ac.cy
†PPGEP,
Federal University of Santa Maria, 97105-900, Santa Maria, Brazil.
email: iaesb@hotmail.com
‡Department of Applied Computing,
Federal University of Santa Maria, 97105-900, Santa Maria, Brazil.
email: felipe@inf.ufsm.br

***Abstract*—Ant colony optimization (ACO) algorithms have proved to be suitable for solving dynamic optimization problems. The intrinsic characteristics of ACO algorithms enables them to transfer knowledge from past optimized environments via their pheromone trails to shorten the optimization process in the current environment. In this work, change-related information is also utilized when a dynamic change occurs. The dynamic vehicle routing problem is addressed where nodes are removed, representing customers that have already been visited, or added, representing customers that placed a new order and need to be visited. These change-related information are used to heuristically repair the solution of the previous environment, based on effective moves of the unstringing and stringing operator. Experimental results show that utilizing change-related information is beneficial in the generated dynamic test cases.**

***Index Terms*—Ant colony optimization, heuristic repair, dynamic vehicle routing problem**

I. Introduction

Ant colony optimization (ACO) algorithms have proved to be powerful problem-solving tools. They are able to provide the optimal (or near optimal) solution for difficult vehicle routing problems (VRPs) [1], [2]. Traditionally, researchers have focused their attention on static optimization problems, where the environment of the problem remains fixed during the optimization process of an algorithm. However, many real-world applications are subject to dynamic environments. Dynamic optimization problems (DOPs) are challenging, since the aim of an algorithm is not only to locate the optimum of the problem quickly, but also to efficiently track the moving optimum when changes occur [3]. A dynamic change may involve factors such as the objective function, input variables, problem instance, and constraints.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

A simple way to address DOPs is to restart the optimization process of an algorithm whenever a dynamic change occurs. However, this strategy is usually used in case the dynamic changes are severe. On the contrary, when dynamic changes are small to medium, it is more efficient to adapt to the changing environment by transferring past knowledge since the new environment will be in some sense related to the previous one. ACO is a good choice in adapting to dynamic changes because it naturally implements a memory structure via the pheromone trails, allowing ACO to remember and transfer the past knowledge [4].

Furthermore, previous works on dynamic versions of the well-known traveling salesperson problem (TSP) proved that a dynamic change also contains information that could be useful in the optimization process of the newly generated environment. Guntsch and Middendorf [5] utilized the location of the dynamic changes to locally repair the pheromone trails of ACO. Later on, the same authors utilized change-related information to repair previous infeasible solutions [6].

In this work, change-related information are utilized for the dynamic VRP (DVRP) where nodes are inserted (i.e., representing orders from new customers) and removed (i.e., representing already visited customers). Suppose that new orders have arrived and need to be served causing a dynamic change to the current solution. But the vehicles have already left the depot serving the already scheduled orders. A new feasible solution that includes the new orders and omits the already served orders is required. Therefore, change-related information is used to repair the previous solution, which becomes infeasible by the insertion and/removal of nodes, when a dynamic change occurs. The Unstringing and Stringing (US) [7] moves are used to heuristically repair the solution. In particular, the unstringing moves are used to remove the affected nodes from the solution, whereas the stringing moves are used to insert the new nodes in the solution. Although the US has been designed for TSP solutions, in this work we extend it for VRP solutions which is the main contribution of this paper.

The rest of the paper is organized as follows. Section II describes the VRP and the construction of the dynamic test cases used in the experiments. Section III describes the application of one of the best variations of ACO, i.e., the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [8], to the DVRP. The core logic of repairing the solution heuristically when dynamic changes occur is also described. Section IV gives the experimental results and analysis. Finally, Section V concludes this paper.

II. Dynamic Vehicle Routing Problem

A. Problem Formulation

The VRP is a challenging $\mathcal{NP}$-hard combinatorial optimization problem [9]. The problem can be described as follows: given a fleet of vehicles with limited cargo load capacity, we need to find the best possible route for each vehicle, starting and ending at the central depot, while satisfying the delivery demands of a set of customers.

Typically, a VRP instance is modeled by a fully connected weighted graph $G = (N, A)$, where $N = \{1, \ldots, n\} \cup \{0\}$ is a set of $n+1$ nodes and $A = \{(i, j) \mid i, j \in N, i \neq j\}$ is a set of arcs connecting these nodes. A non-negative value $w_{ij} \in \mathbb{R}^+$ is associated with each arc, representing the euclidean distance between nodes i and j. Node 0 denotes the central depot whereas the remaining nodes denote the customers. Each customer $i \in N$ is assigned a positive value δ_i indicating the customer's delivery demand[1]. Each vehicle[2] has a maximal cargo capacity C.

Let x_{ij} denote the binary decision variables with the following interpretation:

$$x_{ij} = \begin{cases} 1, & \text{if a vehicle visits node } j \text{ immediately after node } i \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the VRP objective is defined as follows:

$$\min \sum_{i,j \in N, i \neq j} w_{ij} x_{ij}, \tag{2}$$

s.t.

$$\sum_{i,j \in N, i \neq j} \delta_i x_{ij} \leq C, \tag{3}$$

$$\sum_{j \in N, i \neq j} x_{ij} - \sum_{j \in N, i \neq j} x_{ji} = 0, \forall i \in N, \tag{4}$$

$$\sum_{j \in N, i \neq j} x_{ij} = 1, \forall i \in N \backslash \{0\}, \tag{5}$$

$$\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \forall S \subseteq \{1, \ldots, n\}, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in N, \ \forall j \in N, i \neq j, \tag{7}$$

where Eq. (2) defines the objective to minimize the total distance traveled, Eq. (3) ensures that the vehicle's cargo capacity constraint is satisfied, Eq. (4) ensures that if a vehicle visits a customer it also leaves the customer, Eq. (5) requires that all customers are visited once, Eq. (6) ensures subtour elimination, and finally, Eq. (7) is the aforementioned binary decision variable.

[1]For the central depot the demand is set to zero, i.e., $\delta_0 = 0$.

[2]A homogeneous fleet of vehicles is considered.

B. Generating Dynamic Test Environments

A DVRP was introduced in [10], [11] in which customers are revealed in different time slices, and a DVRP was considered in [12], [13] in which the weights of arcs connecting the customers increase/decrease. For more details, a comprehensive survey of the DVRP is available in [14].

In this work, we adopt the dynamic framework recently proposed in [4] as follows. Every VRP instance consists of a weight matrix that contains all the weights associated with the arcs of the corresponding graph G. In order to generate dynamic test cases the weight matrix of the problem is subject to changes as follows:

$$\mathbf{W}(T) = \{w_{ij}(T)\}_{(n+1)\cdot(n+1)}, \tag{8}$$

where $\mathbf{W}(\cdot)$ is the weight matrix and T is the environmental period index which is synchronized with the algorithm during the optimization process. Therefore, the environmental period index is defined as $T = \lceil t/f \rceil$, where f is the frequency of change and t is the evaluation counter of the algorithm.

The key idea is to replace nodes from the current working node set $N_{in}(T)$, where $N_{in}(0) = N$, with newly introduced nodes drawn from another node set $N_{out}(T)$. The latter node set $N_{out}(T)$ is initially generated with n new random nodes in the range of the N set. A dynamic change occurs as follows. Every f evaluations exactly $\lceil mn \rceil$ nodes are randomly selected from $N_{out}(T)$ to replace exactly $\lceil mn \rceil$ randomly selected nodes from $N_{in}(T)$, where m ($m \in (0, 1]$) denotes the magnitude of change. The higher the value of m, the more nodes will be replaced. In this way, the weight matrix will be affected because the weights on the arcs connecting the nodes that have been replaced will be modified. Note that the introduced dynamic changes are synchronized with the optimization process of the algorithm. Hence, the parameter f is expressed in algorithmic evaluations.

Such a dynamic change to the node components will also cause a change to the weight matrix defined in Eq. (8) and, thus, it may affect the algorithm's output: the best output before a change may not be the best (or even feasible) after the change. Real-world applications that encompass the aforementioned dynamic change can be found in many fields, including transportation. For example, changes in the visiting locations (e.g., removal of nodes denoting customers already served and the addition of nodes denoting arrival of new customer orders). Suppose that new customer orders have arrived and need to be served causing a dynamic change to the current solution. But the vehicles have already left the depot serving the already scheduled customer orders. A new feasible solution that includes the new orders and omits the already served orders is required.

III. ACO FOR DVRP

The $\mathcal{MM}$AS [8] variant is used which is one of the most-studied ACO variants. In this section we describe the application of $\mathcal{MM}$AS to the DVRP, including the proposed method to heuristically repair solutions when dynamic changes occur.

A. Initialization

A colony of ω ants is initially positioned at the central depot, i.e., component 0. All the solution components of the problem are associated with a pheromone trail value which is uniformly initialized at the start of the execution as follows:

$$\tau_{ij} \leftarrow \tau_0, \forall (i,j) \in A, \tag{9}$$

where τ_{ij} is the pheromone trail value associated with arc (i,j) connecting solution components i and j, and τ_0 is the initial pheromone trail value. A good value for τ_0 was found to be $1/\rho C^{nn}$, where ρ is the evaporation rate (more details are provided later on) and C^{nn} is the solution quality of the solution generated by the nearest-neighbor heuristic [8].

B. Solution Construction

Each ant k represents a complete VRP solution T^k (i.e., the routes of all vehicles) and makes selections biased by the existing pheromone trails and some heuristic information associated with the solution components of the problem, until all the customers are selected.

The probability distribution with which ant k selects the next customer component j from solution component i is defined as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in \mathcal{N}_i^k, \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{10}$$

where τ_{ij} and $\eta_{ij} = 1/w_{ij}(T)$ are, respectively, the existing pheromone trail and the heuristic information available a priori between components i and j. Parameters α and β are the two parameters which determine the relative influence of τ_{ij} and η_{ij}, respectively, while $\mathcal{N}_i^k$ is the set of unselected customers components for k–th ant adjacent to component i. The customer included in $\mathcal{N}_i^k$ must satisfy the vehicle capacity constraint defined in Eq. (4). Whenever the $\mathcal{N}_i^k$ set is empty it denotes that there is no unvisited customer that can be visited (e.g., the capacity constraint is violated). Subsequently, the depot component is added to the VRP solution to close the route of the vehicle (i.e., denotes the return of the vehicle to the central depot). Note that the depot is never included in the $\mathcal{N}_i^k$ set.

C. Pheromone Update

In the $\mathcal{MM}$AS variant [8], [15], the pheromone trails are updated by first decreasing the pheromone trails on all arcs (using pheromone evaporation), and then increasing the pheromone trails on the arcs of the solution constructed by the best ant (using pheromone deposit). The pheromone evaporation is applied as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \forall (i,j) \in A, \tag{11}$$

where $\rho \in (0,1]$ is the evaporation rate.

After pheromone evaporation, the best ant deposits pheromone on the arcs of its solution components as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall (i,j) \in T^{best}, \tag{12}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}(t)$ is the amount of pheromone that the best ant deposits and $C^{best}(t)$ is the quality of the best solution T^{best} at time[3] t. The "best" ant that is allowed to deposit pheromone may be either the best-so-far ant[4], in which case $C^{best}(t) = C^{bs}(t)$, or the iteration-best ant, in which case $C^{best}(t) = C^{ib}(t)$. These two ants are allowed to deposit pheromone in an alternate way. More precisely, the iteration-best ant deposits pheromone at each iteration and the best-so-far ant deposits pheromone occasionally (i.e., every 25 iterations in this work – more details are provided in [15]).

The $\mathcal{MM}$AS variant explicitly imposes the lower and upper limits on the pheromone trail values, preventing in this way the excessive growth of pheromone trails on the arcs of the best ant, as follows:

$$\tau_{ij} \leftarrow \begin{cases} \tau_{max}, & \text{if } \tau_{ij} > \tau_{max}, \\ \tau_{min}, & \text{if } \tau_{ij} < \tau_{min}, \\ \tau_{ij}, & \text{otherwise}, \end{cases} \quad \forall (i,j) \in A, \tag{13}$$

where τ_{min} and τ_{max} are, respectively, the minimum and maximum pheromone trails.

D. Heuristic Repair When a Dynamic Change Occurs

ACO algorithms are able to use knowledge from previous environments via their pheromone trails and can be applied directly to DOPs without any modifications [16], [17]. For example, when the changing environments are similar, the pheromone trails of the previous environment may provide knowledge to speed up the optimization process to the new environment. However, the algorithm must be flexible enough to accept the knowledge transferred from the pheromone trails, or eliminate the pheromone trails, in order to better adapt to the new environment. When a dynamic change occurs, evaporation eliminates the pheromone trails of the previous environment from areas that are generated on the old optimum and helps ants to explore areas for the new optimum. In case the changing environments are completely different, then pheromone reinitialization may be a better choice rather than transferring the knowledge from previous pheromone trails [6], [16], [17].

In this work, we also transfer change-related information [4], (e.g., the nodes to be replaced), to the optimization process of the algorithm. The removal (or unstringing) and

[3]Synchronized with algorithmic evaluations

[4]A special ant that represents the best solution from all iterations so far and, hence, may not necessarily belong to the constructing colony at the current iteration.

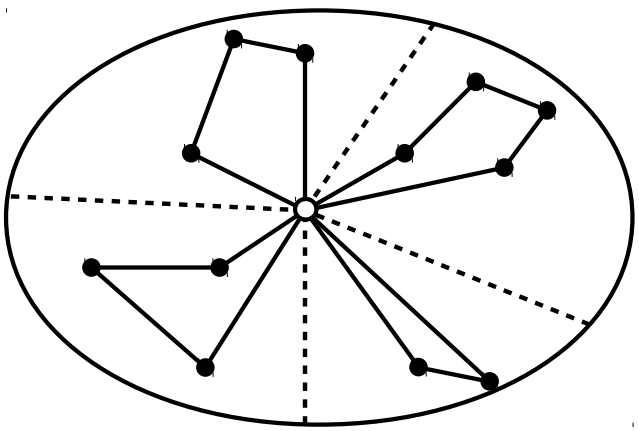


Fig. 1: Segmentation of a complete VRP solution. Black circles denotes customers and the white circle in the middle denotes the depot.

insertion (or stringing) moves of the US heuristic [7] are used to heuristically repair the current best-so-far solution generated by ACO when dynamic changes occur. In particular, the unstringing moves are used to remove the affected nodes from the solution, whereas the stringing moves are used to insert the new nodes in the solution.

The US heuristic was initially designed for the TSP problem, whereas in this work we extend it to the VRP. Since one route of a VRP solution is in fact a TSP solution (e.g., a Hamiltonian path starting and ending at the central depot), a segmentation phase is applied to the VRP solution initially to separate the vehicle routes as presented in Fig. 1. In the example of Fig. 1, four vehicle routes are extracted from the VRP solution.

The new node will be added in the same route where the removed node existed to avoid violating the capacity constraint of the route. The overall procedure of the heuristic repair is presented in Algorithm 1. In the following, a description of the unstringing and stringing moves applied to replace nodes for the DVRP is provided.

1) Removal of Nodes: The main feature of the US heuristic is that the re-insertion of nodes occurs between non-adjacent nodes, resulting in a tour where both nodes become adjacent to the node being inserted [7], [18]. Suppose that we wish to insert V_x between any two nodes V_i and V_j. For a given orientation of a tour, consider node V_k in the subtour from V_j to V_i, and node V_l in the subtour from V_i to V_j. We also consider for any node V_h on the tour, node V_{h+1} (successor) and node V_{h-1} (predecessor). The re-insertion of V_x between V_i and V_j can be done in several ways using different types of insertions and removals.

In [7], [19] two types of removals [i.e., Type I in Fig. 2(a) and Type II in Fig. 2(b)] for the unstringing procedure have been proposed. The unstringing procedure removes a given node from the tour and repairs the connections with the remaining nodes in order to have a closed tour. The procedure

Algorithm 1 HeuristicRepair(T^{bs})

```
INPUT: T^bs   %current best-so-far VRP solution
Apply segmentation in T^bs
for (each subtour S_i in T^bs) do
  if (S_i contains a node to be replaced) then
    while (no further improvement in S_i) do
      for (each node j in S_i) do
        Calculate cost of all Type I and Type II removals
        S'_i ← Apply best removal move to S_i
        Calculate cost of all Type I and Type II insertions
        S'_i ← Apply best insertion move to S_i
        if (S'_i is better than S_i) then
          S_i ← S'_i
        end if
      end for
    end while
  end if
end for
OUTPUT: T^bs   %repaired VRP solution
```

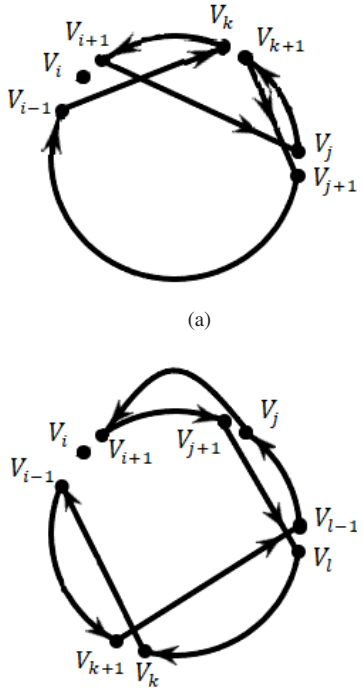
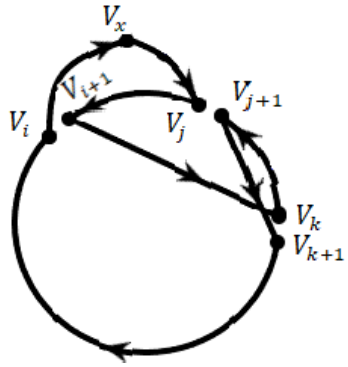


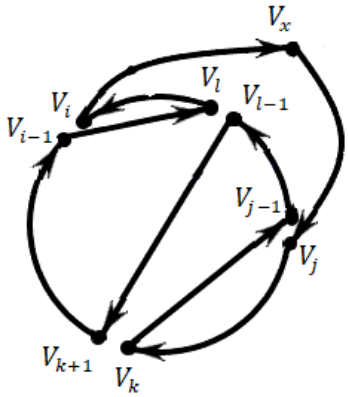
Fig. 2: (a) Type I removal, (b) Type II removal

consists of the following removals:

- Type I removal: Assume that V_j belongs to the neighborhood of V_{i+1} and V_k belongs to the neighborhood of V_{i-1}, with V_k being part of the subtour $(V_{i+1}, \ldots, V_{j-1})$. The removal of node V_i results in the deletion of arcs $(V_{i-1}, V_i), (V_i, V_{i+1}), (V_k, V_{k+1})$ and (V_j, V_{j+1}); and the



(a)



(b)

Fig. 3: (a) Type I insertion, (b) Type II insertion

insertion of arcs (V_{i-1}, V_k), (V_{i+1}, V_j) and (V_{k+1}, V_{j+1}). Also, the subtours $(V_{i+1}, \ldots, V_k)$ and $(V_{k+1}, \ldots, V_j)$ are reversed.

- Type II removal: Assume that V_j belongs to the neighborhood of V_{i+1}, V_k belongs to the neighborhood of V_{i-1}, with V_k being part of the subtour $(V_{j+1}, \ldots, V_{i-2})$ and V_l belongs to the neighborhood of V_{k+1}, with V_l being part of the subtour $(V_j, \ldots, V_{k-1})$. The removal of node V_i results in the deletion of arcs (V_{i-1}, V_i), (V_i, V_{i+1}), (V_{j-1}, V_j), (V_k, V_{k+1}) and (V_l, V_{l+1}); and the insertion of arcs (V_{i-1}, V_k), (V_{l+1}, V_{j-1}), (V_{i+1}, V_j) and (V_l, V_{k+1}). As above, the subtours $(V_{i+1}, \ldots, V_{j-1})$ and $(V_{l+1}, \ldots, V_k)$ are reversed.

2) Insertion of Nodes: In [7], [19] two types of insertions [i.e., Type I in Fig. 3(a) and Type II in Fig. 3(b)] for the stringing procedure have been proposed. The stringing procedure is basically the reverse of the unstringing procedure and consists of the following insertions:

- Type I insertion: Assume that $V_k \neq V_i$ and $V_k \neq V_j$. The insertion of V_x results in the deletion of arcs (V_i, V_{i+1}), (V_j, V_{j-1}) and (V_k, V_{k+1}), and the insertion of arcs (V_i, V_x), (V_x, V_j), (V_{i+1}, V_k) and (V_{j+1}, V_{k+1}). Also, the subtours $(V_{i+1}, \ldots, V_j)$ and $(V_{j+1}, \ldots, V_k)$ are reversed.
- Type II insertion: Assume that $V_k \neq V_j$, $V_k \neq V_{j+1}$, $V_l \neq V_i$, and $V_l \neq V_{i+1}$. The insertion of V_x results in the deletion of arcs (V_i, V_{i+1}), (V_{l-1}, V_l), (V_j, V_{j+1}) and (V_{k-1}, V_k), and the insertion of arcs (V_i, V_x), (V_x, V_j), (V_l, V_{j+1}), (V_{k-1}, V_{l-1}) and (V_{i+1}, V_k). As above, the subtours $(V_{i+1}, \ldots, V_{l-1})$ and $(V_l, \ldots, V_j)$ are reversed.

IV. Experimental Studies

A. Experimental Setup

In the experiments, we investigate the effect of repairing the best solution heuristically, utilizing change-related information when dynamic changes occur. The $\mathcal{MM}$AS with the proposed heuristic repair (denoted $\mathcal{MM}$AS+H) is compared against the $\mathcal{MM}$AS with complete re-initialization of pheromone trails when dynamic changes occurs (denoted $\mathcal{MM}$AS+R), and the $\mathcal{MM}$AS with simple repair in which the inserted nodes are placed in the position of the removed nodes (denoted $\mathcal{MM}$AS+S).

All $\mathcal{MM}$AS algorithmic parameters were set to commonly used values: $\alpha = 1$, $\beta = 5$, $\rho = 0.8$ and the number of ants was set to $\omega = 50$. Dynamic test cases are generated from six static benchmark instances [22] obtained from CVRPLIB[5]: X-n101-k25, X-n143-k7, X-n219-k73, X-n313-k71, X-n429-k61 and X-n561-k42 using the dynamic generator described in Section II. The frequency of change f was set to $10e4$ algorithmic evaluations and the magnitude of change m was set to 0.1, 0.25, 0.5, and 0.75, indicating small to medium to severe dynamic changes. Totally, a series of 4 DVRP test cases were constructed from each stationary instance to systematically analyze the performance of the algorithms. For each algorithm on a DVRP, 30 independent runs were executed on the same set of random seed numbers. For each run, 50 environmental changes were allowed and an observation (i.e., the value of the best-so-far ant after a dynamic change) was recorded. For a fair comparison, all the algorithms performed the same number of evaluations. The proportional evaluations required when applying the moves for the heuristic repair used in $\mathcal{MM}$AS+H are added to the total evaluations of the algorithm between the dynamic changes.

The *offline performance* [23] was used to evaluate the overall performance of the algorithms, which is defined as:

$$\bar{P}_{offline} = \frac{1}{E} \sum_{t=1}^{E} C^{bs}(t), \tag{14}$$

where E is the total number of evaluations and $C^{bs}(t)$ is the best-so-far solution quality after a change.

B. Experimental Results and Discussion

The experimental results regarding the offline performance of the investigated algorithms for all DVRPs are presented in Table I. The corresponding statistical results are presented in Table II, in which pairwise *Mann–Whitney* statistical tests with a significance level of 0.05 were performed. In Table II, the results are shown as "+", "−" and "∼" when the first algorithm is significantly better than the second one,

[5]Available from http://vrp.galgos.inf.puc-rio.br/index.php/en/

TABLE I: Experimental results regarding the average offline performance of ACO algorithms on DVRP test cases.

Problem Instance	m	$\mathcal{MM}$AS+H	$\mathcal{MM}$AS+S	$\mathcal{MM}$AS+R
X-n101-k25	0.1	29089	29098	29125
	0.25	29386	29401	29417
	0.5	29473	29504	29517
	0.75	29299	29337	29352
X-n143-k7	0.1	17772	17830	17851
	0.25	17726	17816	17795
	0.5	17765	17849	17815
	0.75	17840	17927	17893
X-n219-k73	0.1	121163	121166	121144
	0.25	121305	121311	121274
	0.5	121750	121764	121723
	0.75	121341	121349	121319
X-n313-k71	0.1	99824	99852	99903
	0.25	99286	99352	99384
	0.5	98342	98439	98428
	0.75	97925	98048	98010
X-n429-k61	0.1	68803	68861	68832
	0.25	68864	69044	68958
	0.5	69218	69484	69327
	0.75	69214	69461	69332
X-n561-k42	0.1	46638	46888	46842
	0.25	47021	47424	47241
	0.5	47143	47583	47371
	0.75	47207	47633	47447

TABLE II: Statistical results regarding the average offline performance of ACO algorithms on DVRP test cases.

Problem Instance	m	$\mathcal{MM}$AS+H **vs.** $\mathcal{MM}$AS+S	$\mathcal{MM}$AS+H **vs.** $\mathcal{MM}$AS+R	$\mathcal{MM}$AS+S **vs.** $\mathcal{MM}$AS+R
X-n101-k25	0.1	$\sim$	$+$	$\sim$
	0.25	$+$	$+$	$\sim$
	0.5	$+$	$+$	$\sim$
	0.75	$+$	$+$	$\sim$
X-n143-k7	0.1	$+$	$+$	$\sim$
	0.25	$+$	$+$	$+$
	0.5	$+$	$+$	$+$
	0.75	$+$	$+$	$+$
X-n219-k73	0.1	$\sim$	$-$	$-$
	0.25	$\sim$	$-$	$-$
	0.5	$\sim$	$-$	$-$
	0.75	$\sim$	$-$	$-$
X-n313-k71	0.1	$+$	$+$	$+$
	0.25	$+$	$+$	$+$
	0.5	$+$	$+$	$+$
	0.75	$+$	$+$	$+$
X-n429-k61	0.1	$+$	$+$	$-$
	0.25	$+$	$+$	$-$
	0.5	$+$	$+$	$-$
	0.75	$+$	$+$	$-$
X-n561-k42	0.1	$+$	$+$	$\sim$
	0.25	$+$	$+$	$-$
	0.5	$+$	$+$	$-$
	0.75	$+$	$+$	$-$

when the second algorithm is significantly better than the first one, and when the two algorithms are not significantly

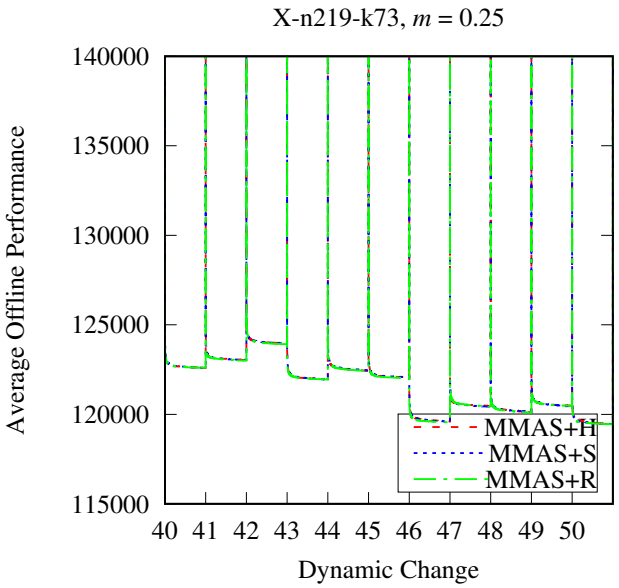


Fig. 4: Dynamic average performance of ACO algorithms on the X-n219-k73 problem instance for the last ten dynamic changes.

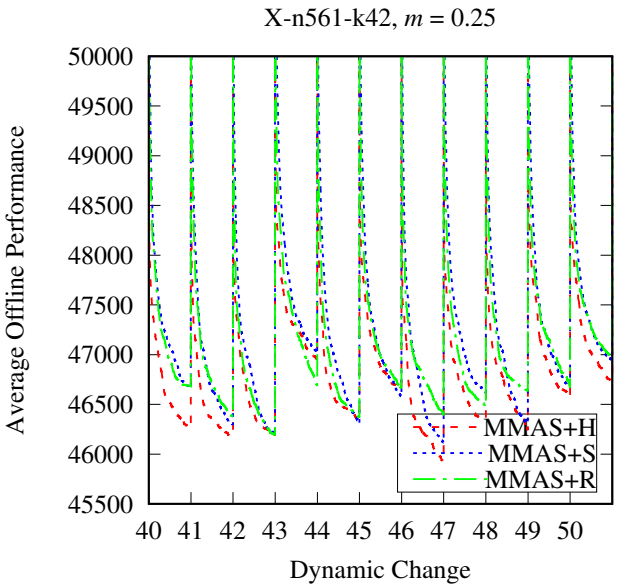


Fig. 5: Dynamic average performance of ACO algorithms on the X-n561-k42 problem instance for the last ten dynamic changes.

different, respectively. In Figs. 4 and 5 the dynamic average offline performance against the algorithmic iterations of $\mathcal{MM}$AS+H, $\mathcal{MM}$AS+S, and $\mathcal{MM}$AS+R are plotted for the last ten dynamic changes to better understand the behavior of the algorithms on X-n219-k73 and X-n516-k42 problem instances, respectively. From the experimental results the following observations can be drawn.

First, $\mathcal{MM}$AS+H significantly outperforms $\mathcal{MM}$AS+S in most DVRP cases (except for the X-n219-k73 problem instance). These results were expected because $\mathcal{MM}$AS+H replaces a node in vehicle routes and at the same time optimizes the route using the heuristic moves. In contrast, $\mathcal{MM}$AS+S replaces the nodes in the positions of the removed nodes. This observation confirms that proper utilization of the change-related information further improves the performance of the algorithm. For example, in Fig. 5 it can be observed that $\mathcal{MM}$AS+H obtains better offline performance than the competing algorithms in most dynamic changes.

Second, $\mathcal{MM}$AS+H significantly outperforms $\mathcal{MM}$AS+R in most DVRP cases (except for the X-n219-k73 problem instance in which $\mathcal{MM}$AS+R significantly outperforms $\mathcal{MM}$AS+H). These results confirm that transferring knowledge from previous environments shortens the re-optimization process compared to the case where the re-optimization process restarts from scratch (it can also be observed in Fig. 5). However, the comparisons between $\mathcal{MM}$AS+S and $\mathcal{MM}$AS+R show that improper utilization of change-related information may have a negative impact on the performance of the algorithm.

Finally, a possible reason concerning the inferior performance of $\mathcal{MM}$AS+H on the X-n219-k73 problem instance is that a solution for this instance consists of several vehicles routes (i.e., 73). Considering the size of the problem (i.e., 219) it is possible that vehicle routes of small size are formed that cannot be optimized by the heuristic moves. Therefore, the replacement procedure will be similar with the one used in $\mathcal{MM}$AS+S. In fact, from Fig. 4 it can be observed that all three algorithms have similar dynamic performance on the X-n219-k73 problem instance.

V. Conclusions

In this work, we utilize change-related information to improve the solution quality of ACO when a dynamic change occurs. The DVRP is used in which nodes are inserted and removed. The unstringing and stringing moves are used to heuristically repair the best solution from the previous environment. The performance of ACO with the proposed heuristic repair is investigated on dynamic test cases of the DVRP that are systematically constructed. The experimental results confirm the positive effect on the performance of ACO when utilizing change-related information.

For future work, it would be interesting to investigate more effective ways of utilizing change-related information, e.g., with additional removal and insertion moves.

References

[1] M. Mavrovouniotis, C. Li, G. Ellinas, and M. Polycarpou, "Parallel ant colony optimization for the electric vehicle routing problem," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 1660–1667.

[2] A. E. Rizzoli, R. Montemanni, E. Lucibello, and L. M. Gambardella, "Ant colony optimization for real-world vehicle routing problems," *Swarm Intelligence*, vol. 1, no. 2, pp. 135–151, Dec 2007.

[3] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments–a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, June 2005.

[4] 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.

[5] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic TSP," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, E. J. W. Boers, Ed., vol. 2037. Springer Berlin Heidelberg, 2001, pp. 213–222.

[6] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," in *Ant Algorithms*, ser. Lecture Notes in Computer Science, M. Dorigo, G. Di Caro, and M. Sampels, Eds., vol. 2463. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 111–122.

[7] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

[8] T. Stützle and H. Hoos, "$\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for the traveling salesman problem," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, April 1997, pp. 309–314.

[9] M. Garey and D. Johnson, *Computer and intractability: A guide to the theory of $\mathcal{NP}$-completeness*. San Francisco: Freeman, 1979.

[10] P. Kilby, P. Prosser, and P. Shaw, "Dynamic VRPs: A study of scenarios," University of Strathclyde, U.K, Tech. Rep. APES-06-1998, 1998.

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

[12] M. Mavrovouniotis and S. Yang, "Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, June 2012, pp. 2645–2652.

[13] M. Mavrovouniotis and S. Yang, "Ant algorithms with immigrants schemes for the dynamic vehicle routing problem," *Information Sciences*, vol. 294, pp. 456–477, 2015.

[14] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Netw.*, vol. 67, no. 1, pp. 3–31, Jan. 2016.

[15] T. Stützle and H. H. Hoos, "$\mathcal{MAX}$–$\mathcal{MIN}$ ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, June 2000.

[16] D. Angus and T. Hendtlass, "Ant colony optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Hendtlass and M. Ali, Eds., vol. 2358. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 618–627.

[17] M. Mavrovouniotis and S. Yang, "Adapting the pheromone evaporation rate in dynamic routing problems," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, A. Esparcia-Alcázar, Ed., vol. 7835. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 606–615.

[18] G. Pesant, M. Gendreaul, and J.-M. Rousseau, "Genius-CP: A generic single-vehicle routing algorithm," in *Principles and Practice of Constraint Programming-CP97*, G. Smolka, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 420–434.

[19] M. Mavrovouniotis, F. M. Müller, and S. Yang, "An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem," in *Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO15)*, 2015, pp. 49–56.

[20] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller, "A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times," *International Journal of Production Economics*, vol. 43, no. 2-3, pp. 79–89, 1996.

[21] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for the dynamic travelling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, July 2017.

[22] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, 2017.

[23] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, ser. Natural Computing Series, A. Ghosh and S. Tsutsui, Eds. Springer Berlin Heidelberg, 2003, pp. 239–262.