Interactive and Non-Interactive Hybrid Immigrants Schemes for Ant Algorithms in Dynamic Environments

Michalis Mavrovouniotis
Centre for Computational Intelligence (CCI),
School of Computer Science and Informatics,
De Montfort University, The Gateway,
Leicester, LE1 9BH, U.K.
Email: mmavrovouniotis@dmu.ac.uk

Shengxiang Yang
Centre for Computational Intelligence (CCI),
School of Computer Science and Informatics,
De Montfort University, The Gateway,
Leicester, LE1 9BH, U.K.
Email: syang@dmu.ac.uk

***Abstract*—Dynamic optimization problems (DOPs) have been a major challenge for ant colony optimization (ACO) algorithms. The integration of ACO algorithms with immigrants schemes showed promising results on different DOPs. Each type of immigrants scheme aims to address a DOP with specific characteristics. For example, random and elitism-based immigrants perform well on severely and slightly changing environments, respectively. In this paper, two hybrid immigrants, i.e., non-interactive and interactive, schemes are proposed to combine the merits of the aforementioned immigrants schemes. The experiments on a series of dynamic travelling salesman problems showed that the hybridization of immigrants further improves the performance of ACO algorithms.**

I. Introduction

Ant colony optimization (ACO) algorithms are inspired from nature, i.e., the foraging behaviour of real ant colonies [2], [3]. ACO has been initially developed for the travelling salesman problem (TSP) and later on has been applied on several other applications [4], [7], [17]. Most of the optimization problems addressed so far by ACO assume a stationary environment. However, the environment in many real-world applications changes over time. The difference between stationary and dynamic optimization problems is that the aim of the former problems is to locate the static global optimum efficiently whereas the aim of the latter problems is to track the moving global optimum efficiently [10].

Addressing dynamic optimization problems (DOPs) is a challenging task to ACO algorithms, and generally to all optimization algorithms. Once the algorithm converges to an optimum, then it is difficult to escape from it in order to track the new optimum when a dynamic change occurs. The pheromone trails, generated with ACO algorithms, of the previous environment may bias the population of ants into the optimum of the previous environment. A straightforward way to address this issue is to consider every dynamic change as the arrival of a new problem instance that needs to be solved from scratch by re-initializing all the pheromone trails with an equal amount. However, such strategy may be computational expensive and requires the detection of a dynamic change. In case the changing environments have similarities, the re-optimization time may be improved by transferring knowledge from previous environments [1], [10], [12].

Over the years, several strategies have been proposed to enhance the performance of ACO algorithms in DOPs including, maintain diversity strategies [5], [8], [12], increase diversity via immigrants [11], [13], [14], memory-based schemes [9] multi-colony schemes [16]. Among the approaches immigrants schemes have proved that they are straightforward to apply and have good performance on difference DOPs. The policy of immigrants schemes in ACO is to replace a small portion of ants from the population with other randomly generated ants in order to deposit pheromone [11]. Another type of immigrants schemes, is when elitism-based immigrants are generated using the best ant from the previous environment as the base to generate ants and replace other ants in the population [14].

Random and elitism-based immigrants have shown good performance on DOPs with severely and slightly changing environments, respectively [13], [14]. Since different immigrants schemes are suitable for different DOPs, then hybrid immigrants may combine the merits of two or more immigrants schemes [18], [19]. In this paper, we propose two types of hybrid immigrants. The first one generates both types of immigrants on each iteration, whereas the second one each type of immigrant is triggered depending on the current of status of the algorithm.

Based on the dynamic benchmark generator proposed in [15], a series of dynamic test cases are constructed from several stationary TSP instances and experiments are systematically carried out for several ACO algorithms, including the proposed ones. The experiments show that the hybrid immigrants further improves the performance of ACO algorithms in many dynamic test cases. The rest of the paper is organized as follows. Section II describes the dynamic TSP generated by the benchmark generator. Section III reviews existing integrations of immigrants schemes with ACO. Section IV describes the proposed hybrid immigrants schemes and section V gives the experimental results. Section VI concludes this paper with discussion on future work.

II. Dynamic Travelling Salesman Problem

A. Description of the Problem

The TSP can be described as follows: given a collection of cities, the objective is to find the shortest path that starts from

one city and visits each of the other cities once before returning to the starting city. The TSP is classified to the $\mathcal{NP}$-complete combinatorial optimization problems. Usually, the problem is represented by a fully connected weighted graph $G = (N, A)$, where $N = \{1, \ldots, n\}$ is a set of nodes and $A = \{(i, j) : i \neq j\}$ is a set of arcs. The collection of cities is represented by the set N and the connections between them by the set A. Each connection (i, j) is associated with a non-negative value d_{ij} which represents the distance between cities i and j.

Formally, the TSP can be described as follows:

$$f(x) = min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \psi_{ij}, \tag{1}$$

subject to:

$$\psi_{ij} = \begin{cases} 1, & \text{if } (i, j) \text{ is used in the tour,} \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\psi_{ij} \in \{0, 1\}$, n is the number of cities, and d_{ij} is the distance between cities i and j.

B. Generate Dynamic Test Environments

In order to generate dynamic TSPs (DTSPs), the dynamic benchmark generator for permutation-encoded problems (DBGP) [15] is used, which can convert any static permutation-encoded benchmark problem instance to a dynamic environment. The main advantage of DBGP compared to other benchmark generators is that in case the optimum of the benchmark problem instance is known, then 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 DTSP benchmark generators, e.g., the DTSP with traffic factors [14] and the DTSP with exchangeable cities [9], modify the fitness landscape and the optimum value changes in every dynamic change.

Considering the TSP description, each city $i \in N$ has a location defined by (x, y) and each arc $(i, j) \in A$ 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 $m \times n$ cities, 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 $\vec{V}(T)$ cities are swapped. Then a randomly re-ordered vector $\vec{U}(T)$ is generated that contains the cities 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.

III. ACO in Dynamic Environments

Conventional ACO algorithms consist of a population of μ ants where they construct solutions and share their information via their pheromone trails. ACO was initially established for the TSP [3]. More precisely, every ant k is able to construct a complete TSP solution by using a probabilistic rule to move from city i to city j as follows:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta}}, \text{if } j \in \mathcal{N}_i^k, \tag{3}$$

where τ_{ij} is the existing pheromone trail between cities i and j, $\eta_{ij} = 1/d_{ij}$ is the heuristic information available a priori, where d_{ij} is the distance between cities i and j and $\mathcal{N}_i^k$ denotes the set of unvisited cities for ant k incident to city i.

The pheromone trails are updated after all ants construct solutions, where each ant deposits pheromone according to their solution quality. In this way the next solution construction is more likely to guide ants to promising areas because of the distribution of the pheromone trails. Additionally, pheromone evaporation is performed where small amount of pheromone is deducted from all trails.

This process continuous until the algorithm eventually converges. For the DTSP, premature convergence is often dangerous because when a dynamic change occurs it is difficult for the population to escape from the old optimum in order to adapt to the newly generated one. This is due to the high concentration of pheromone trails around the old optimum that bias the ants to still construct solutions for the previous environment rather than exploring for the new one. The pheromone evaporation may be able to eliminate the high intensity of pheromone trails in case they are not frequently used from the ants.

Several other strategies were proposed to eliminate the high intensity of pheromone trails in ACO and enhance their performance in different DOPs such as adaptive pheromone evaporation [12], immigrant schemes [14], population-based ACO [9], pheromone re-initialization strategies [8] and multi caste ACO [16]. The integration of ACO with immigrants schemes proved that enhances the performance on different DOPs such as the DTSP with traffic factors [14], [16] and the DTSP with exchangeable cities [9], [11]. The basic principle of immigrants schemes in ACO is to introduce new ants (immigrants) every iteration to replace other ants from the constructed population and, thus, distribute the pheromone trails into several areas in the search space rather than to a single area (usually near the global optimum).

IV. Hybrid Immigrants Schemes

A. Algorithms' Framework

Immigrants schemes mainly differ on the way immigrant ants are generated. Random immigrants ACO (RIACO) and elitism-based immigrants ACO (EIACO) were proposed for DTSPs [14]. Random immigrants are generated randomly to represent random TSP tours, whereas elitism-based immigrants are generated by swapping cities from the best solution of the previous environment.

The solution construction of RIACO and EIACO is defined as in Eq. (3). However, the pheromone update of RIACO and EIACO differs from conventional ACO algorithms in the following ways: a) only the k_s best ants are allowed to deposit pheromone in addition with immigrant ants; b) pheromone evaporation is not available; and c) the pheromone trails exist only for a single iteration.

Every t iteration, the pheromone table is associated with a short-term memory of k_s size, denoted $k_{short}(t)$ in this paper, and any change to $k_{short}(t+1)$ causes an update to the pheromone table. Within RIACO and EIACO the k_s best ants from the current iteration t replace the ants in $k_{short}(t-1)$. Furthermore, for RIACO a set S_{ri} of n_{ri} random immigrants are generated to replace the worst ants in $k_{short}(t)$. For EIACO, the best ant from the previous environment is used as the base to generate a set S_{ei} of n_{ei} elitism-based immigrants to replace the worst ants in $k_{short}(t)$. For the DTSP an elitism-based immigrant is generate by swapping cities with a p_m^i probability (usually $p_m^i = 0.01$).

When the worst ants are replaced by immigrant ants, the pheromone trails of each k-th worst ant are removed, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} - \Delta\tau_{ij}^k, \forall\ (i,j) \in T^k, \tag{4}$$

where T^k represents the tour of ant k and $\Delta\tau_{ij}^k = (\tau_{max} - \tau_0)/k_s$, where τ_{max} and τ_0 denote the maximum and initial pheromone values, respectively. Furthermore, the pheromone trails of each k-th immigrant ant are added, as follows:

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

where $\Delta\tau_{ij}^k$ and T^k are as defined in Eq. (4).

B. Non-Interactive Hybrid Immigrants

RIACO and EIACO were proposed to address severely and slightly changing DTSPs, respectively. This is natural because in case the environment changes slightly EIACO is more suitable due to the knowledge transferred via elitism-based immigrants. On the other hand, RIACO is suitable in more severely changing environments.

However, there is a high risk with RIACO to disturb the optimization process of ACO because of too much randomization and with EIACO to trap the optimization process of ACO into a local optimum because of too much knowledge transferred. In order to limit both risks and further improve the performance of ACO algorithms in DOPs, RIACO are hybridized together with EIACO, to form hybrid immigrants ACO (HIACO-I). In [11] and [13] hybrid immigrants showed good offline performance when applied on the DTSP with exchangeable cities and with traffic factors, respectively. In this paper HIACO-I is applied on the DTSPs constructed from the DBGP.

Every iteration t within HIACO-I a set $S_{hi}(t)$ of n_{ri} random immigrants and n_{ei} elitism-based immigrants are generated. In this way, the half of the generated immigrants will be random immigrants and the other half elitism-based immigrants. HIACO-I follows the solution construction defined in Eq. (3) and the same pheromone policy with RIACO and EIACO defined in Eqs 4 and 5.

C. Interactive Hybrid Immigrants

Within HIACO-I, the two types of immigrants, random and elitism-based, are generated without the consideration of the status of algorithm. For example, when the ACO algorithm reaches stagnation behaviour[1] or when too much knowledge in transferred, random immigrants may be more suitable than elitism-based immigrants, since more diversity is needed to help escape from local optima. Therefore, another type of hybrid immigrants in which random and elitism-based immigrants are triggered interactively can be proposed to combine the merits of RIACO and EIACO, denoted HIACO-II in this paper. The basic idea is to generate only elitism-based immigrants every iteration and when the algorithm reaches stagnation behaviour to generate only random immigrants.

The λ-branching factor [6] is an efficient way to define whether ACO has reached stagnation behaviour. The idea of λ-branching is described as follows: If for a given city $i \in V$, the concentration of pheromone trails on almost all the incident arcs becomes very small but is large for a few others, then the freedom of exploring other paths from city i is very limited. Therefore, if this situation arise simultaneously for all cities of graph G, the search space that is searched by ants becomes relatively small.

The average $\bar{\lambda}(t)$ branching factor at iteration t is defined as follows:

$$\bar{\lambda}(t) = \frac{1}{2n}\sum_{i=1}^{n}\lambda^i, \tag{6}$$

where n is the number of cities and λ^i is the λ-branching factor for city i, which is defined as follows:

$$\lambda^i = \sum_{j=1}^{d} L_{ij}, \tag{7}$$

where d is the number of available arcs incident to city i and L_{ij} is defined as follows:

$$L_{ij} = \begin{cases} 1, & \text{if } (\tau_{min}^i + \lambda(\tau_{max}^i - \tau_{min}^i)) \leq \tau_{ij}, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where λ is a constant parameter ($\lambda = 0.05$ by default [6]), τ_{min}^i and τ_{max}^i are the minimum and maximum pheromone trail values on the arcs incident to city i, respectively. A value of $\bar{\lambda}(t)$ close to 1 indicates stagnation behaviour.

According to the behaviour of the algorithm in terms of searching and the rules described above, either random or elitism-based immigrants are triggered as follows:

$$S_{hi}(t) = \begin{cases} S_{ei}, & \text{if } \bar{\lambda}(t) > \epsilon, \\ S_{ri}, & \text{otherwise,} \end{cases} \tag{9}$$

where $\bar{\lambda}(t)$ is defined in Eq. (6) that indicates the status of the algorithm and ϵ is a threshold value. When $\epsilon = 1$, which is the default value, indicates stagnation behaviour whereas a higher value $\epsilon > 1$ indicates exploration. Hence according to the rules defined in Eq. (9) whenever HIACO-II reaches stagnation behaviour random immigrants are triggered to move the population into different areas in the search space.

V. Experimental Study

A. Experimental Setup

In the experiments, we investigate the performance of HIACO-I and HIACO-II compared with standard ACO

[1] A term use to define that behaviour when all ants construct the same solution from early stages

TABLE I. EXPERIMENTAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF ACO ALGORITHMS

Algorithms & DTSPs	kroA100(Optimum=21282)				kroA150(Optimum=26524)				kroA200(Optimum=29368)			
$f = 10$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
SACO	25103.7	26944.7	28749.8	29397.2	31977.0	34683.4	37484.0	38419.8	36939.2	39901.5	43211.3	44379.3
RIACO	24327.5	25843.7	26600.2	26680.7	30871.2	32544.6	33267.3	33414.2	34854.8	36746.6	37446.0	37538.6
EIACO	22992.1	24696.1	25388.5	25440.9	29426.7	31220.2	31824.9	31967.9	33170.7	35228.0	35853.7	35938.9
HIACO-I	23588.1	25395.7	26139.3	26209.6	30222.2	32069.3	32723.2	32874.0	34187.7	36211.0	36894.8	36949.8
HIACO-II	23260.9	24812.1	25144.6	25292.9	29583.8	31350.8	31570.8	31714.6	33192.2	35238.3	35648.0	35719.8
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
SACO	23347.9	23956.2	24187.8	24286.0	29737.6	30422.8	30931.8	30973.5	33895.2	34508.6	35103.3	35195.1
RIACO	22002.0	22465.4	22779.2	22777.8	28015.7	28441.1	28703.2	28689.7	30782.4	31557.3	32057.9	32028.0
EIACO	22096.6	22494.5	22834.0	22874.8	27865.6	28410.3	28567.5	28622.1	30749.1	31347.1	31716.1	31805.9
HIACO-I	21921.6	22288.7	22675.2	22694.4	27708.1	28263.7	28504.5	28497.6	30603.2	31262.6	31710.9	31803.1
HIACO-II	21917.4	22345.8	22756.3	22785.6	27935.5	28378.3	28595.3	28544.7	30620.8	31285.6	31684.2	31766.8

(SACO), RIACO and EIACO. All common algorithmic parameters were set as follows: $\alpha = 1$, $\beta = 5$ and the population size $\mu = 50$. k_s was set to 6 and the number of immigrants was set to $n_{ri} = n_{ei} = n_{hi} = 2$ for RIACO, EIACO, HIACO-I and HIACO-II. ρ was set to 0.5 for SACO and ϵ was set to 1 for HIACO-II.

DTSPs are generated from three stationary benchmark instances obtained from TSPLIB[2] using the DBGP generator with f set to 10 and 100 indicating quickly and slowly changing environments, respectively, and m set to 0.1, 0.25, 0.5 and 0.75, indicating slowly, to medium, to severely changing environments, respectively. Totally, a series of 8 DTSPs are constructed from each stationary instance.

For each ACO algorithm on a DTSP, 30 independent runs were executed on the same set of random seed numbers. For each run 1000 iterations were allowed and the best so far ant after a dynamic change was recorded every iteration. The overall offline performance [10] of an ACO on a DTSP is defined as follows:

$$\bar{P}_{OFF} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R}P_{ij}^{*}\right), \tag{10}$$

where R is the number of runs, E is the number of iterations, and P_{ij}^{*} defines the tour cost of the best ant of iteration i of run j. Moreover, the population diversity [13] was recorded as follows:

$$\bar{T}_{DIV} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R}DIV_{ij}\right), \tag{11}$$

where DIV_{ij} is the diversity of the population in iteration i of run j. For the DTSPs, DIV_{ij} can be calculated as follows:

$$DIV_{ij} = \frac{1}{\mu(\mu-1)}\sum_{p=1}^{\mu}\sum_{q\neq p}^{\mu}M(p,q), \tag{12}$$

where μ is the size of population and $M(p,q)$ is the similarity metric between ants p and q defined as:

$$M(p,q) = 1 - \frac{c_{E_{pq}}}{n}, \tag{13}$$

where c_E is defined as the number of common edges (arcs) between the solutions of ants p and q, and n is the number of cities. A value $M(p,q)$ closer to 0 means that the ants are more similar.

B. Basic Experimental Results and Analysis

The experimental results regarding the offline performance of the investigated ACO algorithms for each dynamic test case are presented in Table I. The corresponding statistical results are presented in Table II, where Kruskal–Wallis tests were applied followed by posthoc paired comparisons using Mann–Whitney tests with the Bonferroni correction. In Table II, the results are shown as "+", "−" and "∼" when the first algorithm is significantly better than the second one, when the second algorithm is significantly better than the first one and when the two algorithms are not significantly different, respectively.

Moreover, the dynamic behaviour of ACO algorithms in terms of offline performance of the first 10 environmental changes with $f = 10$ and $m = 0.1$ for the first 5 environmental changes with $f = 100$ and $m = 0.75$ is presented in Fig. 1 and Fig. 2, respectively. The corresponding dynamic behaviour in terms of population diversity is presented in Fig. 3 and Fig. 4. From the experimental results, several observations can be made by comparing the behaviour of the investigated algorithms.

First, RIACO significantly outperforms SACO in all dynamic test cases; see the comparisons of RIACO ⇔ SACO in Table II. This supports our claim that the conventional SACO biases the population towards non-promising areas due to the high concentration of pheromone trails generated from previous environments. On the other hand, RIACO eliminates the pheromone trails of the previous environments directly because of the different pheromone update policy. Furthermore, RIACO significantly outperforms or it is comparable with EIACO

[2]http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

TABLE II. EXPERIMENTAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF ACO ALGORITHMS

Algorithms & DTSPs	kroA100				kroA150				kroA200			
$f = 10, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
RIACO $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
EIACO $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
EIACO $\Leftrightarrow$ RIACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-I $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-I $\Leftrightarrow$ RIACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-I $\Leftrightarrow$ EIACO	−	−	−	−	−	−	−	−	−	−	−	−
HIACO-II $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-II $\Leftrightarrow$ RIACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-II $\Leftrightarrow$ EIACO	−	−	+	+	−	−	+	+	∼	∼	+	+
HIACO-II $\Leftrightarrow$ HIACO-I	+	+	+	+	+	+	+	+	+	+	+	+
$f = 100, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
RIACO $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
EIACO $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
EIACO $\Leftrightarrow$ RIACO	∼	∼	−	−	+	∼	+	∼	∼	+	+	+
HIACO-I $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-I $\Leftrightarrow$ RIACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-I $\Leftrightarrow$ EIACO	∼	+	+	+	+	+	∼	+	+	+	∼	∼
HIACO-II $\Leftrightarrow$ SACO	+	+	+	+	+	+	+	+	+	+	+	+
HIACO-II $\Leftrightarrow$ RIACO	+	+	∼	∼	∼	∼	+	+	+	+	+	+
HIACO-II $\Leftrightarrow$ EIACO	∼	∼	+	+	∼	∼	∼	∼	+	∼	∼	∼
HIACO-II $\Leftrightarrow$ HIACO-I	∼	∼	−	−	−	−	−	∼	∼	∼	∼	∼

in some dynamic test cases (i.e., kroA100) with $f = 100$ and $m = 0.5$ and $m = 0.75$ which can be observed from Fig. 2. However, RIACO usually performs significantly worse than EIACO in most DTSPs because random immigrants may disturb the optimization process. This can be observed from Fig. 3 and Fig. 4 where RIACO maintains the highest diversity among other ACO algorithms that follow the same pheromone update policy.

Second, EIACO significantly outperforms SACO in all dynamic test cases and RIACO in most dynamic test cases (except DTSPs with $m = 0.5$ and $m = 0.75$); see the comparisons of EIACO $\Leftrightarrow$ SACO and EIACO $\Leftrightarrow$ RIACO, respectively, in Table II. This supports our claim when the changing environments are similar then the knowledge transferred via elitism-based immigrants in EIACO may guide the population towards promising areas which can be observed from Fig. 1. Moreover, it can be observed from Fig. 3 and Fig. 4 that EIACO maintains the lowest diversity than its competitors.

Third, HIACO-I significantly outperforms SACO and RIACO in most dynamic test cases; see the comparisons of HIACO-I in Table II. When comparing the performance of HIACO-I over EIACO it can be observed that HIACO-I significantly outperforms EIACO on DTSPs with $f = 100$, whereas EIACO significantly outperforms HIACO-I on DTSPs with $f = 10$. Fig. 2 shows that HIACO-I often performs better than its competitors. This supports our claims that the hybridization may further improve the performance of ACO algorithms since it may keep a good balance of the diversity generated. This can be observed from Fig. 3 and Fig. 4 where HIACO-I maintains lower level of diversity than RIACO but higher level of diversity than EIACO.

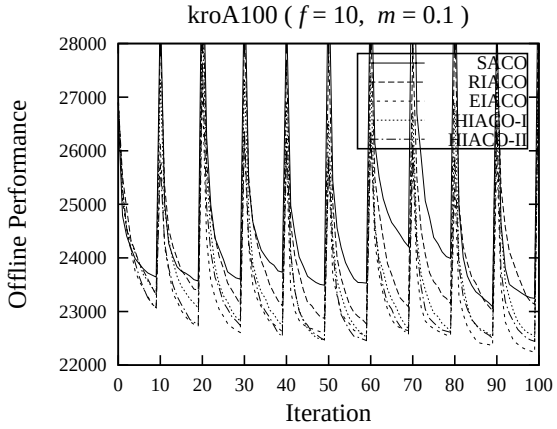
Finally, HIACO-II significantly outperforms SACO and RIACO in most dynamic test cases; see the comparison of HIACO-II in Table II. When comparing the performance of HIACO-II over EIACO it can be observed that HIACO-II significantly outperforms EIACO on DTSPs with $f = 10$ and $m = 0.5$ and $m = 0.75$, whereas EIACO is comparable with HIACO-II on most DTSPs with $f = 100$. When comparing the performance of the two hybrid algorithms it can be seen that HIACO-II significantly outperforms HIACO-I in DTSPs with $f = 10$ whereas HIACO-I outperforms HIACO-II in DTSPs with $f = 100$; see the comparisons of HIACO-II $\Leftrightarrow$ HIACO-I in Table II.

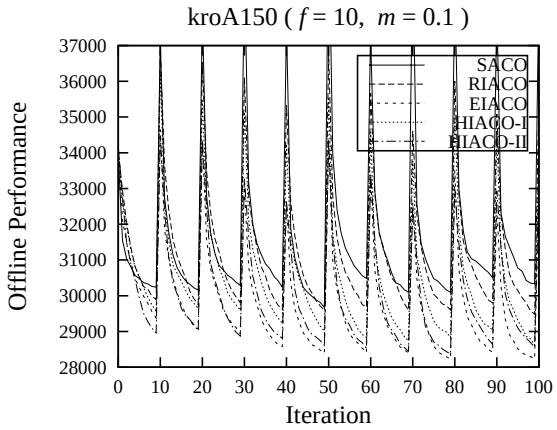
C. Experimental Results of the ϵ Parameter in HIACO-II

The performance of HIACO-II is furthermore investigated with different ϵ parameters defined in Eq. (9). This parameter defines the threshold to trigger random immigrants. For example, when $\epsilon = 1.0$ then random immigrants are triggered when HIACO-II reaches stagnation behaviour. When $\epsilon > 1.0$, then random immigrants are triggered when HIACO-II reaches near to stagnation behaviour. The offline performance of HIACO-II with different ϵ values is given in Fig. 5 for DTSPs with $f = 100$[3].

In the basic experiments ϵ was set to 1.0 that indicates stagnation behaviour. However, it can be clearly observed from Fig. 5 that when $\epsilon = 1.1$ the performance of HIACO-II is improved on most DTSPs. This shows that once the population reaches stagnation behaviour, then it is difficult to escape from it even if only random immigrants are generated. However,

[3]The corresponding results for DTSPs with $f = 10$ have similar observations and thus they are not presented





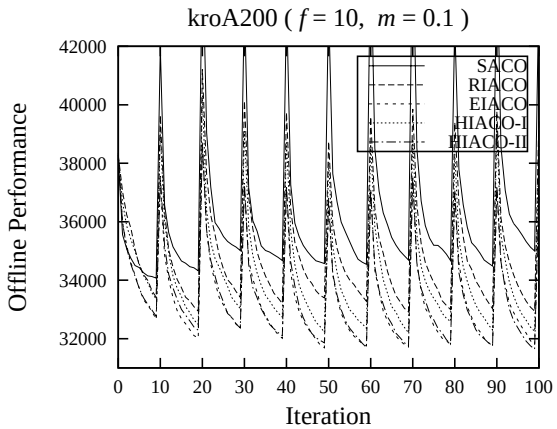
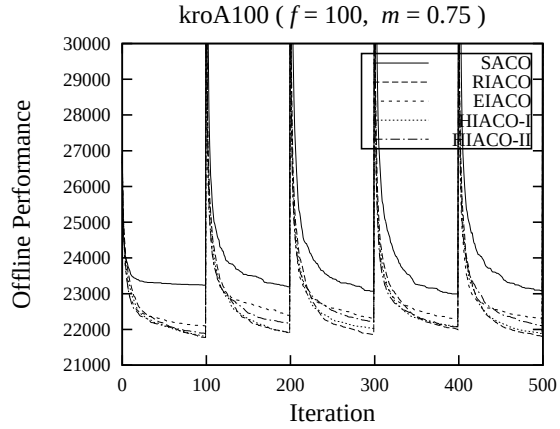
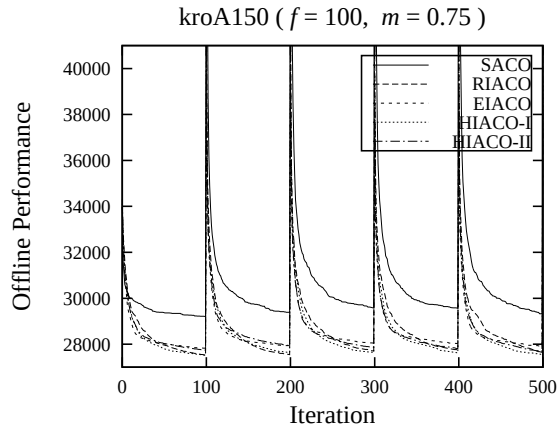


Fig. 1. Dynamic overall offline performance of ACO algorithms on DTSPs with $f = 10$ and $m = 0.1$ for the first 10 environmental changes.





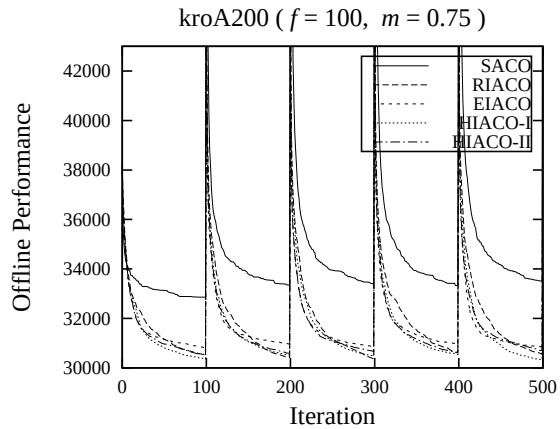


Fig. 2. Dynamic overall offline performance of ACO algorithms on DTSPs with $f = 100$ and $m = 0.75$ for the first 5 environmental changes.

when ϵ is set to a higher value (i.e., $\epsilon > 1.1$) the performance is always degraded because random immigrants are generated most of the times and may disturb the optimization process. Hence a good value of ϵ of HIACO-II in DOPs is in the range of $[1.0, 1.1]$.

VI. Conclusions and Future Work

The integration of immigrants schemes with ACO algorithms showed good performance on different DOPs. The performance of different immigrants schemes depends on the characteristics of the DOP. In this paper, we hybridize random and elitism-based immigrants and investigate their interaction. Two hybrid immigrants schemes are proposed, i.e., non-interactive and interactive hybrid immigrants, denoted HIACO-I and HIACO-II in this paper, respectively. Within the former scheme random and elitism-based immigrants are both generated in every algorithmic iteration, whereas within the latter scheme elitism-based immigrants are generated in every algorithmic iteration and random immigrants are triggered whenever stagnation behaviour occurs.

The two hybrids are compared with other peer ACO algorithms on different dynamic tests cases of the DTSP. From

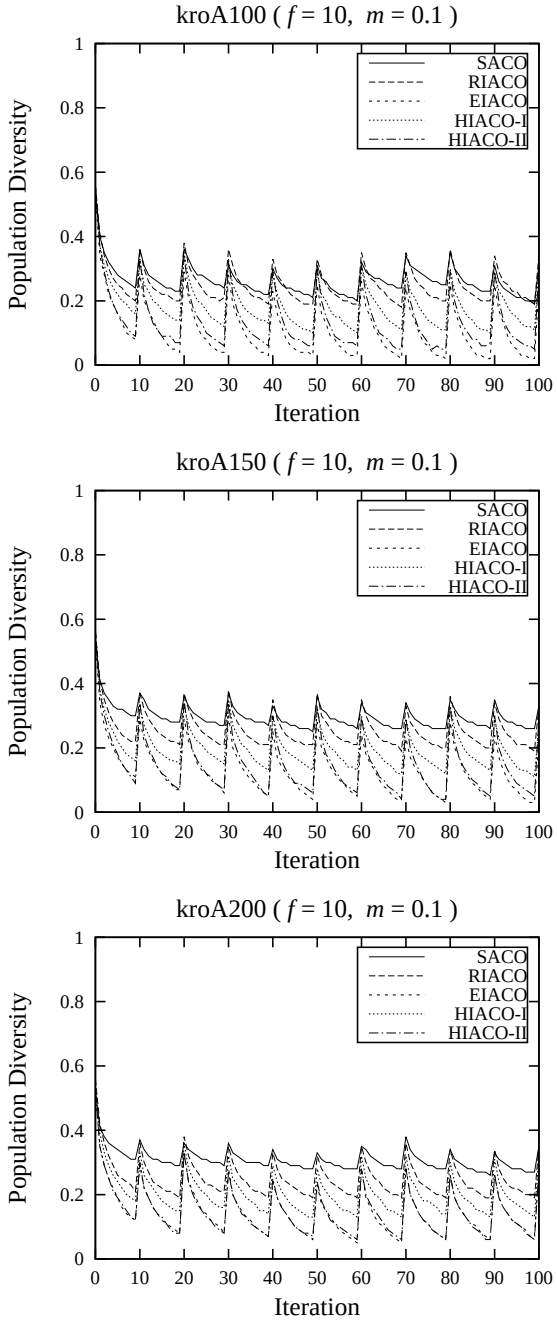


Fig. 3. Dynamic total population diversity of ACO algorithms on DTSPs with $f = 10$ and $m = 0.1$ for the first 10 environmental changes.

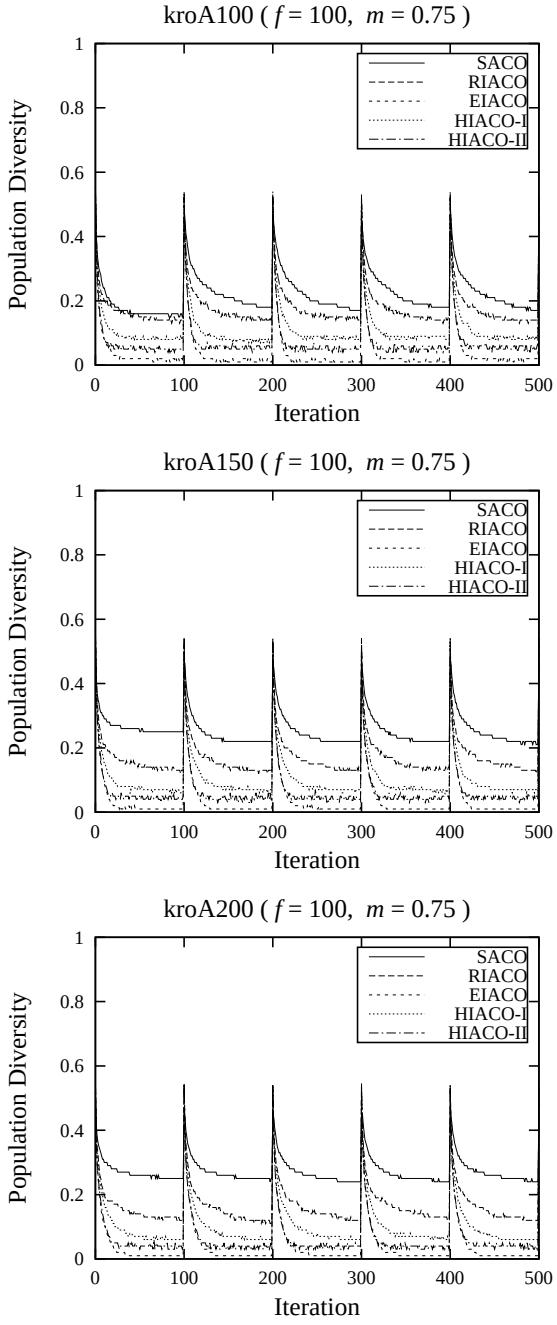


Fig. 4. Dynamic total population diversity of ACO algorithms on DTSPs with $f = 100$ and $m = 0.75$ for the first 5 environmental changes.

the experiments several concluding remarks can be drawn. First, random immigrants schemes are helpful for some DTSPs but they may be harmful on other DTSPs because of their inconsistent behaviour. Second, elitism-based immigrants are more consistent than random immigrants and perform well on DTSPs that the changing environments are similar. Third, both interactive and non-interactive hybrid immigrants combine the principles of random and elitism-based immigrants effectively. Finally, a high level of diversity generated or too much knowledge transferred does not always lead to a better performance of ACO algorithms in dynamic environments.

Generally speaking, interactive and non-interactive hybrid immigrants schemes are a good choice for ACO algorithms to address quickly and slowly changing environments, respectively.

There are several relevant future works. First, hybridize other immigrants schemes, e.g., random immigrants and memory-based immigrants [13], [14] that can be useful in DOPs where the environments re-appear (i.e., cyclic dynamic environments). Second, it will be interesting to apply hybrid immigrants to other DOPs, e.g., dynamic vehicle routing problem [12].

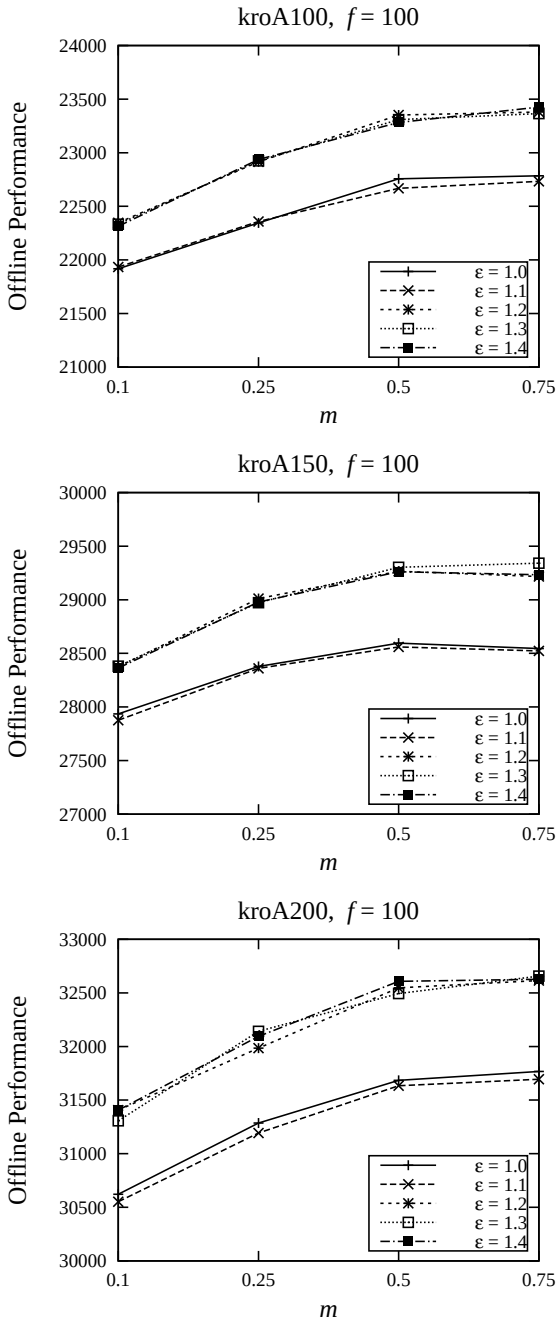


Fig. 5. Offline performance of HIACO-II with different ϵ values on DTSPs with $f = 100$.

ACKNOWLEDGMENT

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/K001310/1

REFERENCES

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

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, Eds., *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press, 1997.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on System Man and Cybernetics-Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[4] M. Dorigo and T. Stützle, Eds., *Ant colony optimization*. London, England: MIT Press, 2004.

[5] C. Eyckelhof and M. Snoek, "Ant systems for a dynamic tsp," in *Proceedings of the 3rd International Workshop on Ant Algorithms*, ser. LNCS, M. Dorigo, G. D. Caro, and M. Sampels, Eds., vol. 2463. Springer-Verlag, 2002, pp. 88–99.

[6] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Proceedings of the 12th Int. Conf. on Machine Learning*. Morgan Kaufmann, 1995, pp. 252–260.

[7] L. M. Gambardella, E. D. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem," *Journal of the Operational Research Society*, vol. 50, pp. 167–176, 1999.

[8] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic tsp," in *EvoWorkshops 2001: Applications of Evolutionary Computing*, ser. LNCS, vol. 2037. Springer-Verlag, 2001, pp. 213–222.

[9] ——, "Applying population based aco to dynamic optimization problems," in *Proceedings of the 3rd International Workshop on Ant Algorithms*, ser. LNCS, M. Dorigo, G. D. Caro, and M. Sampels, Eds., vol. 2463. Springer-Verlag, 2002, pp. 111–122.

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

[11] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for dynamic environments," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, ser. LNCS, vol. 6239. Springer-Verlag, 2010, pp. 371–380.

[12] ——, "Adapting the pheromone evaporation rate in dynamic routing problems," in *EvoApplications 2013: Applications of Evolutionary Computation*, ser. LNCS, vol. 7835. Springer-Verlag, 2013, pp. 606–615.

[13] ——, "Ant colony optimization algorithms with immigrants schemes for the dynamic travelling salesman problem," in *Evolutionary Computation for Dynamic Optimization Problems*, S. Yang and X. Yao, Eds. Springer-Verlag, 2013, ch. 13, pp. 331–357.

[14] ——, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," *Applied Soft Computing*, vol. 13, no. 10, pp. 4023–4037, 2013.

[15] M. Mavrovouniotis, S. Yang, and X. Yao, "A benchmark generator for dynamic permutation-encoded problems," in *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature*, ser. LNCS, vol. 7492. Springer-Verlag, 2012, pp. 508–517.

[16] L. Melo, F. Pereira, and E. Costa, "Multi-caste ant colony algorithm for the dynamic traveling salesperson problem," in *Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms*, ser. LNCS, M. Tomassini, A. Antonioni, F. Daolio, and P. Buesser, Eds., vol. 7824. Springer-Verlag, 2013, pp. 179–188.

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

[18] S. Yang and R. Tinós, "A hybrid immigrants scheme for genetic algorithms in dynamic environments," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 243–254, 2007.

[19] X. Yu, K. Tang, T. Chen, and X. Yao, "Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization," *Memetic Computing*, vol. 1, no. 1, pp. 3–24, 2009.