# Ant algorithms with immigrants schemes for the dynamic vehicle routing problem

Michalis Mavrovouniotis*, Shengxiang Yang

*Centre for Computational Intelligence (CCI),*
*School of Computer Science and Informatics, De Montfort University,*
*The Gateway, Leicester LE1 9BH, United Kingdom*

**Abstract**

Many real-world optimization problems are subject to dynamic environments that require an optimization algorithm to track the optimum during changes. Ant colony optimization (ACO) algorithms have proved to be powerful methods to address combinatorial dynamic optimization problems (DOPs), once they are enhanced properly. The integration of ACO algorithms with immigrants schemes showed promising performance on different DOPs. The principle of immigrants schemes is to introduce new solutions (called immigrants) and replace a small portion in the current population. In this paper, immigrants schemes are specifically designed for the dynamic vehicle routing problem (DVRP). Three immigrants schemes are investigated: random, elitism- and memory-based. Their difference relies on the way immigrants are generated, e.g., in random immigrants they are generated randomly whereas in elitism- and memory-based the best solution from previous environments is retrieved as the base to generate immigrants. Random immigrants aim to maintain the population diversity in order to avoid premature convergence. Elitism- and memory-based immigrants aim to maintain the population diversity and transfer knowledge from previous environments, simultaneously, to enhance the adaptation capabilities. The experiments are based on a series of systematically constructed DVRP test cases, generated from a general dynamic benchmark generator, to compare and benchmark the proposed ACO algorithms integrated with immigrants schemes with other peer ACO algorithms. Sensitivity analysis regarding some key parameters of the proposed algorithms is also carried out. The experimental results show that the performance of ACO algorithms depends on the properties of DVRPs and that immigrants schemes improve the performance of ACO in tackling DVRPs.

*Keywords:* Ant colony optimization, Dynamic optimization problem, Dynamic vehicle routing problem, Immigrants schemes,

## 1. Introduction

Ant colony optimization (ACO) algorithms have been successfully applied for solving different combinatorial optimization problems, e.g., vehicle routing problems (VRPs) [12, 15]. Traditionally, researchers have drawn their attention on stationary optimization problems, where the environment remains fixed during the execution of an algorithm [23, 48]. 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 find the optimum of the problem quickly, but to efficiently track the moving optimum when changes occur [29]. A dynamic change in a DOP may involve factors like the objective function, input variables, problem instance, constraints, and so on.

Conventional ACO algorithms have been designed for stationary optimization problems [14], e.g., to converge fast into the global (or near) optimum solution, and may face a serious challenge to tackle DOPs. This is because the pheromone trails of the previous environment may bias the population to search into the old optimum, making it difficult to track the moving optimum. As a result, ACO will not adapt well once the population converges into an optimum. Considering that DOPs can be taken as a series of stationary problem instances, a simple way to tackle them is to re-initialize the pheromone trails and consider every dynamic change as the arrival of a new problem instance which needs to be solved from scratch [37]. However, this restart strategy is generally not efficient.

In contrast, once ACO algorithms are enhanced properly, they are able to adapt to dynamic changes since they are inspired from nature, which is a continuously changing process [2, 3, 29]. Recently, ACO algorithms have been successfully applied in combinatorial optimization problems with dynamic environments since they are able to reuse knowledge from previously generated pheromone trails [25, 26, 38]. More precisely, 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 needs to be flexible enough to accept the knowledge transferred from the pheromone trails, or eliminate older unused pheromone trails, to better adapt to the new environment.

Several strategies have been proposed and integrated with ACO to shorten the re-optimization time and maintain a high quality of the output efficiently, simultaneously. These strategies can be categorized as: increasing diversity after a dy-

*Corresponding author.
*Email addresses:* mmavrovouniotis@dmu.ac.uk (Michalis Mavrovouniotis), syang@dmu.ac.uk (Shengxiang Yang)

namic change [25, 37]; maintaining diversity during the execution [16, 38]; memory-based schemes [26, 27, 36]; and hybrid/memetic algorithms [39].

Among these strategies, immigrants schemes have shown promising results on binary DOPs [55, 61], dynamic travelling salesman problems [38], and recently dynamic vehicle routing problems (DVRPs) [35, 36]. Within immigrants schemes, a small portion of newly generated ants, called immigrant ants, replace the worst ants in the current population. Each immigrants scheme differs in the way immigrant ants are generated, e.g., random immigrants represent random solutions of the problem [24], elitism- or memory-based immigrants represent solutions that differ slightly from the best solution of a previous environment [54, 55]. In this paper, we focus on the immigrants schemes for the DVRP, and thus, the immigrant ants represent a feasible VRP solution.

Random immigrants ACO (RIACO) and elitism-based immigrants ACO (EIACO) [35] were previously applied only on a DVRP where the pattern of dynamic changes is random, denoted as *random DVRPs*, whereas memory-based immigrants ACO (MIACO) [36] was applied only on a DVRP where the pattern of dynamic changes is cyclic, denoted as *cyclic DVRPs*. However, these algorithms were extended from the previous developments proposed for dynamic travelling salesman problems [38], and thus, their behaviour was unexpected in most dynamic test cases. In this paper, RIACO, EIACO, and MIACO are redesigned specifically for the DVRP and their performance is investigated on both random and cyclic DVRPs generated by a different benchmark generator. The proposed algorithms differ from their previous versions [35, 36] as follows: 1) the way random, elitism- and memory-based immigrant ants are generated; 2) the selection of ant as the base to generate elitism-based immigrants; and 3) the ants selected to replace other ants in the memory in order to generate memory-based immigrants.

The main issue with different dynamic benchmark generators for the DVRP currently used in the literature [30, 35, 36, 41] is that the optimum value is not known during the dynamic changes. The same case stands for the DVRPs considered on the initial developments of RIACO, EIACO and MIACO [35, 36]. Therefore, it is impossible to observe how close to the optimum each algorithm converges after a change. In binary and continuous optimization functions, algorithms are benchmarked in dynamic generators where the optimum value is known during the dynamic changes [5, 32, 53, 56]. Comprehensive surveys regarding benchmark generators for DOPs are available in [9, 42]. In this paper, the dynamic benchmark generator for permutation problems (DBGP) [40] is mainly used which can generate DVRPs with known optimum over the environmental changes and hence facilitate the observation on how close to the optimum an algorithm performs. Based on the DVRPs generated from DBGP, this paper benchmarks and compares the performance of the re-designed algorithms with other peer ACO algorithms. In addition, the algorithms are compared on the DVRP with traffic factors [38], which models a real-world scenario.

To summarize, the contributions of the paper are as follows: 1) RIACO, EIACO and MIACO, which were developed previously, are re-designed specifically to address the DVRP; 2) the dynamic test cases are generated from the recently proposed DBGP [40]; and 3) the experimental studies are extended on both random and cyclic DVRPs for all ACO algorithms. Sensitivity analysis on key parameters of the algorithms is also carried out.

The rest of the paper is organized as follows. Section 2 describes the basic VRP and its stationary and dynamic extensions. Section 3 briefly reviews existing work on ACO for DVRPs. Section 4 describes the benchmark generator used which can generate DVRPs where the optimum value is known over dynamic changes. Section 5 describes the algorithms proposed in this paper for addressing the DVRP. Section 6 gives the experimental results, including the statistical tests, and analysis. Finally, Section 7 concludes this paper with discussions on relevant future work.

## 2. Vehicle routing problems

### *2.1. Basic VRP description*

The basic VRP can be described as follows: we need to route a number of vehicles with a fixed capacity to satisfy the demands of all the customers, starting from and finishing at the depot [10]. Hence, a VRP without the capacity constraint and with one vehicle can be seen as a travelling salesman problem. The basic problem is also known as the capacitated VRP and belongs to the class of $\mathcal{NP}$-hard combinatorial problems [31]. The important symbols used in this section and for the remaining paper are summarized in Table 1.

Usually, the VRP is represented by a complete weighted graph $G = (N, A)$ with $n + 1$ nodes, where $N = \{v_0, \ldots, v_n\}$ is a set of vertices corresponding to the customers (or delivery points) $v_i$ $(i = 1, \ldots, n)$ and the central depot $v_0$, and $A = \{(v_i, v_j) : i \neq j\}$ is a set of arcs. Each arc $(i, j)$ is associated with a non-negative value $d_{ij}$, which represents the distance (or travel time) between customers $i$ and $j$. For each customer $i$, a non-negative demand $q_i$ is given, whereas for the central depot, a zero demand is associated. The number of vehicles is denoted by $u$ and each vehicle $k$ has a limited capacity $Q^k$. If the fleet of vehicles is homogeneous, $Q^k$ is the same for all vehicles; whereas if the fleet of vehicles is heterogeneous, $Q^k$ is different for each vehicle. In this paper, a homogeneous fleet of vehicles is considered.

Formally, the VRP can be described as follows. Let $\psi_{ij}^k$ denote the binary decision variables defined as follows:

$$\psi_{ij}^k = \begin{cases} 1, & \text{if arc } (i, j) \text{ is covered by vehicle } k, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

Then, the objective of the VRP is defined as follows:

$$f(s) = \min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} \sum_{k=1}^{u} \psi_{ij}^k, \tag{2}$$

subject to:

$$\sum_{i=0}^{n} q_i \sum_{j=0}^{n} \psi_{ij}^k \leq Q^k, \forall k \in \{1, \ldots, u\}, \tag{3}$$

Table 1: Mathematical symbols used in this paper

| Symbol | Description |
|---|---|
| $G = (N, A)$ | Fully connected weighted graph |
| $N$ | Set of nodes |
| $A$ | Set of arcs |
| $n$ | Number of customers (nodes) |
| $(i, j)$ | Connection between customers $i$ and $j$ |
| $d_{ij}$ | Distance (or travel time) between customers $i$ and $j$ |
| $q_i$ | Demand of customer $i$ |
| $Q^k$ | Capacity of vehicle $k$ |
| $u$ | Number of vehicles |
| $\psi_{ij}^k$ | Binary decision variable |
| $\mathbf{D}$ | Distance matrix |
| $\vec{V}$ | Random vector of customers |
| $\vec{U}$ | Re-ordered vector of $\vec{V}$ |
| $f$ | Frequency of dynamic change |
| $m$ | Magnitude of dynamic change |
| $t$ | Current iteration count |
| $P(t)$ | Population of ants for iteration $t$ |
| $\tau_0$ | Initial pheromone trail value |
| $\tau_{max}$ | Maximum pheromone trail value |
| $\tau_{ij}$ | Existing pheromone trails between customer $i$ and $j$ |
| $\eta_{ij}$ | Heuristic information between customer $i$ and $j$ |
| $T^k$ | The tour of ant $k$ |
| $k_{long}(t)$ | Long-term memory for iteration $t$ |
| $k_{short}(t)$ | Short-term memory for iteration $t$ |
| $r_i$ | Immigrant ant replacement rate |
| $p_m^i$ | Immigrants' mutation probability |
| $t_M$ | Iteration number where the next memory update will occur |
| $s^{bs}$ | Global best solution |
| $s^{ib}$ | Iteration best solution |
| $s^{elite}$ | Best solution for a given environment |
| $s^{cm}$ | Most similar ant in long-term memory |
| $s^{mb}$ | Best ant in long-term memory |

$$\sum_{i=0}^{n} \psi_{il}^k - \sum_{j=0}^{n} \psi_{lj}^k = 0, \forall k \in \{1, \ldots, u\}, l \in \{0, \ldots, n\}, \quad (4)$$

$$\sum_{i=1}^{n} \psi_{i0}^k \leq 1, \forall k \in \{1, \ldots, u\}, \quad (5)$$

$$\sum_{j=1}^{n} \psi_{0j}^k \leq 1, \forall k \in \{1, \ldots, u\}, \quad (6)$$

where Eq. (3) is the capacity constraint that ensures that no vehicle can be overloaded; Eq. (4) ensures that if a vehicle arrives at a customer, it must also leave the customer; Eq. (5) and (6) ensure that each vehicle is not scheduled more than once.

The VRP is closely related with many real-world applications since it concerns the transportation of goods between a depot and customers by a fleet of vehicles [20, 48]. Examples of real-world applications include: mail delivery, school bus routing, solid waste collection, heating oil distribution, parcel pick-up and delivery, and many others.

### 2.2. Stationary extensions of the VRP

Apart from the capacity constraint, other possible constraints can be imposed to the VRP taken from real-world applications. For example, in the VRP with service time constraint, each vehicle, apart from satisfying the capacity constraint described in Eq. (3), needs to satisfy a service time constraint. Another popular extension is the VRP with time windows, where each customer $i$ is associated with a time interval $[e_i, l_i]$, called the time window, together with its demand $q_i$, where $e_i$ and $l_i$ is the earliest and latest possible service time for customer $i$, respectively.

Other VRP extensions include: the VRP with backhauls, where customers are classified into two groups, i.e., linehaul customers whose demand needs to be delivered and backhaul customers whose demand needs to be picked up; the multi-depot VRP, where vehicles can be served by several depots instead of a single one; the VRP with pick-up and delivery, where a vehicle fleet must satisfy a set of requests in which the delivery goods are not originally concentrated in the depot(s), but they are distributed over the deliver points; and combinations of the aforementioned extensions, e.g., the VRP with pick-up and delivery with time-windows.

### 2.3. Dynamic extensions of the VRP

Although several additional constraints, described above, make the VRP more challenging and realistic, most real-world scenarios are subject to dynamic environments. A comprehensive survey on DVRPs is available in [44].

The most common and well studied source of dynamics in the DVRP is the arrival of customer requests during the operation, i.e., the DVRP with stochastic demands [30, 41]. More precisely, the customer requests are not completely known a prior, but they arrive dynamically during the distribution process, and thus, the routes have to be re-planned immediately to include the new requests. Similarly, new requests may arrive in the VRP, where a new shipment of goods needs to be delivered from an existing pick-up point to an existing drop-off point.

Recently, the travel time is taken into account, which is a dynamic component of most real-world applications, i.e., the DVRP with traffic factors [36]. More precisely, the cost of links, i.e., $d_{ij}$, depends on the period of time. For example, in the rush hour periods the traffic is higher and the routes have to be replaned immediately to avoid traffic jams.

## 3. ACO applications for the vehicle routing problem

ACO is a metaheuristic developed especially for combinatorial optimization problems. ACO algorithms are inspired from the foraging behaviour of real ant colonies where ants communicate via their pheromone trails to optimize their paths. The research on ACO has mainly focused on applications in stationary environments, such as the travelling salesman problem [14], VRP [20], quadratic assignment problem [22], capacitated arc routing problem [52], and many others [15].

Regarding the VRP application, there are two main ACO developments in the literature: the rank-based ant system for the capacitated VRP ($AS_{rank}$-CVRP) [6, 7, 47] and the multi-colony ant colony system for the VRP with time windows (MACS-VRPTW) [21]. The former development is applied to the basic capacitated VRP. In the case of $AS_{rank}$-CVRP, only the $w$ best ants are allowed to deposit pheromone. The quantity of pheromone is based on the ranked position of the ant where the

best ant always deposits more pheromone [8]. The latter development is applied to the well-studied VRP with time windows. In the case of MACS-VRPTW, the ACO algorithm consists of two colonies that aim to optimize two objectives. Both colonies are based on the ant colony system [13] variation, where only the best ant deposits pheromone. One colony is responsible to minimize the travel time (or distance) whereas the other colony is responsible to minimize the number of vehicles. The two colonies interact as follows: if a solution with less vehicles is found by one colony, then the number of vehicles is passed to the other colony to find the optimal routes. ACO has also been applied to several other VRP variations, including the VRP with pick up and delivery [11, 19] and the VRP with backhauls and time windows [18, 45, 46].

$AS_{rank}$-CVRP can be applied to the DVRP directly without any modifications due to the pheromone evaporation mechanism. The pheromone evaporation in ACO algorithms can eliminate the areas with high intensity of pheromones that are generated by ants due to the stagnation behaviour (i.e., when all ants follow the same path and construct the same solution). The adaptation via pheromone evaporation may be a sufficient choice when the changing environments are similar; otherwise, a complete re-initialization of the pheromone trails (i.e., a complete restart of the ACO algorithm) after a dynamic change may be a better choice. In fact, $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [50] has been applied to dynamic routing problems, including DVRP, with a global re-initialization of the pheromone trails, denoted as $\mathcal{MM}AS_R$ in this paper [37].

MACS-VRPTW has also been applied to the DVRP with stochastic demands, known as the ACS-DVRP [41]. However, instead of using two interactive ant colonies, ACS-DVRP uses a single colony to minimize the distance. When a dynamic change occurs, a pheromone conservation scheme is performed to regulate previously generated pheromone trails.

Other ACO algorithms that have been developed for the dynamic travelling salesman problem, such as the memetic ACO (M-ACO), have also been applied to the DVRP with traffic factors [39]. Within M-ACO, local search operators based on the swap operator are integrated with the ACO algorithm. The framework of M-ACO is based on the well-known population-based ACO [26] in which a memory of the best ants is used to update the pheromone trails. Additional random immigrants are introduced to the memory when all stored ants are identical in order to maintain diversity.

Recently, RIACO, EIACO and MIACO algorithms have been extended from their dynamic travelling salesman problem with traffic factors developments [38] to solve the DVRP with traffic factors. Within RIACO [35], a random immigrant is generated as follows. A randomly selected unvisited customer is selected until a feasible route is completed (i.e., just before the capacity constraint is violated), and then a new route is started. This process is repeated until all customer demands are satisfied. Within EIACO [35], the best ant of the previous environment is used as the base to generate elitism-based immigrants as follows. The depot objects are removed from the best ant's solution and the inver-over operator [51] is applied in the way similar to the elitism-based immigrants scheme used in EIACO on the dynamic travelling salesman problem [38]. Then, the depot objects are inserted back to the solution such that the capacity constraint is satisfied. Similarly, memory-based immigrants are generated for MIACO using a best ant extracted from the memory [36].

In this paper, we further develop the RIACO, EIACO and MIACO algorithms to address DVRPs, where the generation of immigrants is designed especially for the DVRP, rather than transferring the ideas from the dynamic travelling salesman problem to the DVRP directly. More details will be described in Section 5, after the description of a recently proposed benchmark generator [40] used to generate DVRPs with different properties in the next section.

## 4. Dynamic vehicle routing problem benchmark

### *4.1. Generate dynamic test cases*

In order to generate dynamic routing problems, we have recently proposed the DBGP [40], which can convert any static permutation-encoded benchmark problem instance to a dynamic environment. In cases where the optimum of the benchmark problem instance is known, it will remain known during the environmental changes. DBGP shifts the population of the algorithm to search to a new location in the fitness landscape. Other existing DVRP benchmark generators, e.g., the DVRP with stochastic demands and the DVRP with traffic factors, modify the fitness landscape and the optimum value is changed in every dynamic change.

For the dynamic cases generated by the DBGP, one can observe "how close to the moving optimum an algorithm performs when a change occurs", whereas for the dynamic cases generated by other existing benchmark generators, it is impossible to do that. However, since a DOP can be considered as a series of static problem instances, a direct way is to solve each one to optimality, which may be non-trivial due to the $\mathcal{NP}$-hardness of most combinatorial optimization problems, especially for large-size problem instances. It may be possible for small-size problem instances, but then the usefulness of benchmarking will be reduced.

Considering the VRP described in Section 2, each node (or object) $i \in N$ has a location defined by $(x, y)$ and each link $(i, j) \in A$ is associated with a non-negative distance $d_{ij}$. Usually, the distance matrix of a problem instance is defined as $\mathbf{D} = (d_{ij})_{n\times n}$. DBGP generates the dynamic case as follows. Every $f$ iterations a random vector $\vec{V}(T)$ is generated that contains exactly $m \times n$ objects of the VRP problem instance, where $T = \lceil t/f \rceil$ is the index of the period of change, $t$ is the iteration count of the algorithm, $f$ determines the frequency of change, $n$ is the size of the problem instance and $m$ determines the magnitude of change. More precisely, $m \in [0.0, 1.0]$ defines the degree of change, in which only the first $m \times n$ of $\vec{V}(T)$ object locations are swapped. Then a randomly re-ordered vector $\vec{U}(T)$ is generated that contains only the objects of $\vec{V}(T)$. Therefore, exactly $m \times n$ pairwise swaps are performed in $\mathbf{D}$ using the two random vectors $(\vec{V}(T) \otimes \vec{U}(T))$, where "$\otimes$" denotes the swap operator. The pseudocode of DBGP is given in Algorithm 1.

**Algorithm 1** DBGP

```
1: Read initial benchmark problem of size n
2: ComputeDistances(D)
3: i ← 0
4: repeat
5:    DoOptimization(i) //e.g., with ACO
6:    i ← i + 1
7:    if (f%i = 0) then
8:       num_swaps ← m × n
9:       for (j = 1 to num_swaps) do
10:         V[j] ⊗ U[j]
11:      end for
12:      ComputeDistances(D)
13:   end if
14: until (optimization not terminated)
```

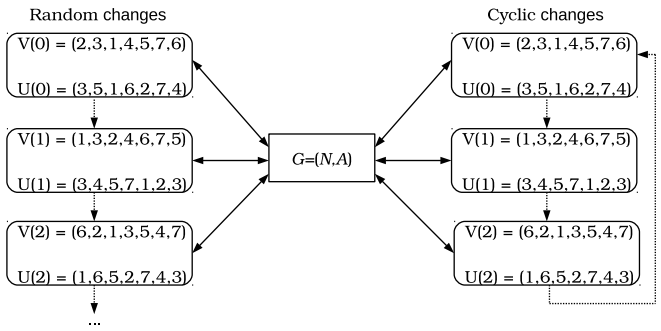


Figure 1: Illustration of the generation of random and cyclic DVRPs using the DBGP.

### *4.2. Random vs cyclic dynamic environments*

The above way for generating dynamic cases for DVRPs occurs in a random pattern. Random DVRPs can be used for testing all ACO algorithms. Another variation of a dynamic environment is where previous environments will appear again in the future. Such dynamic environments are quite common in real-world scenarios and are essential when algorithms that are enhanced with memory structures are investigated because their performance heavily depends on such dynamic cases [5, 56].

Cyclic DVRPs can be further generated by the DBGP as follows. First, $K$ random vectors $(\vec{V}(0), \ldots, \vec{V}(K-1))$ are generated with their corresponding re-ordered vectors $(\vec{U}(0), \ldots, \vec{U}(K-1))$, subject to the magnitude value $m$, as the base states of the search space. Second, the initial base state $(\vec{V}(0) \otimes \vec{U}(0))$ is applied in **D**. Then, the environment moves among these base states in a fixed logical ring. Hence, every $f$ iterations the dynamic changes of the previous base state $(\vec{V}(T-1) \otimes \vec{U}(T-1))$ are reversed, which will cause **D** to return to its original state, and then the new dynamic changes are applied from the next base state $(\vec{V}(T) \otimes \vec{U}(T))$. In this way, it is guaranteed that the environments generated from the base states will re-appear. For example, after $K$ dynamic changes the previous environments will begin to re-appear. Fig. 1 illustrates the generation of random and cyclic DVRPs using the DBGP.

### *4.3. Real-world application model*

The DVRP generated from the DBGP as discussed above is in fact a DVRP with stochastic demands because the swap between two objects causes a change to the demands associated with these objects. The existing DVRP with stochastic demands models the following real-world situation. The vehicles may start their route as planned to distribute a number of goods to the customers. During the execution, a customer may notify the central depot that it needs more or less goods. Hence, the customers' demands change. However, the vehicles have already started their route considering the old demands. Therefore, a new route needs to be planned to satisfy the new demands of customers.

The difference of the existing model from the DBGP model lies in that in the former model, the demands arrive incrementally. For example, at the first stages of execution a predefined number of customer demands is visible. After a few stages, other customer demands become visible, and so on. In the latter model, all the customer demands are available from the beginning but they change during the execution.

Of course, in the real world, the demands are not only swapped between two customers. But, the DBGP still models the aforementioned real-world scenario. The reason that demands are pairwise-swapped in the model is for the sake of benchmarking. For example, when comparing algorithms on the DVRP with stochastic demands or the DVRP with traffic factors, the optimum value is unknown. Hence, even though one can see that one algorithm performs better than the other, it may be the case that both algorithms converge far away from the optimum. With the DBGP, the optimum value remains the same over environmental changes and, thus, one can see how close to the optimum an algorithm converges as well as comparing the performance between different algorithms. Apart from the comparison benefits, it is possible to assess the difficulty of a DOP. For instance, in case all the algorithms perform far away from the optimum on a particular DOP, and closer to the optimum on another DOP, it gives an indication that the former DOP is more difficult to solve than the latter one. Finally, the DBGP can be adopted easily to algorithms due to its simplicity.

## 5. Proposed immigrants schemes for the DVRP

### *5.1. Framework*

Conventional ACO algorithms are constructive heuristics. The solutions generated by the ants are not stored in an actual population, but only in the pheromone trails, which are used by the ants of the next iteration. In contrast, evolutionary algorithms consist of an actual population of feasible solutions, which is directly transferred from one iteration to the next using selection [28, 34]. Search operators (i.e., crossover and mutation) are used in evolutionary algorithms to generate the new population of solutions.

The investigated framework of ACO algorithms used to tackle DVRP maintains an actual population of the best ants of every iteration [35, 36]. The aim of the framework is to maintain the diversity within the population and transfer knowledge

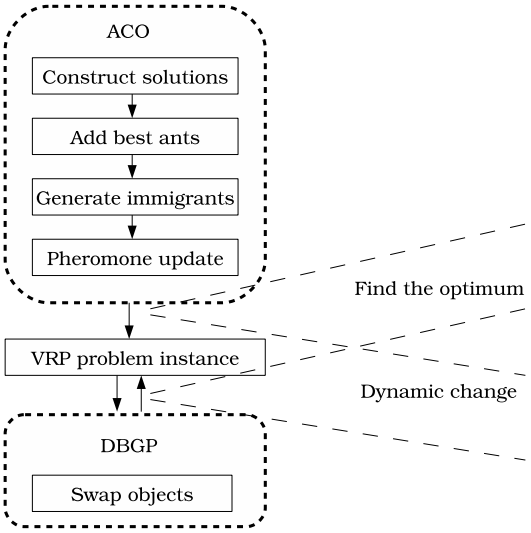


Figure 2: Illustration of the integration of the proposed ACO algorithms with the DBGP and the DVRP problem instance.

from previous environments to the pheromone trails of a new environment. More precisely, the framework re-generates the pheromone information for every iteration of running the algorithm, considering information only from the previous environment and extra information from the newly generated immigrant ants. Therefore, a short-term memory, denoted $k_{short}(t)$, of size $k_s$ is used where all ants stored from iteration $t-1$ are replaced by the best $k_s$ ants of the current iteration $t$. Furthermore, a number of immigrant ants are generated to replace the worst ants in the short-term memory. The overall ACO framework, integrated with the DBGP for the DVRP, is illustrated in Fig. 2.

Initially, all the pheromone trails are equally set to $\tau_0 = 1/C^{nn}$, where $C^{nn}$ is the quality of a solution constructed by a nearest neighbour heuristic and $k_{short}(0)$ is empty.

### 5.2. Constructing solutions

A population of $\mu$ ants start the solution construction from the depot since vehicles are supposed to start from the depot. Each ant selects the next customer to visit according to a probability defined by the pheromone trails and heuristic information as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in \mathcal{N}_i^k, \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{7}$$

where $p_{ij}^k$ is the probability of ant $k$ to visit customer $j$ when located to customer $i$, $\tau_{ij}$ is the existing pheromone trail on the link between customers $i$ and $j$, $\eta_{ij}$ is the heuristic information available a priori (i.e., $1/d_{ij}$), $\mathcal{N}_i^k$ denotes the neighbourhood of unvisited customers for ant $k$ when the current customer is $i$, $\alpha$ and $\beta$ are two parameters that determine the relative influence of $\tau$ and $\eta$, respectively.

When the choice of the next customer would lead to an infeasible solution (i.e., violating the maximum capacity of the vehicle) the depot is chosen, meaning that the vehicle will return to the depot, and a new vehicle route should start. This process is repeated until all customers' demands are satisfied and finally an ant constructs a feasible VRP solution. The number of routes in a VRP solution defines the number of vehicles used. An iteration $t$ is completed when all ants have constructed feasible solutions and a population $P(t)$ is generated.

### 5.3. Pheromone update

The difference of the ACO framework used in [35, 36] from the conventional ACO framework lies in that the short-term memory is associated with the pheromone matrix. At the end of an iteration, the $k_s$ best ants of $P(t)$ are added to $k_{short}(t)$. For each ant $k$ that enters $k_{short}(t)$, a positive pheromone update is performed, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^k, \forall\ (i,j) \in T^k, \tag{8}$$

where $T^k$ is the tour of ant $k$ $\Delta\tau_{ij}^k = (\tau_{max} - \tau_0)/k_s$ is the constant amount of pheromone that ant $k$ deposits to the corresponding trails, $\tau_{max}$ and $\tau_0$ are the maximum and initial pheromone value, respectively. Accordingly, a negative pheromone update is performed to each ant in $k_{short}(t-1)$ when it is removed to make space for ants in $k_{short}(t)$, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} - \Delta\tau_{ij}^k, \forall\ (i,j) \in T^k, \tag{9}$$

where $T^k$ and $\Delta\tau_{ij}$ are defined as in Eq. (8). This pheromone update policy is applied because no ant should survive in more than one iteration due to the dynamic environment.

Additionally, immigrant ants replace the worst ants in the short-term memory every iteration and further adjustments are performed to the pheromone trails, using Eq. (8) and (9), since the short-term memory changes. The main concern when dealing with immigrants schemes is how to generate immigrant ants that should represent feasible VRP solutions, which is described below.

### 5.4. Diversity maintaining schemes

#### 5.4.1. Random immigrants

Traditionally, immigrants are randomly generated and replace the worst ants in the population to maintain the diversity of the population. Here, a random immigrant ant for the DVRP is generated as follows. First, the depot is added as the starting point; then, an unvisited customer is randomly selected as the next point. This process is repeated until the current route of customers (starting from the most recent visit to the depot) violates the capacity constraint of the vehicle, or the depot is added as a component into the current route. When the capacity constraint is violated, the depot is added before the last customer selected for the current route and another vehicle route starts. When all customers' demands are satisfied, one feasible VRP solution is obtained. The difference of this random immigrant scheme from the one used in [35] lies in that the depot object

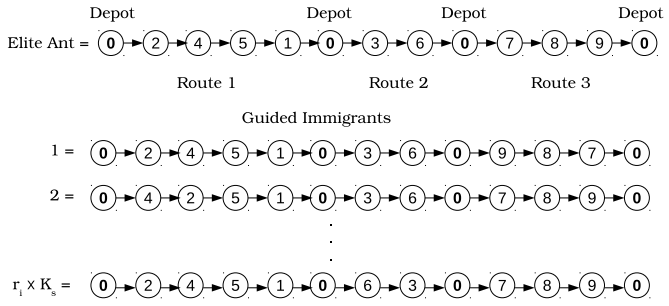


Figure 3: Illustration of immigrants generated in EIACO and MIACO for the DVRP.

also has chances to be added even when the capacity constraint is not violated.

Considering the investigated framework described above, before the pheromone trails are updated, $r_i \times k_s$ random immigrants are generated to replace the worst ants in $k_{short}(t)$, where $r_i$ is the immigrants replacement rate and $k_s$ is the size of short-term memory. Algorithm 2 presents the pseudocode of the RIACO investigated in this paper. RIACO is expected to handle well dynamic environments that change quickly and severely due to the diversity generated via random immigrants. This is because when the changing environments are not similar, it is better to randomly increase the diversity instead of knowledge transfer, which was confirmed for the dynamic travelling salesman problem [38]. Some preliminary experiments on the DVRP showed that RIACO totally disturbs the optimization process [35].

**Algorithm 2** RIACO and EIACO

```
1: t ← 0
2: P(0) ← InitializePopulation(μ)
3: InitiliazePheromoneTrails(τ_0)
4: k_short(0) ← empty
5: s^bs ← empty solution
6: while (termination condition not satisfied) do
7:    P(t) ← ConstructAntSolutions
8:    k_short(t) ← AddBestAnts(k_s)
9:    if (RIACO is selected) then
10:      GenerateRandomImmigrants(r_i)
11:      k'_short(t) ← ReplaceAntsWithImmigrants
12:   end if
13:   if (EIACO is selected) then
14:      s^elite ← FindBest(P(t − 1))
15:      GenerateElitismBasedImmigrants(s^elite', r_i)
16:      k'_short(t) ← ReplaceAntsWithImmigrants
17:   end if
18:   UpdatePheromone(k'_short(t))
19:   s^ib ← FindBest(P(t))
20:   if (f(s^ib) < f(s^bs)) then
21:      s^bs ← s^ib
22:   end if
23:   t ← t + 1
24: end while
25: return s^bs
```

### 5.4.2. Elitism-based immigrants

EIACO generates diversity via transferring knowledge from the best ant of the previous environment. An elitism-based immigrant ant for the DVRP is generated as follows. The best ant of the previous environment is selected in order to use it as the base to generate elitism-based immigrants. Swaps are performed between the customers with a mutation probability $p_m^i$ for each vehicle route of the VRP solution. Swaps between customers that belong to different routes are not allowed because it may lead to an infeasible solution. Fig. 3 illustrates the generation of an elitism-based immigrant for the DVRP. One difference of this elitism-based immigrant from the one used in [35] is that the depot objects remain to their position and the swap operator is applied. Another difference is that the elite ant used in [35] was the iteration-best ant just before a dynamic change, i.e., the best from $k_{short}(t-1)$, whereas in this paper it is the best-so-far ant of the previous environment. Note that the best-so-far ant is a special ant and may not necessarily belong to the current population of ants.

Considering the investigated framework described above, on iteration $t$, the best-so-far ant from the previous environment $P(t-1)$ is used as the base to generate $r_i \times k_s$ elitism-based immigrants, where $r_i$ and $k_s$ are defined as in the RIACO above. Algorithm 2 also presents the pseudocode of EIACO. The elitism-based immigrants replace the worst ants in $k_{short}(t)$ before the pheromone trails are updated. EIACO is expected to perform well on dynamic environments that are similar due to the knowledge transfer via elitism-based immigrants. This was confirmed for the dynamic travelling salesman problem [38]. Some preliminary experiments on the DVRP showed that EIACO outperforms RIACO in all dynamic test cases and that elitism-based immigrants improve the performance of ACO significantly [35].

### 5.4.3. Memory-based immigrants

MIACO maintains a long-term memory, denoted as $k_{long}(t)$, structure of limited size $k_l$, which is updated by replacing the closest ant in $k_{long}(t)$ with the best-so-far ant of $P(t)$. The metric to define how close ant $p$ is to ant $q$ is defined as follows:

$$M(p,q) = 1 - \left( \frac{c_{E_{pq}}}{n + avg(n_{V_p}, n_{V_q})} \right), \tag{10}$$

where $c_{E_{pq}}$ is the number of common edges (arcs) between ants $p$ and $q$, $n$ is the size of the problem instance, $n_{V_p}$ and $n_{V_q}$ are the number of vehicles of ants $p$ and $q$, respectively. A value $M(p,q)$ closer to 0 means that the two ants are similar.

Initially, $k_{long}(0)$ contains random VRP solutions. Therefore, the metric in Eq. (10) is triggered when all the random solutions are replaced. Every iteration $t$, the ants in $k_{long}(t)$ are repaired/re-evaluated in order to be valid with the newly generated environment in case a dynamic change occurs. Moreover, the solutions stored in the memory are used as detectors to detect an environmental change. If a dynamic change occurs, the cost of any solution stored in $k_{long}(t)$ at iteration $t+1$ will be

**Algorithm 3** UpdateMemory($k_{long}(t)$)

**if** ($t = t_M$) **then**
  $s^{mb} \leftarrow$ FindBest($P(t)$)
**end if**
**if** (dynamic change is *detected*) **then**
  $s^{mb} \leftarrow$ FindBest($P(t-1)$)
**end if**
**if** (still any random ant in $k_{long}(t)$) **then**
  ReplaceRandomWithBest($s^{mb}$,$k_{long}(t)$)
**else**
  $s^{cm} \leftarrow$ FindClosest($s^{mb}$,$k_{long}(t)$)
  **if** ($f(s^{mb}) < f(s^{cm})$) **then**
    $s^{cm} \leftarrow s^{mb}$
  **end if**
**end if**

different when it is re-evaluated. In this way, a dynamic change is detected.

The long-term memory is updated whenever a dynamic change is detected, where the best-so-far ant from $P(t-1)$ replaces the closest ant in $k_{long}(t)$ if its solution quality is better. However, the update does not depend only on the detection of dynamic changes since in some real-world applications it is difficult (or impossible) to detect changes. Therefore, instead of updating long-term memory only in a fixed time interval, e.g., every $f$ iterations, long-term memory is also updated in a dynamic pattern and the best-so-far ant from $P(t-1)$ replaces the closest ant in $k_{long}(t)$ as explained previously. For the dynamic pattern, for each update of the long-term memory, a random number in $[5, 10]$ (i.e., $rand(5, 10)$) is generated, which indicates the next update time. For example, if the memory is updated at iteration $t$ and no change has been detected before iteration $t_M = t + rand(5, 10)$, the next update will occur at iteration $t_M$ [56]. The ants that replace the closest ants in the memory for this paper are the best-so-far ants for both cases (i.e., update due to change detection and update due to the dynamic pattern), whereas in [36] they are the best ant from $k_{short}(t-1)$ and the iteration best ant for the two cases, respectively. The update procedures of the long-term memory for every iteration are presented in Algorithm 3.

MIACO generates diversity similarly to EIACO. The difference lies in that in MIACO the memory-best ant is selected as the base to generate memory-based immigrants. A memory-based immigrant ant for the DVRP is generated in exactly the same way as in EIACO (see Fig. 2). MIACO combines the merits of both memory schemes to guide the population directly to an old environment already visited and immigrants schemes to maintain diversity. MIACO stores the best solutions of previously optimized environments and reuses them to generate memory-based immigrants. It is very important to store different solutions in the long-term memory which represent different environments that might be useful in the future. This is achieved by the replacement strategy described in Eq. (10).

Considering the investigated framework described above, on iteration $t$, the best ant from $k_{long}(t)$ is used as the base to generate $r_i \times k_s$ memory-based immigrants, where $r_i$ and $k_s$ are defined as in the RIACO above. These immigrants replace the worst ants in $k_{short}(t)$ before the pheromone trails are updated. Algorithm 4 presents the pseudocode of MIACO. The algorithm is expected to do well in dynamic environments that are similar because of the knowledge transfer via memory-based immigrants (similar to elitism-based immigrants). Moreover, in case where the changing environments reappear, the memory guidance may speed up the re-optimization even more. This was confirmed for the dynamic travelling salesman problem [38]. Some preliminary experiments on the DVRP showed that MIACO performs better than other ACO approaches in cyclic environments [36]. However, it was not investigated whether the good performance of MIACO is only for cyclic dynamic environments, or only for random dynamic environments, or for both types of dynamic environments. This will be investigated in this paper.

**Algorithm 4** MIACO

$t \leftarrow 0$
$P(0) \leftarrow$ InitializePopulation($\mu$)
InitializePheromoneTrails($\tau_0$)
$k_{short}(0) \leftarrow$ *empty*
$k_{long}(0) \leftarrow$ InitializeRandomly($k_l$)
$t_M \leftarrow rand(5, 10)$
$s^{bs} \leftarrow$ *empty solution*
**while** (termination condition *not* satisfied) **do**
  $P(t) \leftarrow$ ConstructAntSolutions
  $k_{short}(t) \leftarrow$ AddBestAnts($k_s$)
  **if** ($t = t_M$ *or* dynamic change is *detected*) **then**
    *UpdateMemory*($k_{long}(t)$) using Algorithm 3
    $t_M \leftarrow t + rand(5, 10)$
  **end if**
  $s^{elite} \leftarrow$ FindBest($k_{long}(t)$)
  *GenerateMemoryBasedImmigrants*($s^{elite'}$,$r_i$)
  $k'_{short}(t) \leftarrow$ ReplaceAntsWithImmigrants
  UpdatePheromone($k'_{short}(t)$)
  $s^{ib} \leftarrow$ FindBest($P'(t)$)
  **if** ($f(s^{ib}) < f(s^{bs})$) **then**
    $s^{bs} \leftarrow s^{ib}$
  **end if**
  $t \leftarrow t + 1$
  $k_{long}(t) \leftarrow k_{long}(t-1)$
**end while**
**return** $s^{bs}$

## 6. Experimental study

### 6.1. Experimental setup

In the experiments, all the algorithms were tested on the DVRP instances that are constructed from three stationary benchmark VRP[1] instances taken from real life vehicle routing applications, which are visually illustrated in Fig. 4. F-n45-k4,

[1] Available in `http://neo.lcc.uma.es/vrp/`

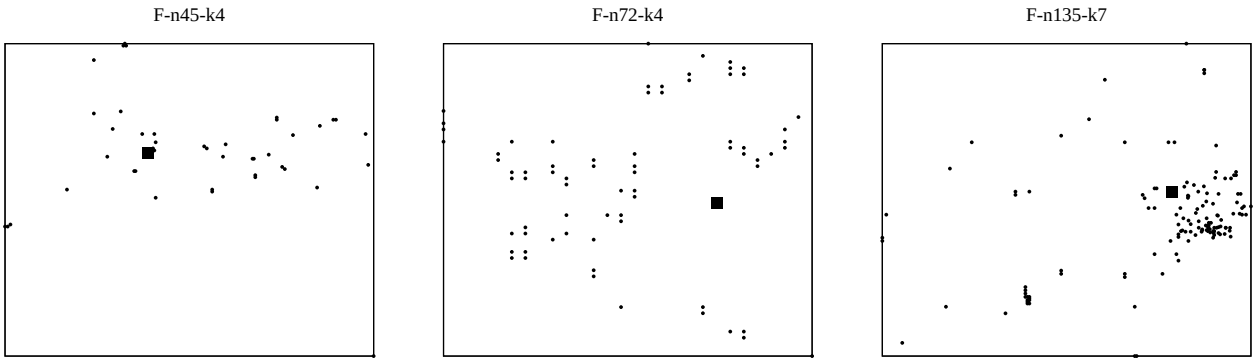


Figure 4: Topological structure of the three VRP benchmark instances used to generate dynamic environments (the square denotes the depot and the circles denote the customers.

F-n72-k4 and F-n135-k7 benchmarks have optimum values of 724, 237 and 1162, respectively [17]. Problems F-n45-k4 and F-n135-k7 represent grocery deliveries from the Peterboro and Brarmalea, Ontario terminals, respectively, of National Grocers Limited. Problem F-n72-k4 is concerned with the delivery of tires, batteries and accessories to gasoline service stations from Exxon in USA. F-n72-k4 is special in the sense that: all customers are located far away from the depot; some clients have very large demands making the capacity constraint crucial; and the customers are clustered as defined in [17].

For each stationary problem instance, we generate dynamic environments using the DBGP generator[2]. The frequency of change was set to $f = 10$ and $f = 100$, indicating environmental changes of high and low frequencies, respectively. The magnitude of change was set to $m = 0.1$, $m = 0.25$, $m = 0.5$, and $m = 0.75$, indicating the degree of environmental changes from small, to medium, and to large, respectively. As a result, eight DOPs (i.e., two values of $f \times$ four values of $m$) were generated from each stationary VRP instance in order to systematically analyse the adaptation and searching capabilities of each algorithm on the DVRP. Both random and cyclic DVRPs are generated in which the base states of cyclic DVRPs was set to $K = 4$. An observation of the best-so-far ant after a dynamic change was recorded every iteration and used to evaluate the performance. Therefore, the overall *offline performance* [29] is defined as follows:

$$\bar{P}_{OFF} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R} P^*_{ij}\right), \tag{11}$$

where $E$ is the total number of iterations, $R$ is the number of runs, and $P^*_{ij}$ is the best-so-far tour cost (after a change) of iteration $i$ of run $j$.

Moreover, the diversity of the population was recorded every iteration. The population *total diversity* [38] of ACO on a DOP is defined as:

$$\bar{T}_{DIV} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R} DIV_{ij}\right), \tag{12}$$

where $E$ and $R$ are as defined in Eq. (11) and $Div_{ij}$ is the population diversity at iteration $i$ of run $j$, which is defined as follows:

$$DIV_{ij} = \frac{1}{\mu(\mu-1)}\sum_{p=1}^{\mu}\sum_{q\neq p}^{\mu} M(p,q), \tag{13}$$

where $\mu$ is the population size and $M(p,q)$ is the similarity metric defined in Eq. (10). For each algorithm 1000 iterations were allowed and 30 independent runs were executed with the same set of random seeds.

Table 2: Parameter settings for the algorithms investigated

| Algorithm | $\alpha$ | $\beta$ | $\rho$ | $k_s$ | $k_l$ | $\mu$ |
|---|---|---|---|---|---|---|
| RIACO | 1 | 5 | - | 6 | - | 30 |
| EIACO | 1 | 5 | - | 6 | - | 30 |
| MIACO | 1 | 5 | - | 6 | 3 | 27 |
| $AS_{rank}$-CVRP | 1 | 5 | 0.3 | - | - | 30 |
| ACS-DVRP | 1 | 5 | 0.1 | - | - | 29 |
| M-ACO | 1 | 5 | - | - | 3 | 20 |
| $\mathcal{MM}AS_R$ | 1 | 5 | 0.5 | - | - | 29 |

In order to compare the proposed algorithms (RIACO, EIACO and MIACO), several other peer ACO algorithms for the DVRP are considered, which are described as follows:

- $AS_{rank}$-CVRP [6]: the first ACO algorithm applied to the basic VRP under stationary environments. Since $AS_{rank}$-CVRP can be applied directly to DOPs, this algorithm is applied to the DVRP.

- ACS-DVRP [41]: the first ACO algorithm applied to the DVRP with stochastic demands. It is an extension of the MACS-VRPTW algorithm that has state-of-the-art results in the VRP under stationary environments. When a dynamic change occurs, a pheromone conservation is performed to regulate previously generated pheromone trails.

[2]The implementation of the investigated ACO algorithms integrated with the DBGP is available from `http://www.tech.dmu.ac.uk/~mmavrovouniotis`

Table 3: Offline performance of ACO algorithms in random DVRPs

| Algorithms | F-n45-k4 | | | | F-n72-k4 | | | | F-n135-k7 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f = 10, m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| $AS_{rank}$-CVRP | 891.76 | 949.55 | 995.94 | 1011.63 | 302.11 | 324.84 | 352.82 | 365.53 | 1462.96 | 1550.41 | 1635.47 | 1664.46 |
| ACS-DVRP | 824.38 | 824.08 | 823.99 | 823.78 | 296.53 | 296.71 | 296.63 | 296.57 | 1385.62 | 1385.64 | 1384.62 | 1384.46 |
| M-ACO | 809.90 | 829.30 | 840.11 | 842.29 | 287.49 | 292.79 | 295.83 | 296.64 | 1357.10 | 1385.56 | 1401.68 | 1407.27 |
| $\mathcal{MM}AS_R$ | 818.20 | 818.38 | 818.37 | 818.52 | 294.20 | 294.27 | 294.27 | 294.23 | 1363.11 | 1364.11 | 1363.49 | 1363.49 |
| RIACO | 806.51 | 817.20 | 824.98 | 826.29 | 289.78 | 291.44 | 292.65 | 292.86 | 1342.80 | 1353.64 | 1361.78 | 1363.69 |
| EIACO | 804.88 | 815.38 | 826.36 | 827.93 | 285.25 | 289.34 | 291.57 | 291.98 | 1324.93 | 1349.47 | 1364.64 | 1369.28 |
| MIACO | 806.83 | 818.46 | 830.23 | 832.13 | 286.04 | 290.59 | 293.35 | 294.07 | 1333.83 | 1355.97 | 1373.73 | 1379.00 |
| $f = 100, m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| $AS_{rank}$-CVRP | 916.10 | 962.11 | 983.84 | 972.29 | 290.76 | 295.08 | 304.77 | 308.78 | 1375.42 | 1398.67 | 1421.08 | 1419.08 |
| ACS-DVRP | 807.80 | 808.07 | 808.08 | 807.62 | 288.53 | 288.34 | 288.43 | 288.26 | 1351.98 | 1352.12 | 1352.33 | 1352.61 |
| M-ACO | 801.81 | 804.65 | 808.34 | 807.66 | 272.99 | 274.59 | 275.87 | 275.82 | 1287.95 | 1299.78 | 1311.63 | 1315.98 |
| $\mathcal{MM}AS_R$ | 803.15 | 802.97 | 803.51 | 803.69 | 273.99 | 274.21 | 273.92 | 274.29 | 1296.62 | 1296.59 | 1296.79 | 1295.96 |
| RIACO | 800.80 | 801.89 | 804.26 | 804.27 | 279.65 | 275.92 | 279.94 | 279.60 | 1304.33 | 1307.52 | 1310.17 | 1309.36 |
| EIACO | 800.10 | 802.04 | 804.73 | 804.27 | 271.94 | 270.61 | 275.27 | 275.22 | 1275.59 | 1286.44 | 1292.68 | 1296.08 |
| MIACO | 800.84 | 803.63 | 805.17 | 806.67 | 274.22 | 276.46 | 278.20 | 277.83 | 1290.60 | 1297.93 | 1303.76 | 1309.24 |

Table 4: Offline performance of ACO algorithms in cyclic DVRPs

| Algorithms | F-n45-k4 | | | | F-n72-k4 | | | | F-n135-k7 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f = 10, m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| $AS_{rank}$-CVRP | 851.66 | 878.74 | 915.08 | 916.73 | 281.58 | 299.61 | 312.92 | 313.48 | 1398.11 | 1464.67 | 1488.55 | 1500.47 |
| ACS-DVRP | 824.53 | 824.00 | 823.80 | 823.86 | 296.59 | 296.71 | 296.56 | 296.46 | 1385.49 | 1385.42 | 1385.45 | 1384.72 |
| M-ACO | 814.74 | 830.61 | 842.02 | 843.13 | 286.23 | 292.37 | 295.87 | 296.56 | 1353.64 | 1379.18 | 1403.94 | 1409.14 |
| $\mathcal{MM}AS_R$ | 818.31 | 818.28 | 818.42 | 818.72 | 294.31 | 295.98 | 294.07 | 295.77 | 1372.93 | 1363.11 | 1363.15 | 1363.41 |
| RIACO | 808.74 | 817.20 | 826.09 | 826.09 | 289.25 | 290.91 | 292.48 | 292.39 | 1344.96 | 1352.38 | 1359.92 | 1363.94 |
| EIACO | 814.31 | 817.39 | 827.68 | 828.99 | 284.06 | 288.82 | 291.18 | 291.54 | 1321.43 | 1345.27 | 1362.93 | 1369.84 |
| MIACO | 813.41 | 813.73 | 822.70 | 825.35 | 283.08 | 288.22 | 289.29 | 289.72 | 1320.35 | 1338.33 | 1354.38 | 1362.56 |
| $f = 100, m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| $AS_{rank}$-CVRP | 918.87 | 958.38 | 966.54 | 970.84 | 283.63 | 292.31 | 302.14 | 303.15 | 1367.31 | 1396.71 | 1407.66 | 1410.67 |
| ACS-DVRP | 808.05 | 807.97 | 807.76 | 807.87 | 288.47 | 288.78 | 288.45 | 288.16 | 1352.70 | 1352.56 | 1352.21 | 1351.90 |
| M-ACO | 802.14 | 805.17 | 808.28 | 808.91 | 272.49 | 274.86 | 275.24 | 276.24 | 1289.24 | 1299.45 | 1314.23 | 1316.91 |
| $\mathcal{MM}AS_R$ | 803.08 | 803.93 | 803.05 | 802.53 | 273.87 | 273.98 | 273.96 | 273.90 | 1296.39 | 1296.93 | 1297.23 | 1297.67 |
| RIACO | 800.80 | 803.06 | 804.38 | 804.74 | 279.46 | 276.08 | 279.81 | 279.65 | 1306.86 | 1306.50 | 1308.16 | 1309.98 |
| EIACO | 801.75 | 802.10 | 803.56 | 804.67 | 271.87 | 271.33 | 274.85 | 275.07 | 1273.62 | 1287.56 | 1292.89 | 1295.10 |
| MIACO | 802.23 | 802.32 | 804.86 | 803.70 | 273.26 | 276.13 | 275.86 | 275.76 | 1287.65 | 1294.24 | 1301.00 | 1300.30 |

- $\mathcal{MM}AS_R$ [37]: one of the best performing ACO algorithms on several combinatorial optimization problems. Recently, it has been used on dynamic routing problems. A mechanism is used to detect environmental changes in order to perform pheromone re-initialization and start the optimization process from scratch.
- M-ACO [39]: a memetic algorithm that uses the same framework with the population-based ACO [26]. However, before entering the population-list, the best ant passes from several local search improvements based on simple and adaptive swaps. It also has a diversity scheme, i.e., triggered random immigrants, where a random immigrant ant enters the population-list whenever all the ants in the population-list are identical.

All the algorithms are benchmarked on the different dynamic test cases generated. The algorithmic parameters used in the experiments are presented in Table 2. In order to have a fair comparison all the algorithms perform the same number of function evaluations in every algorithmic iteration. Hence, the population size $\mu$ of the ACO algorithm was set accordingly. For example, MIACO, ACS-DVRP, M-ACO and $\mathcal{MM}AS_R$ use extra function evaluations (from detectors) in order to detect dynamic changes. Therefore, the population size was set according to the number of detectors used. Furthermore, M-ACO perform 7 local search steps for solution improvements. The immigrant replacement rate for RIACO, EIACO and MIACO was set to $r_i = 0.4$ and the immigrant mutation probability for EIACO and MIACO was set to $p_m^i = 0.01$.

### *6.2. Results of comparing proposed ACO algorithms*

The experimental results regarding the performance of RIACO, EIACO and MIACO are presented in Table 3 and Table 4 for random DVRPs and cyclic DVRPs, respectively. The corresponding statistical tests are presented in Table 5, where Kruskal–Wallis tests were applied followed by posthoc paired comparisons using Mann–Whitney tests with the Bonferroni correction. In Table 5, the results are shown as "−", "+" or "∼" when the first algorithm is significantly better than the second one, when the second algorithm is significantly better than the first one, or when the two algorithms are not significantly different, respectively. In order to better understand the behaviour of the algorithms in dynamic environments, their offline performance against iterations is plotted in Fig. 5 on cyclic DVRPs with $f = 10$ and $m = 0.25$ for the last ten environmental changes and in Fig. 6 on random DVRPs with $f = 100$ and

Table 5: Statistical test results of comparing the offline performance of ACO algorithms in random and cyclic DVRPs

| Statistical tests | F-n45-k4 | | | | | | | | F-n72-k4 | | | | | | | | F-n135-k7 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Environment Dynamics | $f$ = 10 | | | | $f$ = 100 | | | | $f$ = 10 | | | | $f$ = 100 | | | | $f$ = 10 | | | | $f$ = 100 | | | |
| *Random*, $m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| RIACO⇔EIACO | + | + | − | − | ∼ | ∼ | ∼ | ∼ | + | + | + | + | + | + | + | + | + | + | − | − | + | + | + | + |
| RIACO⇔MIACO | ∼ | ∼ | − | − | ∼ | ∼ | ∼ | − | + | + | − | − | + | ∼ | + | + | + | − | − | − | + | + | + | ∼ |
| EIACO⇔MIACO | − | − | − | − | ∼ | ∼ | ∼ | − | ∼ | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| RIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| RIACO⇔ACS-DVRP | − | − | ∼ | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| RIACO⇔M-ACO | − | − | − | − | ∼ | ∼ | − | − | + | ∼ | − | − | + | + | + | + | − | − | − | − | + | + | − | ∼ |
| RIACO⇔ $\mathcal{MM}$AS$_R$ | − | − | + | + | − | ∼ | ∼ | ∼ | − | − | − | − | + | + | + | + | − | − | − | ∼ | + | + | + | + |
| EIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| EIACO⇔ACS-DVRP | − | − | + | + | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| EIACO⇔M-ACO | − | − | − | − | − | ∼ | − | − | − | − | − | − | ∼ | − | ∼ | ∼ | − | − | − | − | ∼ | − | − | − |
| EIACO⇔ $\mathcal{MM}$AS$_R$ | − | − | + | + | − | ∼ | + | ∼ | − | − | − | − | − | − | + | ∼ | − | − | + | + | − | − | ∼ | ∼ |
| MIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| MIACO⇔ACS-DVRP | − | − | + | + | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| MIACO⇔M-ACO | − | − | − | − | ∼ | ∼ | ∼ | ∼ | ∼ | − | − | − | + | + | + | + | − | − | − | − | ∼ | ∼ | ∼ | ∼ |
| MIACO⇔ $\mathcal{MM}$AS$_R$ | − | ∼ | + | + | − | ∼ | ∼ | + | − | − | − | ∼ | ∼ | + | + | + | − | − | + | + | ∼ | ∼ | + | + |
| *Cyclic*, $m \Rightarrow$ | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 | 0.1 | 0.25 | 0.5 | 0.75 |
| RIACO⇔EIACO | − | ∼ | − | − | ∼ | ∼ | ∼ | ∼ | + | + | + | + | + | + | + | + | + | + | − | − | + | + | + | + |
| RIACO⇔MIACO | − | + | + | ∼ | ∼ | ∼ | ∼ | ∼ | + | + | + | + | + | ∼ | + | + | + | + | + | ∼ | + | + | + | + |
| EIACO⇔MIACO | ∼ | + | + | + | ∼ | ∼ | ∼ | ∼ | ∼ | ∼ | + | + | ∼ | − | ∼ | ∼ | ∼ | + | + | + | − | − | − | − |
| RIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | + | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| RIACO⇔ACS-DVRP | − | − | + | + | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| RIACO⇔M-ACO | − | − | − | − | ∼ | ∼ | − | − | + | ∼ | − | − | + | + | + | + | ∼ | − | − | − | + | + | ∼ | ∼ |
| RIACO⇔ $\mathcal{MM}$AS$_R$ | − | − | + | + | − | ∼ | ∼ | + | − | − | − | − | + | + | + | + | − | − | − | ∼ | + | + | + | + |
| EIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | ∼ | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| EIACO⇔ACS-DVRP | − | − | + | + | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| EIACO⇔M-ACO | − | − | − | − | − | ∼ | − | − | ∼ | − | − | − | ∼ | − | ∼ | ∼ | − | − | − | − | − | − | − | − |
| EIACO⇔ $\mathcal{MM}$AS$_R$ | − | ∼ | + | + | ∼ | − | ∼ | + | − | − | − | − | − | − | ∼ | + | − | − | ∼ | + | − | − | ∼ | ∼ |
| MIACO⇔AS$_{rank}$-CVRP | − | − | − | − | − | − | − | − | ∼ | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| MIACO⇔ACS-DVRP | − | − | ∼ | ∼ | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − |
| MIACO⇔M-ACO | ∼ | − | − | − | ∼ | ∼ | ∼ | − | ∼ | − | − | − | ∼ | + | ∼ | ∼ | − | − | − | − | ∼ | ∼ | − | − |
| MIACO⇔ $\mathcal{MM}$AS$_R$ | − | − | + | + | ∼ | − | + | + | − | − | − | − | ∼ | + | + | + | − | − | − | ∼ | − | ∼ | ∼ | ∼ |

$m = 0.25$ for the first five environmental changes, respectively. The corresponding dynamic total diversity of the algorithms against iterations on DVRPs is plotted in Fig. 7. From the experimental results, several observations regarding the weaknesses and strengths of the proposed algorithms can be made and analysed below.

First, RIACO significantly outperforms EIACO in most random and cyclic DVRPs with $f = 10$ and $m = 0.5$, and with $f = 10$ and $m = 0.75$, whereas EIACO significantly outperforms RIACO in most random and cyclic DVRPs with $f = 10$ and $m = 0.1$, and with $f = 10$ and $m = 0.75$; see the comparisons regarding RIACO ⇔ EIACO in Table 5. This is because EIACO requires the changing environments to be similar in order for the knowledge transferred from the previous environment via elitism-based immigrants to be suitable for the newly generated environment. Differently, RIACO often maintains higher diversity than EIACO, which can be observed from Fig. 7, that is useful in changing environments where knowledge transfer is inconvenient (e.g., on severely and quickly changing environments).

Second, EIACO significantly outperforms RIACO in most random and cyclic DVRPs with $f = 100$; see the comparisons regarding RIACO ⇔ EIACO in Table 5. This is because EIACO has enough time to gain knowledge from previous environment and transfer it to the newly generated environment. The effect of the elitism mechanism can be observed from Fig. 6, where EIACO maintains a better solution quality than other ACO algorithms in most environmental changes. Furthermore, EIACO significantly outperforms MIACO in most random DVRPs with $f = 100$, whereas it has comparable performance in most cyclic random DVRPs with $f = 100$. Even though both EIACO and MIACO generate immigrants that transfer knowledge from previous environments, the memory best ant in MIACO may misguide the population into non-promising areas. On the other hand, the best ant of the previous environment in EIACO requires enough time to express its effect in DVRPs.

Third, MIACO significantly outperforms RIACO and EIACO in most cyclic DVRPs with $f = 10$, whereas RIACO and EIACO significantly outperform MIACO in most random DVRPs with $f = 10$; see the comparisons regarding RIACO ⇔ MIACO and EIACO ⇔ MIACO in Table 5. This is because MIACO can move the population directly to a previously visited environment via memory-based immigrants. This can be observed from Fig. 6, where MIACO is able to maintain better performance over EIACO. Furthermore, EIACO significantly outperforms MIACO in most cyclic DVRPs with $f = 100$, whereas MIACO outperforms EIACO in cyclic DVRPs with $f = 10$. This can be explained by the generated cyclic environments. For example, the number of base states used in the generated

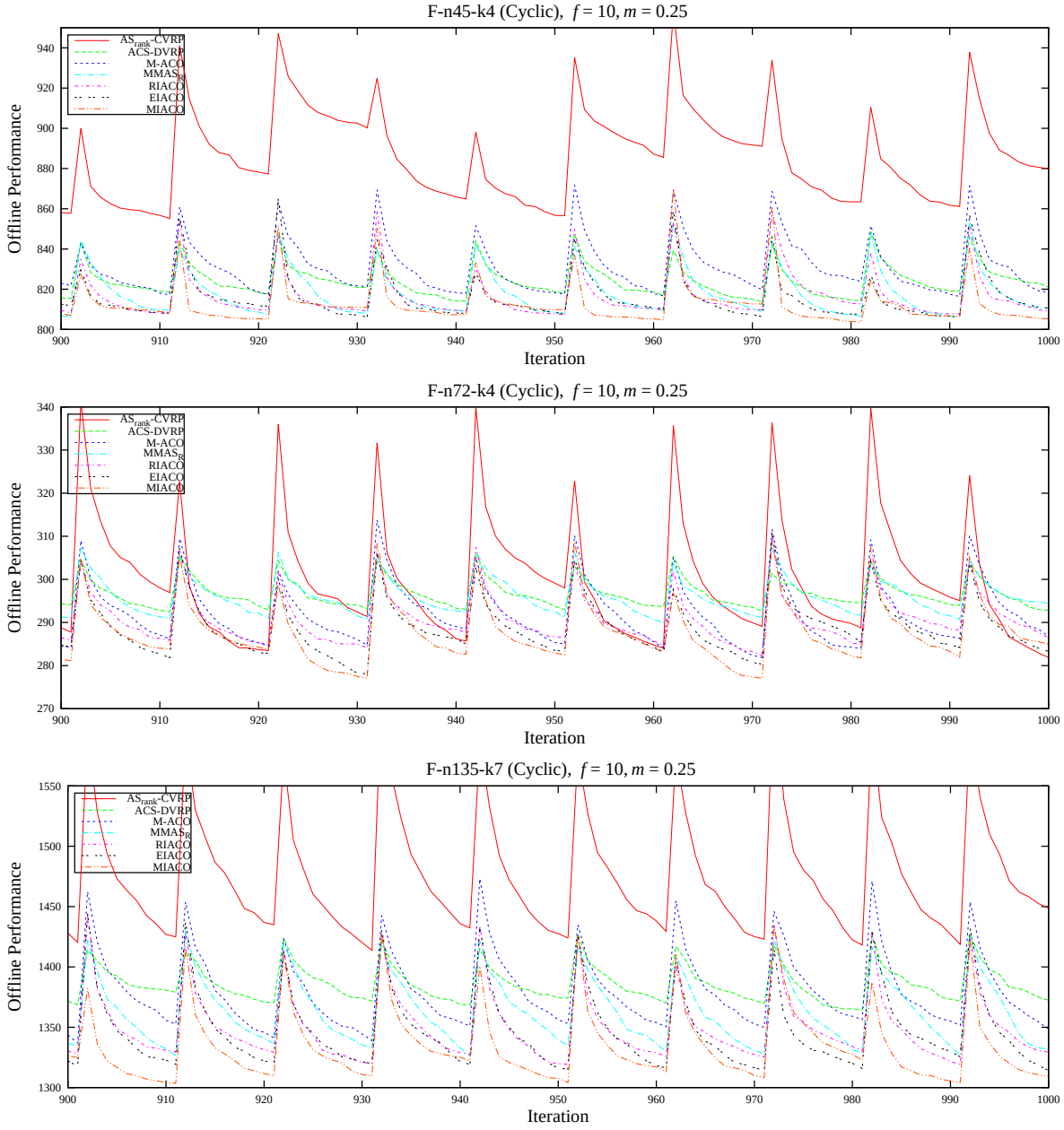


Figure 5: Dynamic offline performance of ACO algorithms in cyclic DVRPs with $f = 10$ and $m = 0.25$ for the last ten environments.

cyclic DVRP for this study was set to $K = 4$. Hence, the changing environment cycles 2.5 and 25 times, when $f = 100$ and $f = 10$, respectively, for $G = 1000$ iterations(i.e., $(G/f)/K$). As a result, MIACO cycles more times when $f = 10$, and the knowledge stored in the memory becomes more accurate. MIACO requires enough cycles of the changing environment to express its effect on DVRPs.

Generally, the observations of RIACO, EIACO and MIACO in this paper are different from the ones in [35, 36]. In the previous papers [35, 36], the performance of the algorithms was unexpected, even though the DVRP with traffic factors was used, since EIACO and MIACO had better performance than RIACO in all dynamic test cases. This is because the immigrant ants were extended from the dynamic travelling salesman problem to the DVRP directly, whereas in this paper they are developed especially for the DVRP. For example, the resulting immigrant in EIACO now has more chances to be similar to the elite ant since the depot object remains unchanged and the different vehicle routes are mutated independently (see Fig. 3); or the resulting immigrant in RIACO now may cover some routes that are less attractive but may be useful to the newly generated environment.

Finally, the proposed algorithms and most of the existing algorithms (except AS$_{rank}$-CVRP) converge into near to the optimum solutions. AS$_{rank}$-CVRP performs far away from the optimum because it was proposed for stationary environments and does not have any enhancements to address DOPs. This can be observed from Fig. 6 in which the algorithm performs similar

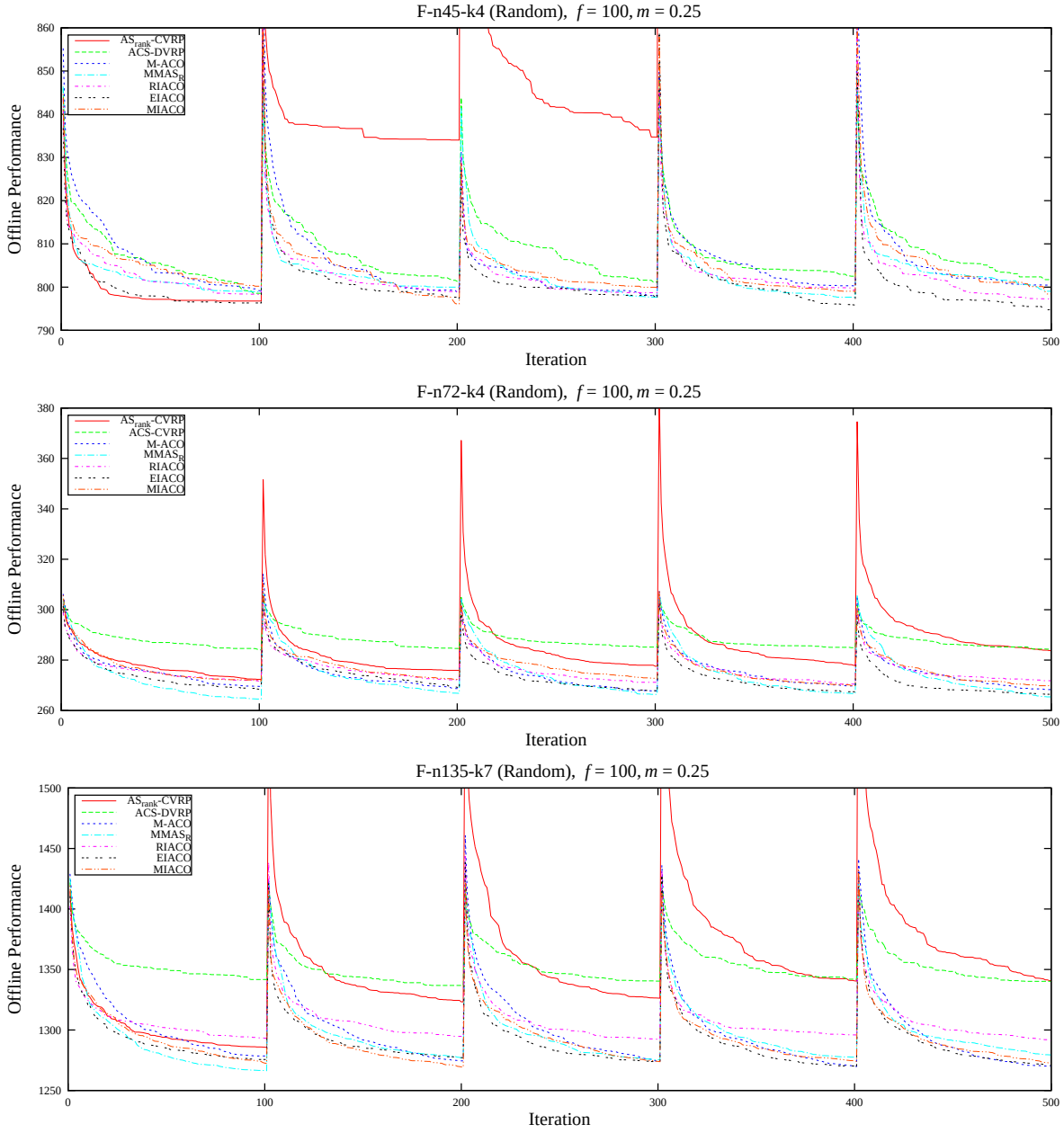


Figure 6: Dynamic offline performance of ACO algorithms in random DVRPs with $f = 100$ and $m = 0.25$ for the first five environmental changes.

with its competitors on the first environment and moves away as the environment changes. This confirms our claim that conventional ACO algorithms cannot track the moving optimum efficiently. Furthermore, on DVRPs with small values of $m$ and $f$ the algorithms perform closer to the optimum. As the $m$ and $f$ values increase the algorithm are moving away from optimum. This indicates that quickly changing environments and severely changing environments are more difficult to address. This is natural because the algorithms may not have enough time to converge and the knowledge transferred may misguide the searching process in quickly and severely changing environments, respectively.

### *6.3. Results of comparing RIACO against other peer algorithms*

RIACO outperforms M-ACO and $\mathcal{MM}\text{AS}_R$ in most DVRPs with $f = 10$ whereas it is outperformed in most DVRPs with $f = 100$, both random and cyclic. This is because RIACO may disturb the optimization process due to too much randomization, especially when enough re-optimization time is available, e.g., on DVRPs with $f = 100$. This can be observed in Fig. 7, where RIACO maintains higher diversity levels on DVRPs with $f = 100$ over other ACO algorithms that transfer knowledge via previous pheromone trails. Even though $\mathcal{MM}\text{AS}_R$ does not transfer knowledge, it also outperforms RIACO in most DVRPs with $f = 100$. This is natural because the global re-initialization of pheromone trails has enough time to re-optimize to the newly

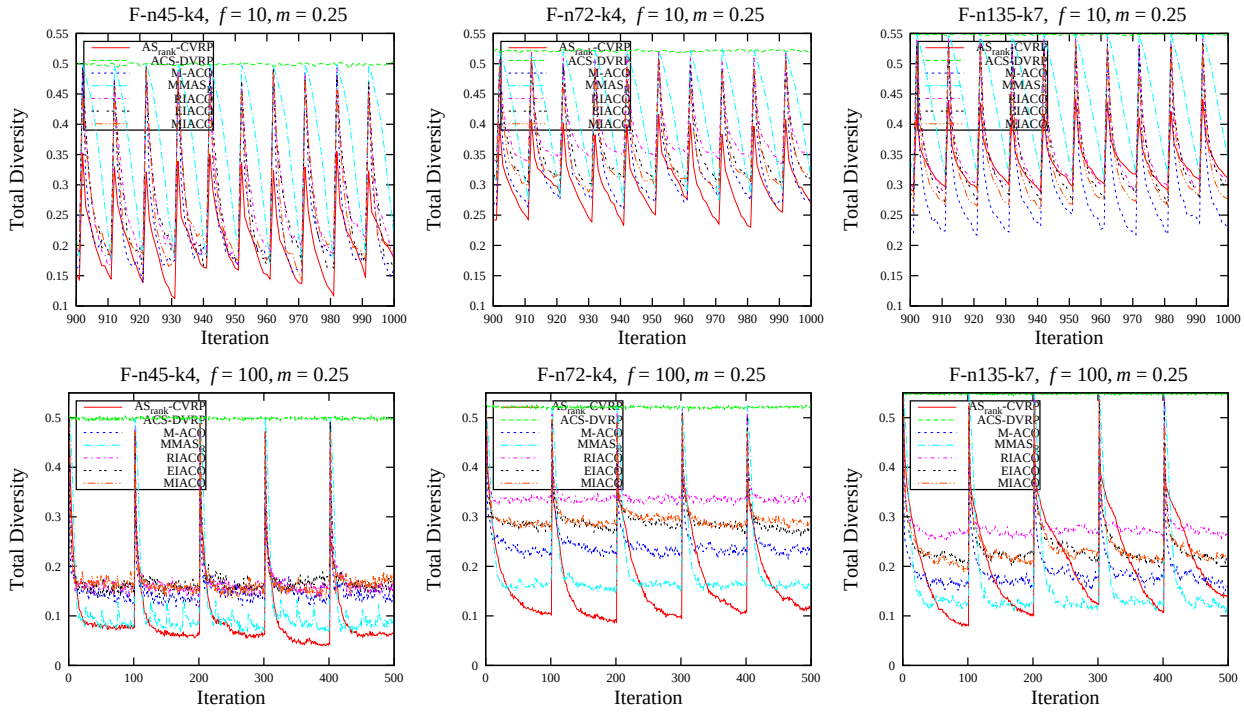


Figure 7: Dynamic population diversity of ACO algorithms in quickly changing cyclic DVRPs (top) and slowly changing random DVRPs (bottom) for the last ten and first five environmental changes, respectively.

generated optimum. Furthermore, RIACO clearly outperforms AS$_{rank}$-CVRP and ACS-DVRP in almost all DVRPs, both random and cyclic. AS$_{rank}$-CVRP and ACS-DVRP use pheromone evaporation and may destroy knowledge from the previous environment or may not be able to eliminate pheromone trails that bias ants into non-promising areas. This can be observed from Fig. 6, where the performance of AS$_{rank}$-CVRP is degraded after the first environmental change.

### 6.4. Results of comparing EIACO against other peer algorithms

EIACO outperforms AS$_{rank}$-CVRP and ACS-DVRP in almost all DVRPs, both random and cyclic, for the same reasons explained previously. Moreover, EIACO outperforms M-ACO and $\mathcal{MM}$AS$_R$ in most DVRPs, both random or cyclic. More precisely, in DVRPs with $f = 10$, EIACO is able to transfer knowledge more effectively than its competitors due to the diversity maintenance mechanism via the elitism-based immigrants, which helps to accept the knowledge transferred. This can be observed in Fig. 7, where the diversity maintenance mechanisms of the competitors are not very effective because the diversity level after a dynamic change is decreased dramatically (except $\mathcal{MM}$AS$_R$). Therefore, when the diversity is decreased, meaning that the population converged towards a solution, it will be difficult to accept the knowledge transferred in order to locate the moving optimum after a dynamic change. Similarly, in DVRPs with $f = 100$, EIACO has even more time to transfer knowledge from previous environments and express its effect. This can be observed in Fig. 6, where EIACO converges faster than other algorithms and to a better solution (after a few environments).

### 6.5. Results of comparing MIACO against other peer algorithms

MIACO outperforms AS$_{rank}$-CVRP and ACS-DVRP in almost all dynamic test cases, both random and cyclic, for the same reasons described previously. The performance of MIACO on random DVRPs is similar to or slightly worse than the performance of EIACO. This is due to the fact that MIACO is a generalization of EIACO. Furthermore, MIACO outperforms M-ACO and $\mathcal{MM}$AS$_R$ on most cyclic DVRPs with $f = 10$, but it is comparable on most cyclic DVRPs with $f = 100$. As it was explained previously, the more times the environment cycles, the better the performance of MIACO is; which can be observed in Fig. 5, where MIACO is able to maintain better solution quality than other algorithms.

### 6.6. Sensitivity analysis on the effect of parameters $r_i$ and $p_m^i$

There are several key parameters within the proposed ACO algorithms, such as the immigrants replacement rate $r_i$, which determines the number of immigrant ants introduced to the current short-term memory, and the mutation probability $p_m^i$ within the elitism-based immigrants, which determines the level of diversity generated by the immigrants. In the basic experiments, we have set $r_i = 0.4$ for RIACO, EIACO and MIACO, and $p_m^i = 0.01$ for EIACO and MIACO. In order to investigate the effect of these parameters, we further carried out experiments on RIACO, EIACO and MIACO on DVRPs with $f = 100$ and $m = 0.1$ and with $f = 10$ and $m = 0.75$. The value of $r_i$ was set to $r_i \in \{0.0, 0.2, 0.4, 0.6, 0.8\}$ and the value of

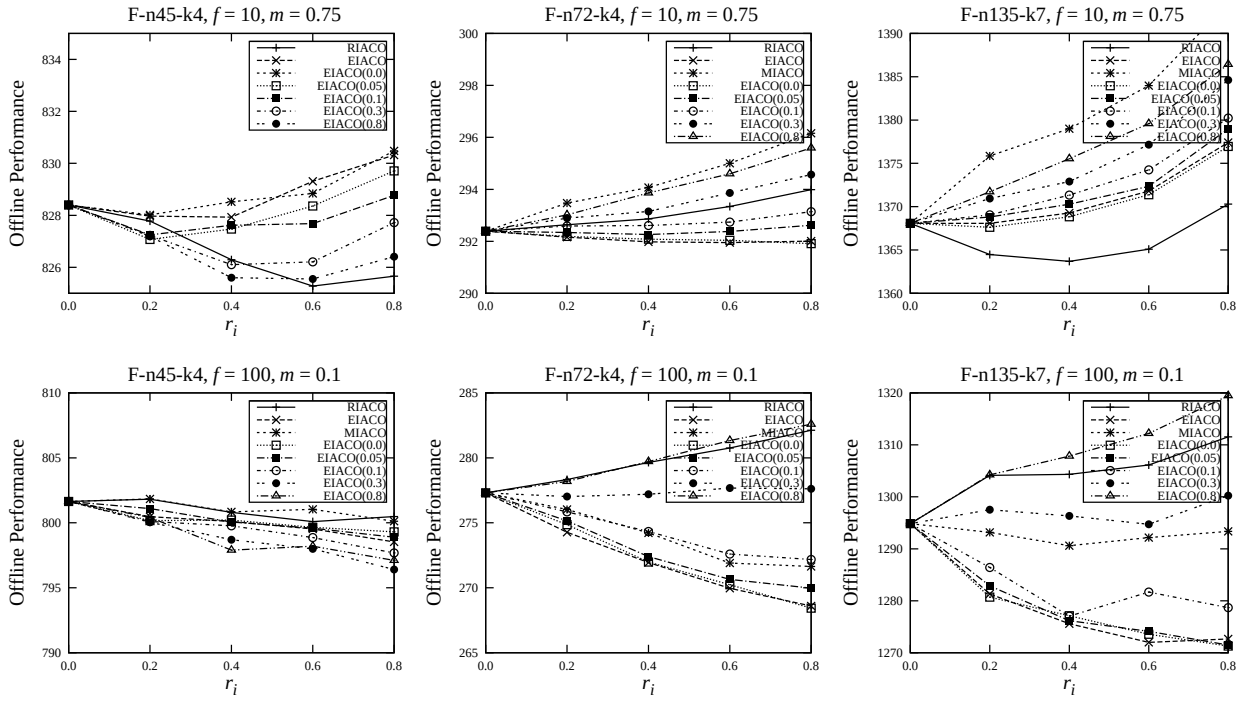


Figure 8: Offline performance of RIACO, EIACO and MIACO with different immigrants replacement rates in random DVRPs.

$p_m^i \in \{0.05, 0.1, 0.3, 0.8\}$, while the remaining experimental settings were the same as in the basic experiments. The experimental results on random DVRPs with the aforementioned dynamic properties are plotted in Fig. 8, where the EIACO with different $p_m^i$ values is denoted as EIACO($p_m^i$). From Fig. 8, several observations can be drawn as follows.

First, the immigrants replacement rate $r_i$ does affect the performance of relevant ACO algorithms. For example, the performance of EIACO is improved on DVRPs with $f = 100$ and $m = 0.1$ whereas the performance of RIACO is improved on DVRPs with $f = 10$ and $m = 0.75$ when $r_i > 0$. However, on DVRPs with $f = 10$ and $m = 0.75$, the performance of EIACO and RIACO is degraded and upgraded, respectively, as $r_i$ increases, whereas on DVRPs with $f = 100$ and $m = 0.75$ the performance of EIACO and RIACO is upgraded and degraded, respectively. These observations further support the claims in the basic experiments that different immigrants schemes perform well on DVRPs with different properties.

Second, the immigrants mutation probability $p_m^i$ also affects the performance of relevant ACO algorithms. In most cases, EIACO performs better than other EIACO with different $p_m^i$ values even when the parameter $r_i$ is different. There is no any significant difference between EIACO and EIACO(0.05). However, as $p_m^i$ increases, e.g., EIACO(0.3) and EIACO(0.8), the performance is degraded (except on F-n45-k4) since randomization is promoted.

Generally, the two investigated parameters control the diversity generated by the immigrants and their sensitivity depends on the properties of a DOP, e.g., $f$ and $m$, but also the type of the algorithm. For example, the parameter $r_i$ in RIACO is very sensitive since it may disturb the optimization process, whereas a similar case occurs with the parameter $p_m^i$ in EIACO on DVRPs with $f = 100$. In contrast, a higher value of $r_i$ in EIACO usually improves the performance.

### 6.7. Experiments on DVRPs with traffic factors

In the basic experiments above the DBGP was used to generated DVRPs for benchmarking purposes. In this section, the DVRP with traffic factors[3] [35, 36] is considered that models a more realistic scenario. The cost of each connection $(i, j) \in A$ is defined as $d_{ij} \times t_{ij}$, where $d_{ij}$ is the normal distance travelled defined in Eq. (2) and $t_{ij}$ represents the traffic factor between customers $i$ and $j$. Every $f$ algorithmic iterations, a random number (i.e., $t_{ij} = rand(F_L, F_U)$) is generated to represent potential traffic jams, where $F_L$ and $F_U$ are the lower and upper bounds of the traffic factor, respectively. Each connection has a probability $m$ to add traffic, whereas the remaining links are set to $t_{ij} = 1$ which indicates no traffic. Furthermore, in the basic experiments the performance of ACO algorithms was investigated on DVRPs with fixed values of $f$ and $m$ for comparison purposes. However, real-world problems may involve different periods and severities of change. In order to investigate the performance of ACO algorithms in such environments, further experiments were performed with random $f$ and $m$ values, which were randomly generated in $[1, 100]$ and $[0, 1]$ (i.e., $f = rand(1, 100)$ and $m = rand(0, 1)$), respectively.

The experimental results regarding the offline performance of ACO algorithms are presented in Table 6 with the corresponding statistical test results performed in the same way as in the

[3]Note that the optimum values in these DOPs are unknown during the execution of the algorithms.

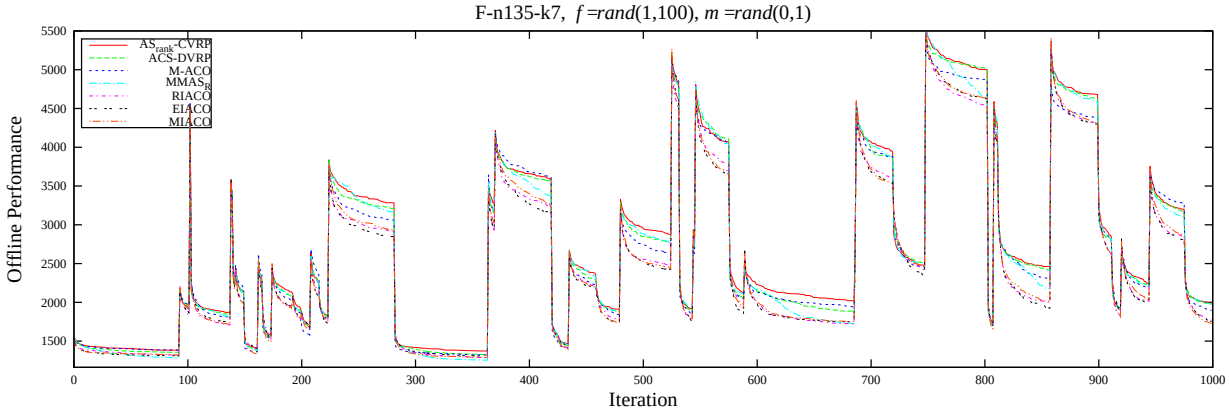


Figure 9: Dynamic offline performance of ACO algorithms in DVRPs with traffic factors with $f = rand(1, 100)$ and $m = rand(0, 1)$.

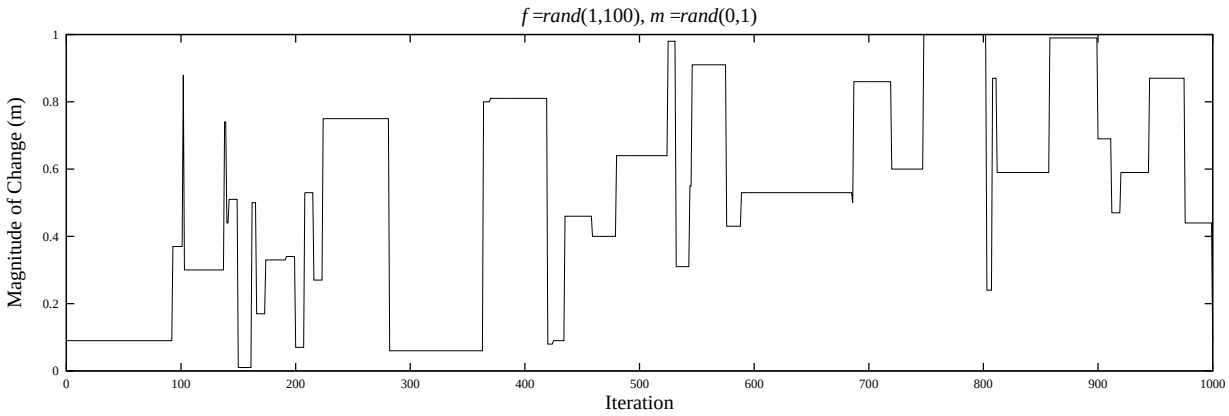


Figure 10: Values of $f = rand(1, 100)$ and $m = rand(0, 1)$.

Table 6: Offline performance and statistical test results on DVRPs with traffic factor with $f = rand(1, 100)$ and $m = rand(0, 1)$

| DVRPs | F-n45-k4 | F-n72-k4 | F-n135-k7 |
|---|---|---|---|
| Offline Performance | | | |
| $AS_{rank}$-CVRP | 1126.69 | 473.27 | 2695.42 |
| ACS-DVRP | 1118.71 | 469.58 | 2656.18 |
| M-ACO | 1177.52 | 466.98 | 2618.42 |
| $\mathcal{MM}AS_R$ | 1112.49 | 461.09 | 2626.73 |
| RIACO | 1106.19 | 436.11 | 2496.29 |
| EIACO | 1117.49 | 429.21 | 2483.48 |
| MIACO | 1120.38 | 433.46 | 2505.39 |
| Statistical Test | | | |
| RIACO⇔EIACO | − | + | + |
| RIACO⇔MIACO | − | + | ∼ |
| EIACO⇔MIACO | ∼ | − | − |
| RIACO⇔$AS_{rank}$-CVRP | − | − | − |
| RIACO⇔ACS-DVRP | − | − | − |
| RIACO⇔M-ACO | − | − | − |
| RIACO⇔ $\mathcal{MM}AS_R$ | − | − | − |
| EIACO⇔$AS_{rank}$-CVRP | − | − | − |
| EIACO⇔ACS-DVRP | ∼ | − | − |
| EIACO⇔M-ACO | − | − | − |
| EIACO⇔ $\mathcal{MM}AS_R$ | ∼ | − | − |
| MIACO⇔$AS_{rank}$-CVRP | − | − | − |
| MIACO⇔ACS-DVRP | ∼ | − | − |
| MIACO⇔M-ACO | − | − | − |
| MIACO⇔ $\mathcal{MM}AS_R$ | − | − | − |

basic experiments. The dynamic behaviour of the algorithms in F-n135-k7 and the $m$ and $f$ values used on each iteration are presented in Figs. 9 and 10, respectively. The experimental setup and parameter settings of the algorithms were the same as in the basic experiments.

RIACO, EIACO and MIACO algorithms outperform their competitors in almost all DVRPs. This observation basically matches the results of the basic experiments. $AS_{rank}$-CVRP is outperformed in all test cases by its competitors. RIACO significantly outperforms EIACO and MIACO in F-n45-k4 whereas it is significantly outperformed in F-n72-k4. EIACO significantly outperforms RIACO and MIACO in F-n135-k7. From Fig. 9, it can be observed that EIACO usually maintains better performance in most environmental changes. In cases where the magnitude of change drops significantly, e.g., after iteration 300, EIACO is outperformed by $\mathcal{MM}AS_R$. This is natural because the knowledge transferred from elitism-based immigrants may not be compatible in the new environments and a complete restart of the algorithm is a better choice.

## 7. Conclusions

The DVRP has attracted less attention than the stationary VRP. In general, combinatorial optimization problems with dynamic environments have attracted less attention than other

DOPs in different domains. Existing benchmark generators for the DVRP generate DOPs where the optimum is unknown during the environmental changes, i.e., the DVRP with traffic factors. In this paper, we mainly use our recently proposed generator, i.e., the DBGP [40], which can generate DVRPs with known optimum over the environmental changes. In this way, one can observe how close to the optimum an algorithm performs. To address the DVRP, ACO algorithms with immigrants schemes (i.e., RIACO, EIACO and MIACO), which were recently developed for the dynamic travelling salesman problem [38], are re-designed specifically for the DVRP. The performance of the algorithm is compared against several existing peer ACO algorithms on different DVRP test cases.

From the experimental studies of benchmarking ACO algorithms on DVRPs with different dynamic properties, the following concluding remarks can be drawn. First, immigrants schemes enhance the performance of conventional ACO for DVRPs. Random immigrants are able to generate diversity whereas elitism- and memory-based immigrants are also able to transfer knowledge. Second, different pheromone strategies within ACO perform well on DVRPs with different properties. RIACO, EIACO and MIACO perform better on quickly, slowly and cyclically changing DVRPs, respectively. Third, the pheromone trails of previous environments are useful when the changing environments are similar; otherwise, a global re-initialization of the pheromone trails performs better, e.g., $\mathcal{MM}\text{AS}_R$. Fourth, ACO algorithms that are enhanced to address dynamic changes perform close to the optimum during the environmental changes, whereas conventional ACO algorithms, e.g., $\text{AS}_{rank}$-CVRP, perform far away from the optimum. Fifth, the internal parameters of the ACO algorithms, e.g., $r_i$ and $p_m^i$, affect the performance difference between RIACO, EIACO and MIACO. However, their effect is much less significant than the effect of the DOP properties, e.g., $f$ and $m$. Sixth, too much diversity may disturb the optimization process and destroy previous knowledge gained by algorithms, e.g., random immigrants. A good balance between the knowledge transferred and the diversity generated is vital to achieve good ACO performance in DVRPs. Finally, the DVRPs that change quickly and severely are more difficult to address than the DVRPs that change slowly and slightly. This is natural because the algorithm may not have enough time to re-optimize, or the changing environments may be completely different to transfer knowledge.

For future work, it would be interesting to adapt the parameters $r_i$ and $p_m^i$ of RIACO, EIACO and MIACO since they have a significant impact on their performance. Another future work is to integrate the dynamic time-linkage property to the DBGP where the solution obtained by the optimizer at time $t$ is dependent on at least one earlier solution [4, 43]. Finally, it would be interesting to consider other real-world problems, such as routing natural gas in buildings [49], optimal control of pumps in water distribution networks [33], manufacturing optimization problems [57, 58, 59, 60] and scheduling of trains and maintenance tracks [1].


## Acknowledgement

The authors would like to thank the anonymous reviewers for their thoughtful suggestions and constructive comments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the UK under Grant EP/K001310/1.


## References


[1] A. Albrecht, D. Panton, D. Lee, Rescheduling rail networks with maintenance disruptions using problem space search, Comput. Oper. Res. 40 (3) (2013) 703–712.

[2] D. Angus, T. Hendtlass, Ant colony optimization applied to dynamically changing problem, in: Developments in Applied Artificial Intelligence, vol. 2358 of LNAI, Springer-Verlag, 2002, pp. 618–627.

[3] E. Bonabeau, M. Dorigo, G. Theraulaz (eds.), Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, New York, 1997.

[4] P. A. N. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: Proc. of the 2005 Genetic and Evol. Comput. Conf., 2005, pp. 39–47.

[5] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proc. of the 1999 IEEE Congr. on Evol. Comput., vol. 3, 1999, pp. 1875–1882.

[6] B. Bullnheimer, R. Hartl, C. Strauss, Applying the ant system to the vehicle routing problem, in: S. Voß, S. Martello, I. Osman, C. Roucairol (eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic, 1997, pp. 285–296.

[7] B. Bullnheimer, R. Hartl, C. Strauss, An improved ant system algorithm for the vehicle routing problem, Annals of Oper. Res. 89 (1999) 319–328.

[8] B. Bullnheimer, R. Hartl, C. Strauss., A new rank-based version of the ant system: A computational study, Central Europ. J. for Oper. Res. and Econom. 7 (1) (1999) 25–38.

[9] C. Cruz, J. R. Gonzalez, D. A. Pelta, Optimization in dynamic environments: A survey on problems, methods and measures, Soft Comput. 15 (7) (2011) 1427–1448.

[10] G. B. Dantzig, J. H. Ramser, The truck dispatching problem, Management Sci. 6 (1) (1959) 80–91.

[11] H. Dawid, K. Doerner, R. Hartl, M. Reimann, Ant systems to solve operational problems, in: H. Dawid, K. Doerner, R. Hartl, M. Reimann, G. Dorffner, T. Fent, M. Feurstein, A. Mild, M. Natter, A. Taudes (eds.), Quantitative Models of Learning Organizations, vol. 3 of Interdisciplinary Studies in Economics and Management, Springer-Verlag, 2002, pp. 65–82.

[12] M. Dorigo, G. D. Caro, L. M. Gambardella, Ant algorithms for discrete optimization, Artif. Life 5 (2) (1999) 137–172.

[13] M. Dorigo, L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. on Evol. Comput. 1 (1) (1997) 53–66.

[14] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. on Syst., Man and Cybern. Part B: Cybern. 26 (1) (1996) 29–41.

[15] M. Dorigo, T. Stützle (eds.), Ant colony optimization, MIT Press, London, England, 2004.

[16] C. Eyckelhof, M. Snoek, Ant systems for a dynamic tsp, in: M. Dorigo, G. D. Caro, M. Sampels (eds.), Proceedings of the 3rd International Workshop on Ant Algorithms, vol. 2463 of LNCS, Springer-Verlag, 2002, pp. 88–99.

[17] M. Fisher, Optimal solution of vehicle routing problems using minimum k-trees, Oper. Res. 42 (4) (1994) 626–642.

[18] Y. Gajpal, P. Abad, Multi-ant colony system for a vehicle routing problem with backhauls, Europ. J. of Oper. Res. 196 (1) (2009) 102–117.

[19] Y. Gajpal, P. Abad, An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup, Comput. Oper. Res. 36 (12) (2009) 3215–3223.

[20] L. M. Gambardella, A. E. Rizzoli, F. Oliverio, N. Casagrande, A. Donati, R. Montemanni, E. Lucibello, Ant colony optimization for vehicle routing in advanced logistics systems, in: Proceedings of the International Workshop on Modelling and Applied Simulation, 2003, pp. 3–9.

[21] L. M. Gambardella, E. D. Taillard, C. Agazzi, Macs-vrptw: A multi-colony ant colony system for vehicle routing problems with time windows, in: New Ideas in Optimization, 1999, pp. 63–76.
[22] L. M. Gambardella, E. D. Taillard, M. Dorigo, Ant colonies for the quadratic assignment problem, J. of the Oper. Res. Soc. 50 (1999) 167–176.
[23] K. Goyal, P. Jain, M. Jain, Applying swarm intelligence to design the reconfigurable flow lines, Int. J. of Simulation Modelling 12 (1) (2013) 17–26.
[24] J. J. Grefenstette, Genetic algorithms for changing environments, in: Proc. of the 2nd Int. Conf. on Parallel Problem Solving From Nature, 1992, pp. 137–144.
[25] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic tsp, in: EvoWorkshops 2001: Applications of Evolutionary Computing, vol. 2037 of LNCS, Springer-Verlag, 2001, pp. 213–222.
[26] M. Guntsch, M. Middendorf, Applying population based aco to dynamic optimization problems, in: M. Dorigo, G. D. Caro, M. Sampels (eds.), Proceedings of the 3rd Int. Workshop on Ant Algorithms, vol. 2463 of LNCS, Springer-Verlag, 2002, pp. 111–122.
[27] M. Guntsch, M. Middendorf, H. Schmeck, An ant colony optimization approach to dynamic tsp, in: Proc. of the 2001 Genetic and Evol. Comput. Conf., 2001, pp. 860–867.
[28] J. Holland, Adaption in Natural and Artifical Systems, University of Michigan Press, Ann Arbor, 1975.
[29] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments - a survey, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 303–317.
[30] P. Kilby, P. Prosser, P. Shaw, Dynamic vrps: A study of scenarios, Tech. Rep. APES-06-1998, University of Strathclyde, U.K (1998).
[31] J. K. Lenstra, A. H. G. R. Kan, Complexity of vehicle and scheduling problems, Networks 11 (2) (1981) 221–227.
[32] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, P. N. Suganthan, Benchmark generator for cec'2009 competition on dynamic optimization, Tech. rep., Department of Computer Science, University of Leicester, U.K. (2009).
[33] M. Lopez-Ibanez, T. Prasad, B. Paechter, Ant colony optimization for optimal control of pumps in water distribution networks, J. of Water Resources Planning and Management 134 (4) (2008) 337–346.
[34] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, INFORMS J. on Computing 11 (4) (1999) 358–369.
[35] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem, in: EvoApplications2012: Applications of Evolutionary Computation, vol. 7248 of LNCS, Springer-Verlag, 2012, pp. 519–528.
[36] M. Mavrovouniotis, S. Yang, Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem, in: Proc. of the 2012 IEEE Congr. on Evol. Comput., 2012, pp. 2645–2652.
[37] M. Mavrovouniotis, S. Yang, Adapting the pheromone evaporation rate in dynamic routing problems, in: EvoApplications 2013: Applications of Evolutionary Computation, vol. 7835 of LNCS, Springer-Verlag, 2013, pp. 606–615.
[38] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Applied Soft Computing 13 (10) (2013) 4023–4037.
[39] M. Mavrovouniotis, S. Yang, Dynamic vehicle routing: A memetic ant colony optimization approach, in: A. Uyar, E. Ozcan, N. Urquhart (eds.), Automated Scheduling and Planning, chap. 9, Springer-Verlag, 2013, pp. 283–301.
[40] M. Mavrovouniotis, S. Yang, X. Yao, A benchmark generator for dynamic permutation-encoded problems, in: Proceedings of the 12th International Conference on Parallel Problem Solving from Nature, vol. 7492 of LNCS, Springer-Verlag, 2012, pp. 508–517.
[41] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, A. V. Donati, Ant colony system for a dynamic vehicle routing problem, Combinat. Optim. 10 (2005) 327–343.
[42] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm Evol. Comput. 6 (2012) 1–24.
[43] T. T. Nguyen, X. Yao, Dynamic time-linkage problems revisited, in: EvoWorkshops 2009: Applications of Evolutionary Computation, vol. 5484 of LNCS, Springer-Verlag, 2009, pp. 735–744.
[44] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, A review of dynamic vehicle routing problems, Europ. J. of Oper. Res. 225 (1) (2011) 1–11.
[45] M. Reimann, K. Doerner, F. Hartl, Analyzing a unified ant system for the vrp and some of its variants, in: EvoApplications2003: Applications of Evolutionary Computing, vol. 2611 of LNCS, Springer-Verlag, 2003, pp. 300–310.
[46] M. Reinmann, K. Doerner, R. Hartl, Insertion based ants for vehicle routing problems with backhauls and time windows, in: M. Dorigo, G. D. Caro, M. Sampels (eds.), Proc. of the 3rd Int. Workshop on Ant Algorithms, vol. 2463 of LNCS, Springer-Verlag, 2002, pp. 135–148.
[47] M. Reinmann, M. Stummer, K. Doerner, A saving based ant system for the vehicle routing problem, in: W. Langdon, E. Cantú, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, N. Jonaska (eds.), Proceedings of the 2002 Int. Conf. on Genetic and Evolutionary Computation, Morgan Kaufmann, 2002, pp. 1317–1325.
[48] A. E. Rizzoli, R. Montemanni, E. Lucibello, L. M. Gambardella, Ant colony optimization for real-world vehicle routing problems - from theory to applications, Swarm Intell. 1 (2) (2007) 135–151.
[49] J. A. Rodger, A fuzzy nearest neighbor neural network statistical model for predicting demand for natural gas and energy cost savings in public buildings, Expert Syst. with Appl. 41 (4, Part 2) (2014) 1813–1829.
[50] T. Stützle, H. Hoos, The max-min ant system and local search for the traveling salesman problem, in: Proc. of the 1997 IEEE Int. Conf. on Evol. Comput., 1997, pp. 309–314.
[51] G. Tao, Z. Michalewicz, Inver-over operator for the tsp, in: A. Eiben, T. Bck, M. Schoenauer, H.-P. Schwefel (eds.), Parallel Problem Solving from Nature PPSN V, vol. 1498 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 803–812.
[52] L. Xing, P. Rohlfshagen, Y. Chen, X. Yao, A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem, IEEE Trans. on Syst., Man and Cybern., Part B: Cybern. 41 (4) (2011) 1110–1123.
[53] S. Yang, Non-stationary problem optimization using the primal-dual genetic algorithm, in: Proc. of the 2003 IEEE Congr. on Evol. Comput., 2003, pp. 2246–2253.
[54] S. Yang, Genetic algorithms with elitism-based immigrants for changing optimization problems, in: M. Giacobini (ed.), Applications of Evolutionary Computing, vol. 4448 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 627–636.
[55] S. Yang, Genetic algorithms with memory- and elitism-based immigrants in dynamic environments, Evol. Comput. 16 (3) (2008) 385–416.
[56] S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Trans. on Evol. Comput. 12 (5) (2008) 542–561.
[57] A. R. Yildiz, A comparative study of population-based optimization algorithms for turning operations, Inform. Sci. 210 (2012) 81–88.
[58] A. R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach, Inform. Sci. 220 (2013) 399–407.
[59] A. R. Yildiz, K. Saitou, Topology synthesis of multi-component structural assemblies in continuum domains, Transactions of ASME, Journal of Mechanical Design 133 (1) (2011) 011008–1 011008–9.
[60] A. R. Yildiz, K. Solanki, Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach, The International Journal of Advanced Manufacturing Technology 59 (1-4) (2012) 367–376.
[61] X. Yu, K. Tang, T. Chen, X. Yao, Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization, Memetic Comput. 1 (1) (2009) 3–24.