Ant Colony Optimization with Memory-Based Immigrants for the Dynamic Vehicle Routing Problem

Michalis Mavrovouniotis
Department of Computer Science,
University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
Email: mm251@mcs.le.ac.uk

Shengxiang Yang
Department of Information Systems and Computing,
Brunel University, Uxbridge,
Middlesex UB8 3PH, United Kingdom
Email: shengxiang.yang@brunel.ac.uk

***Abstract*—A recent integration showed that ant colony optimization (ACO) algorithms with immigrants schemes perform well on different variations of the dynamic travelling salesman problem. In this paper, we address ACO for the dynamic vehicle routing problem (DVRP) with traffic factor where the changes occur in a cyclic pattern. In other words, previous environments will re-appear in the future. Memory-based immigrants are used with ACO in order to collect the best solutions from the environments and use them to generate diversity and transfer knowledge when a dynamic change occurs. The results show that the proposed algorithm, with an appropriate size of memory and immigrant replacement rate, outperforms other peer ACO algorithms on different DVRP test cases.**

I. Introduction

Ant colony optimization (ACO) algorithms consist of a population of ants that cooperate via their pheromone trails, where each ant deposits pheromone to its trails and the remaining ants can exploit it [5]. ACO algorithms have been applied and showed good performance on difficult optimization problems [4]. The environment of most of the problems addressed with ACO remains fixed during the execution of the algorithm. However, many real-world applications have dynamic environments, where the changing optimum needs to be tracked over time [19].

ACO algorithms can adapt to dynamic optimization problems (DOPs) since they are inspired from nature, which is a continuous adaptation process [11]. More precisely, they can adapt by transferring knowledge from past environments [1]. However, ACO algorithms face a serious challenge because of the stagnation behaviour, where all ants follow the same path from early stages of the algorithm. As a result, the algorithm loses the adaptation capability because high intensity of pheromone trails are generated to a single path, and it is difficult for the population of ants to adapt to the new environment when a dynamic change occurs. A simple way to address this challenge is to re-initialize the pheromone trails equally after the change and consider each environment as the arrival of a new problem instance. Usually, a global restart of the algorithm is not efficient and it is computationally expensive. It would only be a sufficient choice when the environment changes completely, and transferring knowledge from previous environments will not make sense [2].

Recently, developing strategies for ACO algorithms to deal with the premature convergence problem and address DOPs has attracted a lot of attention, which includes local and global restart strategies [9], memory-based approaches [8], pheromone manipulation schemes to maintain diversity [6], and immigrants schemes to increase diversity [14]–[16]. These approaches have been applied to the dynamic travelling salesman problem (DTSP), which is the simplest case of a DVRP, i.e., only one vehicle is used. Among them, immigrants schemes and memory-based approaches showed good performance in dynamic environments.

In this paper, we apply a hybridization of a memory-based approach and an immigrants scheme, called memory-based immigrants ACO (MIACO), to the dynamic VRP (DVRP) with traffic factor that changes in a cyclic pattern, which means that old environments are guaranteed to re-appear again in the future. The environmental changes are applied in such a way as to represent potential traffic jams over 24 hours. For example, during rush hours, the traffic factor is high, whereas during evening hours it is low. The aim of MIACO is to maintain the diversity via immigrants during the execution of the algorithm, and to transfer knowledge via memory to guide the population on environments that are previously visited.

The rest of the paper is organized as follows. Section II describes the DVRP with cyclic traffic factors. Section III describes the ant colony system (ACS), which is one of the best performing algorithms for the VRP. Section IV describes MIACO where immigrants and memory schemes are integrated to ACO. Section V describes the experiments carried out by comparing MIACO with other peer ACO algorithms. Finally, Section VI concludes this paper with directions for future work.

II. The DVRP with Cyclic Traffic Factor

The VRP is classified as $\mathcal{NP}$-hard [13]. The basic VRP can be described as follows: a number of vehicles with a fixed capacity need to satisfy the demand of all the customers,

starting from and finishing at the depot. There are many variations and extensions of the VRP, due to its similarities with many real-world application [18], such as the VRP with multiple depots, the VRP with service time, the VRP with pickup and delivery, the VRP with time windows, and so on.

Usually, the VRP is represented by a complete weighted graph $G = (V, E)$, with $n+1$ nodes, where $V = \{u_0, \ldots, u_n\}$ is a set of vertices corresponding to the customers (or delivery points) u_i $(i = 1, \cdots, n)$ and the depot u_0, and $E = \{(u_i, u_j) : i \neq j\}$ is a set of edges. Each edge (u_i, u_j) is associated with a non-negative d_{ij} which represents the distance (or travel time) between u_i and u_j. For each customer u_i, a non-negative demand q_i is given, whereas for the depot u_0, a zero demand is associated, i.e., $q_0 = 0$.

In this paper we consider the capacitated VRP where the aim is to find the route (or a set of routes) with the lowest cost without violating the following constraints:

1) every customer is visited exactly once by only one vehicle;
2) every vehicle starts and finishes at the depot;
3) the total demand of every route of a vehicle must not exceed the capacity Q of the vehicle.

The number of routes identifies the corresponding number of vehicles used to generate one VRP solution, which is not fixed but chosen by the algorithm.

From the stationary capacitated VRP described above we generate a dynamic variation, where the environmental changes are cyclic. A typical benchmark problem, is the DVRP with dynamic demands [12], [18]. However, in this paper we generate a DVRP with traffic factors, where each edge (u_i, u_j) is associated with a traffic factor t_{ij}. Therefore, the cost to travel from u_i to u_j is $c_{ij} = d_{ij} \times t_{ij}$. Note that the cost to travel from u_j to u_i may differ due to different traffic factor. For example, one road may have more traffic in one direction and no traffic on the opposite direction.

Every f iterations, a random number $R \in [F_L, F_U]$ 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 edge has a probability m to have a traffic factor, by generating a different R to represent high and low traffic jams on different roads, i.e., $t_{ij} = 1 + R$, where the remaining edges are set to $t_{ij} = 1$ (indicates no traffic). Note that f and m represent the frequency and magnitude of changes in the DVRP, respectively.

A cyclic environment can be constructed by generating different dynamic cases with traffic factors as the base states, representing DVRP environments with either low, normal, or high traffic. Then, the environment cycles among these base states in a fixed logical ring. Depending on the period of the day, environments with different traffic factors can be generated. For example, during the rush hour periods, a higher probability is given to generate R closer to F_U, whereas during evening hour periods, a higher probability is given to generate R closer to F_L.

III. ACO FOR THE DVRP

ACO algorithms consists of a population of μ ants where they construct solutions and share their information with each other via their pheromone trails. Ants "read" pheromone from others and "write" pheromone to their trails. The first ACO algorithm developed is the Ant System (AS) [5]. Many variations and extensions of the AS have been developed over the years and applied to different optimization problems [3], [4], [20].

The best performing ACO algorithm for the DVRP is the ACS [3]. There is a multi-colony variation of this algorithm applied to the VRP with time windows [7]. In this paper, we consider the single colony which has been applied to the DVRP [17]. Initially, all the ants are placed on the depot and all pheromone trails are initialized with an equal amount. With a probability $1-q_0$, where $q_0 (0 \leq q_0 \leq 1)$ is a parameter of the *pseudo-random* proportional decision rule (usually $q_0 = 0.9$ for ACS), an ant k chooses the next customer j from customer i, as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in N_i^k, \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{1}$$

where τ_{ij} is the existing pheromone trail between cities i and j, η_{ij} is the heuristic information available a priori, which is defined as $\eta_{ij} = 1/c_{ij}$, where c_{ij} is the distance travelled (including t_{ij}) between cities i and j, N_i^k denotes the neighbourhood of unvisited customers of ant k when its current customer is i, and α and β are the two parameters that determine the relative influence of pheromone trail and heuristic information, respectively. With the probability q_0, ant k chooses the next city, i.e., z, with the maximum probability, which satisfies the following formula:

$$z = \underset{j \in N_i^k}{\operatorname{argmax}} \left[\tau_{ij}\right]^\alpha \left[\eta_{ij}\right]^\beta. \tag{2}$$

However, if the choice of the next customer will lead to an infeasible solution, i.e., exceeding the maximum capacity of the vehicle, the depot is chosen and a new vehicle route starts.

When all ants construct their solutions, the best ant retraces the solution and deposits pheromone globally according to its solution quality on the corresponding trails, as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \Delta \tau_{ij}^{best}, \forall (i,j) \in T^{best}, \tag{3}$$

where $0 < \rho \leq 1$ is the pheromone evaporation rate and $\Delta \tau_{ij}^{best} = 1/C^{best}$, where C^{best} is the total cost of the best tour T^{best}. Moreover, a local pheromone update is performed every time an ant chooses another customer j from customer i as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \tau_0, \tag{4}$$

where ρ is defined as in Eq. (3) and τ_0 is the initial pheromone value.

The pheromone evaporation is the mechanism that helps the population to "forget" bad solutions constructed in previous environments by eliminating unnecessary pheromone trails and

Algorithm 1 MIACO and MEACO

```
initialize parameters
initialize ants P(0)
initialize pheromone trails τ_init
initialize k_long(0) with random ants
k_short(0) := empty, t_M := rand[5, 10] and t := 0
while termination condition not satisfied do
    for k := 1 to μ do
        construct VRP solution by ant k
        evaluate k
        update P(t)
    end for
    Add K_s best ants from P(t) into k_short(t)
    evaluate(k_long(t))
    if t := t_M || change detected then
        UpdateMemory(k_long(t)) using Algorithm 2
        t_M := t + rand[5, 10]
    end if
    if t := 0 then
        update pheromone using k_short(t)
    else
        if MIACO is selected then
            best_M := find the best in k_long(t)
            generate the set S_mi using best_M
            replace worst ants in k_short(t) with S_mi
        end if
        if MEACO is selected then
            update pheromone using k_long(t)
        end if
        update pheromone using k_short(t)
    end if
    t := t + 1
    k_long(t) := k_long(t − 1)
end while
```

adapt to the new environment. The recovery time depends on the size of the problem and magnitude of change.

IV. Memory-based Immigrants and Memory Enhanced ACO for the DVRP

A. Framework

The framework of the ACO algorithms enhanced with memory, i.e., MIACO and memory enhanced ACO (MEACO), as shown in Algorithm 1, is based on the ACO-based algorithms with immigrants used for the DTSP in previous work [14], [15]. It will be interesting to observe whether the framework based on immigrants schemes is beneficial for more realistic problems, such as the DVRP with cyclic traffic factors, as described in Section II. MEACO is identical with MIACO except that it does not generate immigrants, and it is considered in our experiments to investigate the effect of memory-based immigrants.

The initial phase of the algorithm and the solution construction of the ants are the same with the ACS (with $q_0 = 0.0$); see Eq. (1). The difference of the proposed framework is that it uses a short-term memory every iteration t, denoted as $k_{short}(t)$, of limited size K_s, and it is associated with the pheromone matrix. Initially, $k_{short}(0)$ is empty where at the end of the iteration the K_s best ants will be added to $k_{short}(t)$. Each ant k stored in $k_{short}(t)$ deposits a constant amount of pheromone to the corresponding trails, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{k}, \forall\ (i,j) \in T^{k}, \tag{5}$$

where $\Delta\tau_{ij}^{k} = (\tau_{max} - \tau_0)/K_s$ and T^k is the tour of ant k. Here, τ_{max} and τ_0 are the maximum and initial pheromone value, respectively.

Every iteration the ants from $k_{short}(t-1)$ are replaced with the K_s best ants from iteration t, and a negative update is performed to their pheromone trails, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} - \Delta\tau_{ij}^{k}, \forall\ (i,j) \in T^{k}, \tag{6}$$

where $\Delta\tau_{ij}$ and T^k are defined as in Eq. (5). This is because no ants can survive in more than one iteration due to the changing environment.

B. Memory Enhanced ACO (MEACO)

Apart from the $k_{short}(t)$ described above, MEACO has a long-term memory, denoted as $k_{long}(t)$, of limited size K_l, which is updated by replacing the closest ant in $k_{long}(t)$ with the best-so-far ant whenever there is a dynamic change. The metric to define how close ant p is to ant q is defined as:

$$M_{pq} = 1 - \frac{CE_{pq}}{n + avg(NV_p, NV_q)} \tag{7}$$

where CE_{pq} is defined as the number of common edges between the ants, n is the number of customers, and NV_p and NV_q are the number of vehicles ants p and q have in their solutions, respectively. A value M_{pq} closer to 0 means that the ants are similar. Every iteration t, the ants in $k_{long}(t)$ are re-evaluated in order to be valid with the new environment in case a dynamic change occurs. Moreover, an environmental change is detected if there is a change to the cost of the solutions stored in $k_{long}(t)$ at iteration $t+1$ when they are re-evaluated.

The update of $k_{long}(t)$ occurs whenever a dynamic change is detected. However, the update does not depend only on the detection of dynamic changes since in some real-world applications it is not easy or impossible to detect changes. For example, in dynamic environments, where noise is added in every iteration of the algorithm, it may also indicate environmental changes using the detection mechanism described above. As a result, the algorithm will not be able to distinguish whether the change of the fitness in a solution is because of noise or an environmental change, and the detection mechanism will not work properly. Note that this is not tested in the experiments, and deserves further investigation. Therefore, instead of updating $k_{long}(t)$ only in a fixed time interval, e.g., every f iterations, which is dependent on the dynamic changes, $k_{long}(t)$ is also updated in a dynamic pattern. For every update of $k_{long}(t)$, a random number $R \in [5, 10]$ is generated, which indicates the next update time; see Algorithm 2. For example,

Algorithm 2 UpdateMemory($k_{long}(t)$)

```
if t := t_M then
    best := find the iteration best ant in P(t)
end if
if change detected then
    best := find the best in k_short(t − 1)
    evaluate(best)
end if
if still any random ant in k_long(t) then
    replace a random ant in k_long(t) with best
else
    find the ant C_M in k_long(t) closest to best
    if best is better than C_M then
        C_M := best
    end if
end if
```

if the memory is updated at iteration t, the next update will occur at iteration $T_M := t + R$ [22], [23] (if no change is detected before iteration T_M).

For every iteration t, $k_{short}(t)$ is used to update the pheromone trails, and whenever a dynamic change is detected, $k_{long}(t)$ is merged with $k_{short}(t)$ to add additional pheromone and transfer knowledge for the ants on iteration $t + 1$ as presented in Algorithm 1. MEACO may perform well on slightly changing environments since the knowledge transfer may still be fit, and on cyclic environments since useful solutions from k_{long} are used to guide the population on re-appeared environments.

C. Memory-based Immigrants ACO (MIACO)

The main concern when dealing with immigrants schemes is how to generate immigrant ants, that represent feasible solutions. Traditional immigrants are generated randomly, i.e., random immigrants ACO (RIACO), which represent random VRP solutions. RIACO has been found to perform better in fast and significantly changing environments for the DTSP, since they enhance diversity [16]. However, there is a high risk of too much randomization with RIACO that may disturb the optimization process and degrade the performance.

Differently from RIACO, which generates diversity randomly with the immigrants, MIACO generates guided diversity by transferring knowledge using the best ant from $k_{long}(t)$ which is updated as in MEACO; see Algorithm 2. A memory-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 memory-based immigrants. The depots of the best ant are removed and adaptive inversion is performed based on the inver-over operator [10]. When the inversion operator finishes, the depots are added so that the capacity constraint is satisfied in order to represent one feasible VRP solution.

Considering the proposed framework above, on iteration t, the best ant from $k_{long}(t)$ is used as the base to generate a set S_{mi} of $r \times K_s$ immigrants, where r is the replacement rate. The memory-based immigrants replace the worst ants in $k_{short}(t)$ before the pheromone trails are updated as presented in Algorithm 1.

The MIACO algorithm has been found to perform better in slowly and slightly changing environments for the DTSP [14]. MIACO inherits the advantages of both MEACO to guide the population directly to an old environment already visited and RIACO to increase diversity. It is very important to store different solutions in $k_{long}(t)$ which represent different environments that might be useful in the future. This is achieved by the replacement strategy used in Eq. (7). The key idea behind MIACO is to provide guided diversity into the pheromone trails in order to avoid the disruption of the optimization process. However, there is a risk to transfer too much knowledge and start the optimization process from a local optimum and get stuck there.

V. Experimental Study

A. Experimental Setup

In the experiments, we compare the proposed MIACO and MEACO algorithms with RIACO and the traditional ACS, described in Section III. All the algorithms have been applied to the `vrp45`, `vrp72`, and `vrp135` problem instances[1].

To achieve a good balance between exploration and exploitation, most of the parameters have been optimized and obtained from our preliminary experiments where others have been inspired from literature [14], [15]. For all algorithms, $\mu = 50$ ants are used (expect on MIACO and MEACO where $\mu = 46$), $\alpha = 1$ and $\beta = 5$. For ACS, $q_0 = 0.9$, and $\rho = 0.7$. Note that a lower evaporation rate has been used for ACS, i.e. $\rho = 0.1$, with similar or worse results. For RIACO and MIACO, $q_0 = 0.0$, $K_s = 10$, $\tau_{max} = 1.0$ and $r = 0.4$. For MIACO and MEACO, $K_l = 4$.

For each algorithm on a DVRP instance, 30 independent runs were executed on the same cyclic environmental changes. The algorithms were executed for 1000 iterations and the overall offline performance is calculated as follows:

$$\vec{P}_{offline} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N}P_{ij}^{*}\right) \tag{8}$$

where N defines the number of runs, G defines the number of iterations for each run and P_{ij}^{*} defines the tour cost of the best ant since the last dynamic change of iteration i in run j [11]. Of course a lower value indicates a better performance.

The value of f was set to 10 and 100, which indicate fast and slowly changing environments, respectively. The value of m was set to 0.1, 0.25, 0.5, and 0.75, which indicate the degree of environmental changes from small, to medium, to large, respectively. The cyclic environment has 4 states and the bounds of the traffic factor are set as $F_L = 0$ and $F_U = 5$. As a result, eight dynamic environments, i.e., 2 values of $f \times 4$ values of m, were generated from each stationary VRP

[1]Taken from the Fisher benchmark instances available at http://neo.lcc.uma.es/radi-aeb/WebVRP/

TABLE I
EXPERIMENTAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF THE ALGORITHMS FOR DVRP WITH TRAFFIC FACTOR

Alg. & Inst.	vrp45				vrp72				vrp135			
$f = 10$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACS	895.5	925.1	1179.3	1542.2	307.6	338.6	396.1	447.3	1409.6	1541.4	1767.8	2494.9
RIACO	851.4	864.8	1053.9	1363.4	295.5	318.0	380.7	426.1	1409.8	1541.0	1757.0	2434.9
MEACO	855.7	863.5	1060.5	1377.5	294.3	319.8	379.6	426.7	1413.4	1535.4	1752.1	2452.4
MIACO	853.3	862.5	1052.6	1361.5	291.9	315.4	379.2	425.0	1394.1	1532.2	1748.3	2433.0
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACS	861.8	886.3	1067.7	1394.3	301.4	325.6	384.0	425.5	1366.2	1484.1	1686.7	2281.9
RIACO	829.6	833.5	975.2	1214.8	282.2	297.2	354.6	390.4	1354.6	1437.7	1623.6	2173.2
MEACO	836.5	834.3	987.4	1241.6	282.9	298.7	351.8	391.7	1353.8	1437.0	1623.2	2178.0
MIACO	833.3	830.4	972.0	1213.1	277.6	288.7	349.1	387.7	1326.2	1425.5	1601.4	2121.2

TABLE II
STATISTICAL TEST RESULTS REGARDING THE OFFLINE PERFORMANCE OF THE ALGORITHMS FOR DVRP WITH TRAFFIC FACTOR

Alg. & Inst.	vrp45				vrp72				vrp135			
$f = 10$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACS⇔MEACO	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	$-$	$-$
RIACO⇔ACS	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$\sim$	$+$	$+$
RIACO⇔MEACO	$+$	$-$	$+$	$+$	$-$	$+$	$-$	$\sim$	$+$	$+$	$-$	$+$
MIACO⇔ACS	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
MIACO⇔MEACO	$+$	$+$	$+$	$+$	$+$	$+$	$\sim$	$+$	$+$	$+$	$+$	$+$
MIACO⇔RIACO	$-$	$+$	$\sim$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$\sim$
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACS⇔MEACO	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
RIACO⇔ACS	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
RIACO⇔MEACO	$+$	$+$	$+$	$+$	$\sim$	$\sim$	$-$	$+$	$\sim$	$\sim$	$\sim$	$+$
MIACO⇔ACS	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
MIACO⇔MEACO	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
MIACO⇔RIACO	$-$	$+$	$+$	$\sim$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$

instance, as described in Section II, to systematically analyze the adaptation and searching capability of each algorithm on the DVRP.

B. Experimental Results and Analysis

The experimental results regarding the offline performance of the algorithms are given in Table I and the corresponding statistical results of Wilcoxon rank-sum test, at the 0.05 level of significance, are presented in Table II, where the symbol "$+$" or "$-$" indicates that the first algorithm is significantly better than or significantly worse than the second one, respectively, and "$\sim$" indicates no significant difference between two algorithms. Moreover, to better understand the dynamic behaviour of algorithms, the results of slow and fast changing environments, are plotted in Figs. 1 and 2, with $m = 0.1$ and $m = 0.75$ for the first 1000 and 500 iterations, respectively. From the experimental results, several observations can be made by comparing the behaviour of algorithms.

First, RIACO outperforms ACS in all the dynamic test cases; see the results of RIACO $\Leftrightarrow$ ACS in Table II. This validates our expectation that ACS need sufficient time to recover when a dynamic change occurs, which can be also observed from Fig. 1. This is because the pheromone evaporation is the only mechanism used to eliminate pheromone trails that are not useful to the new environment, and may bias the population to areas that are not near the new optimum. On the other hand, RIACO uses the proposed framework where the pheromone trails exist only in one iteration and are regenerated in the next iteration.

Second, MEACO outperforms ACS in all dynamic test cases; see the results of MEACO $\Leftrightarrow$ ACS in Table II. This is due to the knowledge transferred using the solutions from $k_{long}(t)$ when the environment changes. However, MEACO is outperformed by RIACO in most fast changing environments $f = 10$ and significantly changing environments $m = 0.75$ (even when $f = 100$); see the results of RIACO $\Leftrightarrow$ MEACO in Table II. The knowledge transferred may not be useful when the environments are not similar or when the available time is not enough to store useful solutions in $k_{long}(t)$.

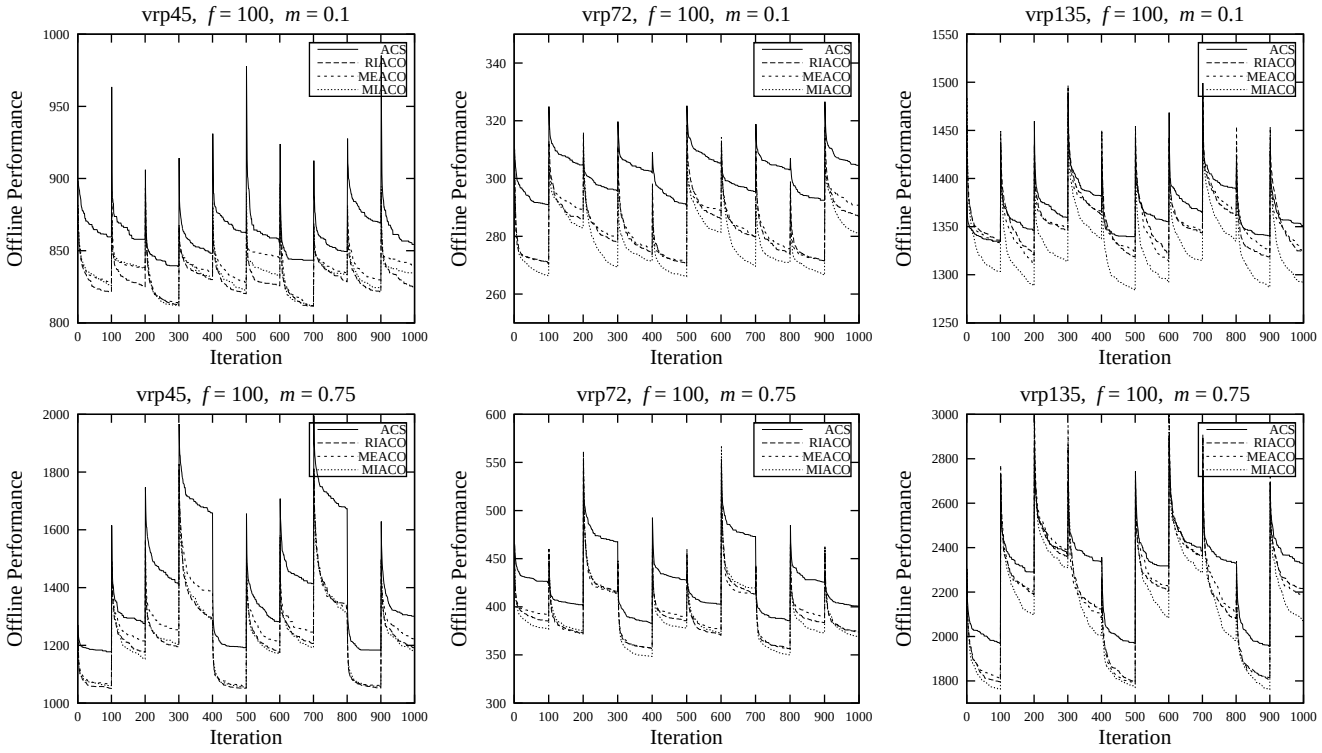

Fig. 1. Dynamic behaviour of ACO algorithms on slow changing environment

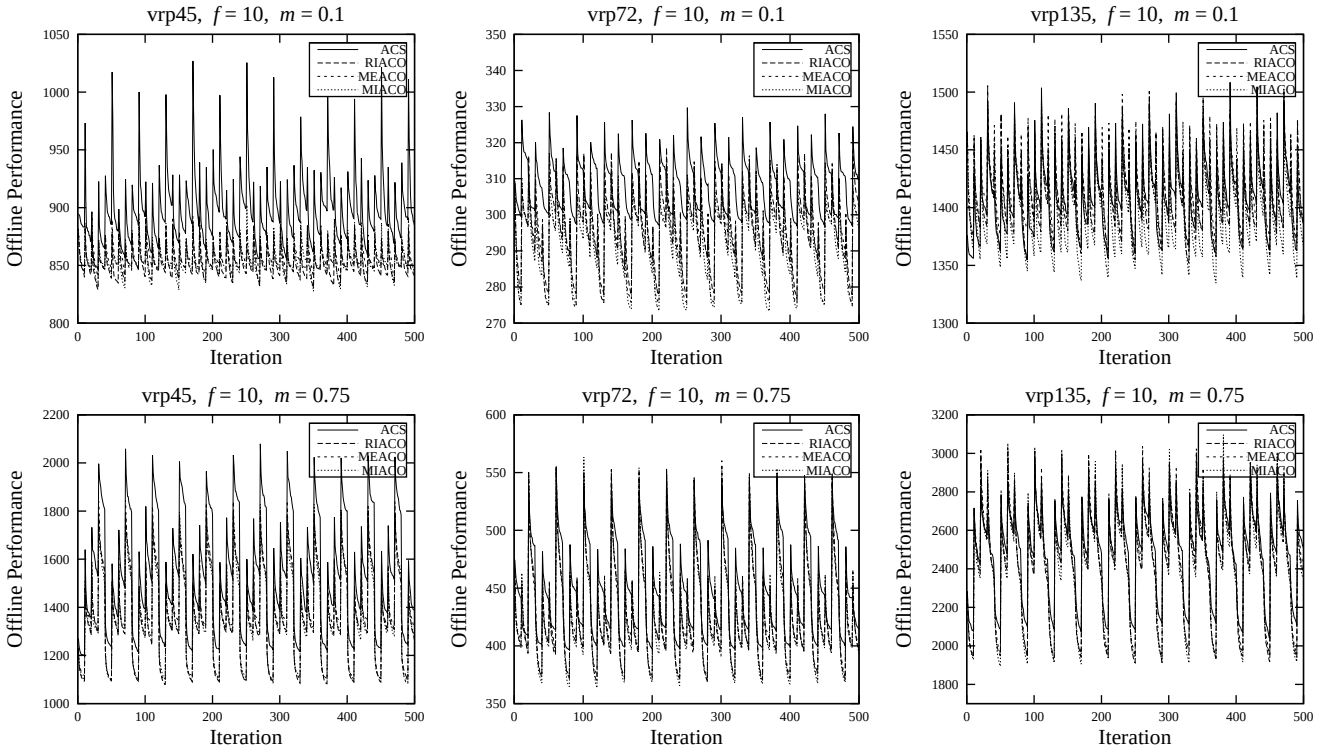

Fig. 2. Dynamic behaviour of ACO algorithms on fast changing environment

Third, MIACO outperforms ACS in all dynamic test cases as RIACO; see the results MIACO $\Leftrightarrow$ ACS in Table II. This is due to the same reasons RIACO outperforms the traditional ACS. Furthermore, MIACO outperforms RIACO and MEACO

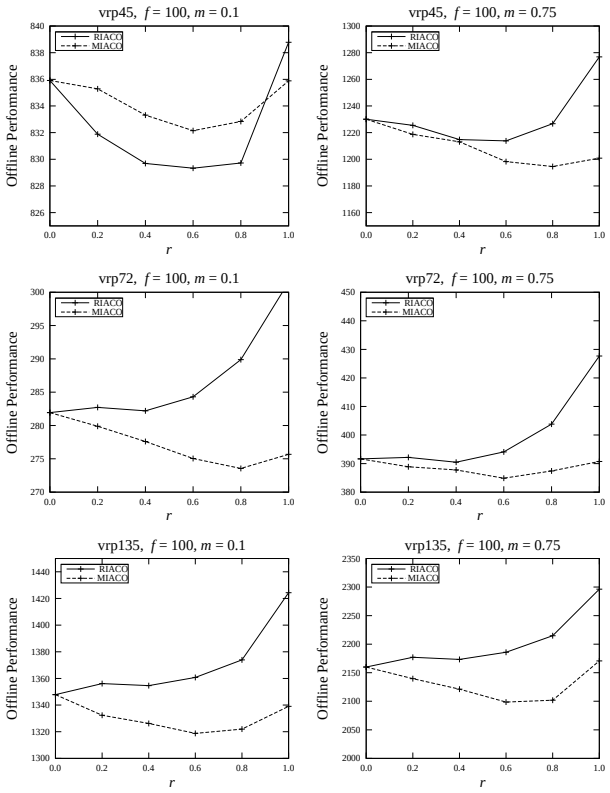

Fig. 3. Offline performance of RIACO and MIACO with different replacement rates r.

in almost all dynamic test cases; see the results of MIACO $\Leftrightarrow$ RIACO and MIACO $\Leftrightarrow$ MEACO in Table II. MIACO is able to guide the population of ants to old environments that will reappear again in the future using solutions from k_{long}. To some extent, this validates our expectation that MIACO inherits the merits from both memory and immigrants schemes, but we furthermore investigate their effect.

C. Experimental Results Regarding the Replacement Rate r

In order to investigate the effectiveness of the immigrants schemes, further experiments have been performed on the same problem instances with the same parameters used before but with different replacement rates, i.e., $r \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. In Fig. 3, the offline performance of RIACO and MIACO with the varying replacement rate for $f = 100$ and $m = 0.1$ and $m = 0.75$ for all problem instances are presented. The experimental results of the remaining dynamic test cases are similar, where $r = 0.0$ means that no immigrants are generated to replace ants in k_{short}.

From the results, it can be observed that memory-based immigrants in MIACO improve the performance and validate our expectation; see the results in Fig. 3 from $r = 0.0$ and $r > 0.0$. On the other hand, random immigrants in RIACO improve the performance on some problem instances and some dynamic test cases, whereas they degrade the performance on the remaining ones. This is because the random immigrants may disturb the optimization process of ACO because of too much randomization on the pheromone trails.

Generally, the performance of RIACO is inconsistent since on `vrp45` the performance is improved, whereas on `vrp72` and `vrp135` it is degraded. On the other hand, the performance of MIACO is consistent since the performance is always improved, even if all ants in $k_{short}(t)$ are replaced by immigrant ants. The optimization is not disturbed as in RIACO, since MIACO with $r = 1.0$ may destroy the knowledge gained in $k_{short}(t)$ but generate new knowledge using the best ant from $k_{long}(t)$.

D. Experimental Results Regarding the Memory Size K_l

In order to investigate the effectiveness of the memory scheme, further experiments have been performed on the same problem instances with the same parameters used before but with different memory sizes K_l, i.e., $K_l \in \{0, 2, 4, 6\}$. The population size μ was set according to K_l in order to have the same number of evaluations. For example, when $K_l = 0$ the number of ants is set to $\mu = 50$ normally, whereas when $K_l > 0$ the number of ants is to $\mu = 50 - K_l$. In Fig. 4 the offline performance of MIACO and MEACO with the varying K_l for $f = 100$ on all problem instances are presented. The experimental results of the remaining test cases showed similar behaviour.

From the results, it is can be observed that the memory in MIACO improves the performance which validates our expectation; see the results in Fig. 4 from $K_l = 0$ to $K_l = 2$ and $K_l = 4$. On the other hand, the memory in MEACO slightly improves the performance on some problem instances and some dynamic test cases, whereas it degrades the performance on the remaining ones. This may be due to the use of $k_{long}(t)$, in which MIACO uses the best ant to generate memory-based immigrants to replace the worst ones in $k_{short}(t)$, MEACO merges $k_{short}(t)$ and $k_{long}(t)$, and then pheromone trails are updated. The knowledge transferred from MIACO is more direct since a solution from one environment is used, whereas from MEACO it is general since all the from various environments are used.

Generally, the performance of MEACO is inconsistent whereas for MIACO it is consistent since it always improves. It is similar with the effect found previously for the immigrants schemes. From the experiments presented in Fig. 4 and on the remaining ones, it is observed that increasing K_l usually degrades the performance since the actual population is decreased for the sake of achieving the same number of evaluations. As a result, the search capability of ACO is degraded because fewer ants explore the search space. In fact, Branke [2] observed that a standalone memory may not be enough to address DOPs, and, it should be combined with a diversity method.

VI. Conclusions and Future Work

Memory-based immigrants have been successfully applied to evolutionary algorithms (EAs) to address different binary-encoded DOPs [21], and to ACO to address DTSPs [15]. In this paper, we apply MIACO to address the DVRP with traffic factor under cyclic environmental changes, which is

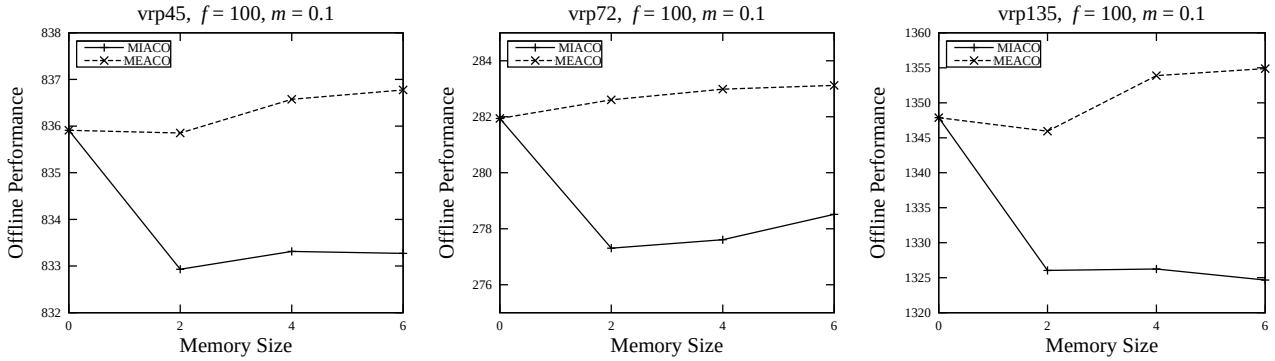

Fig. 4. Offline performance of MEACO and MIACO with different memory sizes K_l.

a DOP closer to a real-world application. It combines the merits of memory and immigrants schemes, where the first one is able to guide the population into previously visited environment directly, and the second one is able to increase and maintain diversity within the population to adapt well in DOPs. The immigrant ants are generated using the best ant from the memory as the base and replace the worst ones in the population.

Comparing MIACO with other peer ACO algorithms on different test cases of DVRPs, the following concluding remarks can be drawn. First, memory-based immigrants perform well in dynamic environments under cyclic changes. Second, random immigrants and memory usually improves the performance of ACO for DVRPs, but they have inconsistent behaviour. Finally, MIACO outperforms other peer ACO algorithms in almost all dynamic test cases, and shows that its behaviour is consistent.

For future work, it would be interesting to apply MIACO in DVRP with traffic factor where the environment changes randomly and not cyclically, and compare it with other ACO-based algorithms with immigrants schemes [14]. The knowledge transferred by memory-based immigrants from previous environments to the pheromone trails of the new one may be useful, even if old environments are not guaranteed to reappear, when the changing environments are similar. Another future work is to investigate the performance of MIACO in dynamic environments with noise where the environmental changes are undetectable.

REFERENCES

[1] E. Bonabeau, M. Dorigo and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

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

[3] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the travelling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1 no. 1, pp. 53–66, 1997

[4] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, London, 2004.

[5] M. Dorigo, V. Maniezzo and A. Colorni. "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. on Syst., Man and Cybern., Part B: Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[6] C. J. Eyckelhof and M. Snoek. "Ant Systems for a Dynamic TSP," in *Proc. of the 3rd Int. Workshop on Ant Algorithms*, 2002, pp. 88–99.

[7] L. M. Gambardella, E. Taillard and G. Agazzi. "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*, D. Corne et al., Eds., 1999, pp. 63–76.

[8] M. Guntsch and M. Middendorf. "Applying population based ACO to dynamic optimization problems," in *Proc. of the 3rd Int. Workshop on Ant Algorithms*, vol. 2463, 2002, pp. 111–122.

[9] M. Guntsch and M. Middendorf. "Pheromone modification strategies for ant algorithms applied to dynamic TSP," in *EvoApplications 2001: Appl. of Evol. Comput*, LNCS 2037, 2001, pp. 213–222.

[10] T. Guo and Z. Michalewicz. "Inver-over operator for the TSP," in *Proc. of the 5th Int. Conf. on Parallel Problem Solving from Nature*, 1998, pp. 803–812.

[11] Y. Jin and J. Branke. "Evolutionary optimization in uncertain environments - a survey," *IEEE Trans. on Evol. Comput.*, vol. 9, no. 3, pp. 303–317, 2005.

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

[13] M. Labbe, G. Laporte and H. Mercure. "Capacitated vehicle routing on trees," *Oper. Res.*, vol. 39, no. 4, pp. 616–622, 1991.

[14] M. Mavrovouniotis and S. Yang. "Ant colony optimization with immigrants schemes for dynamic environments," in *Proc. of the 11th Int. Conf. on Parallel Problem Solving from Nature*, LNCS 6239, 2010, pp. 371–380.

[15] M. Mavrovouniotis and S. Yang. "Memory-based immigrants for ant colony optimization in changing environments," in *EvoApplications 2011: Appl. of Evol. Comput.*, LNCS 6624, 2011, pp. 324–333.

[16] M. Mavrovouniotis and S. Yang. "An immigrants scheme based on environmental information for ant colony optimization for the dynamic travelling salesman problem," *Proc. of the 10th Int. Conf. on Artificial Evolution* (EA-2011), 2001, pp. 23–34.

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

[18] H. Psaraftis. "Dynamic vehicle routing: status and prospects," *Annals of Oper. Res.*, vol. 61, pp. 143–164, 1995.

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

[20] T. Stützle and H. Hoos, "The MAX-MIN ant system and local search for the traveling salesman problem," in *Proc. 1997 IEEE Int. Conf. on Evol. Comput.*, 1997, pp. 309–314.

[21] S. Yang. "Genetic algorithms with memory and elitism based immigrants in dynamic environments," *Evol. Comput.*, vol. 16, no. 3, pp. 385–416, 2008.

[22] S. Yang and X. Yao, "Population-based incremental learning with associative memory for dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 542–561, 2008.

[23] S. Yang, H. Cheng and F. Wang, "Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks," *IEEE Trans. Syst., Man, and Cybern. Part C: Appl. and Rev.*, vol. 40, no. 1, pp. 52-63, 2010.