An Immigrants Scheme Based on Environmental Information for Ant Colony Optimization for the Dynamic Travelling Salesman Problem

Michalis Mavrovouniotis[1] and Shengxiang Yang[2]

[1] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
mm251@mcs.le.ac.uk
[2] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex UB8 3PH, United Kingdom
shengxiang.yang@brunel.ac.uk

Abstract. Ant colony optimization (ACO) algorithms have proved to be powerful methods to address dynamic optimization problems. However, once the population converges to a solution and a dynamic change occurs, it is difficult for the population to adapt to the new environment since high levels of pheromone will be generated to a single trail and force the ants to follow it even after a dynamic change. A good solution is to maintain the diversity via transferring knowledge to the pheromone trails. Hence, we propose an immigrants scheme based on environmental information for ACO to address the dynamic travelling salesman problem (DTSP) with traffic factor. The immigrants are generated using a probabilistic distribution based on the frequency of cities, constructed from a number of ants of the previous iteration, and replace the worst ants in the current population. Experimental results based on different DTSP test cases show that the proposed immigrants scheme enhances the performance of ACO by the knowledge transferred from the previous environment and the generation of guided diversity.

1 Introduction

Ant colony optimization (ACO) algorithms have proved to be powerful metaheuristics for solving difficult real-world optimization problems under static environments [2, 12]. ACO is inspired from real ant colonies, where a population of ants searches for food from their nest. Ants communicate via their pheromone trails and they are able to track shortest paths between their nest and food sources. Inspired from this behaviour, ACO algorithms are able to locate the optimum, or a near optimum solution, in stationary optimization problems, efficiently [10]. However, in many real-world problems we have to deal with dynamic environments, where the optimum is moving, and the objective is not only to locate the optimum but to track it also [7]. ACO algorithms face a serious challenge in dynamic optimization problems (DOPs) because they lose their adaptation capabilities once the population has converged [1].

Over the years, several approaches have been developed for ACO algorithms to enhance their performance for DOPs, such as local and global restart strategies [6], memory-based approaches [5], pheromone update schemes to maintain diversity [3], and immigrants schemes to increase diversity [8, 9]. Most of these approaches have been applied in different variations of the dynamic travelling salesman problem (DTSP), since it has many similarities with many real-world applications [11]. Among these approaches, immigrants schemes have been found beneficial to deal with DTSPs, where immigrant ants are generated and used to replace other ants of the current population [8, 9]. The most important concern when applying immigrants schemes is how to generate immigrants.

In this paper, we propose an immigrants scheme which is based on environmental information for ACO for the DTSP. The environmental information-based immigrants ACO (EIIACO) selects the first n best ants from the previous environment and creates a probabilistic distribution based on the frequency of cities that appear next to each other. Using the information obtained from the previous population, immigrants are generated to replace the worst ants in the current population. The introduced immigrants transfer knowledge and can guide the population toward the promising regions after a change occurs. It is expected that the environmental information-based immigrants scheme may enhance the performance of ACO for DTSPs, especially when the environments before and after a change are similar.

The remaining of this paper is organized as follows. Section 2 describes the DTSP used in the experimental study in this paper. Section 3 describes existing ACO algorithms for the DTSP, which are also investigated as peer algorithms in the experiments. Section 4 describes the proposed EIIACO algorithm. The experimental results and analysis are presented in Section 5. Finally, the conclusions and relevant future work are presented in Section 6.

2 DTSP with the Traffic Jam

The TSP is the most fundamental and well-known *NP*-hard combinatorial optimization problem. It can be described as follows: Given a collection of cities, we need to find the shortest path that starts from one city and visits each of the other cities once and only once before returning to the starting city.

In this paper, we generate a DTSP via introducing the traffic factor. We assume that the cost of the link between cities i and j is $C_{ij} = D_{ij} \times F_{ij}$, where D_{ij} is the normal travelled distance and F_{ij} is the traffic factor between cities i and j. Every f iterations a random number R in $[F_L, F_U]$ is generated probabilistically to represent traffic between cities, where F_L and F_U are the lower and upper bounds of the traffic factor, respectively. Each link has a probability m to add traffic by generating a different R each time, such that $F_{ij} = 1 + R$, where the traffic factor of the remaining links is set to 1 (indicates no traffic). Note that f and m denote the frequency and magnitude of the changes in the dynamic environment, respectively. The TSP becomes more challenging and realistic when it is subject to a dynamic environment. For example, a traffic factor

closer to the upper bound F_U represents rush hour periods which increases the travelled distance significantly. On the other hand, a traffic factor closer to the lower bound F_L represents normal hour periods which increases the travelled distance slightly.

3 ACO for the DTSP

3.1 Standard ACO

The standard ACO (S-ACO) algorithm, i.e., Max-Min AS (MMAS), consists of a population of μ ants [12]. Initially, all ants are placed to a randomly selected city and all pheromone trails are initialized with an equal amount of pheromone. With a probability $1 - q_0$, where $0 \leq q_0 \leq 1$ is a parameter of the decision rule, ant k chooses the next city j, while its current city is i, probabilistically, as follows:

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

where τ_{ij} and $\eta_{ij} = 1/C_{ij}$ are the existing pheromone trails and heuristic information available a priori between cities i and j, respectively, N_i^k denotes the neighbourhood of cities of ant k that have not yet been visited when its current city 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 with the maximum probability, i.e., $[\tau]^\alpha[\eta]^\beta$, and not probabilistically as in Eq. (1).

Later on, the best ant retraces the solution and deposits pheromone according to its solution quality on the corresponding trails as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall\ (i,j) \in T^{best}, \tag{2}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}$ is the amount of pheromone that the best ant deposits and C^{best} is the tour cost of T^{best}. However, before adding any pheromone, a constant amount of pheromone is deduced from all trails due to the pheromone evaporation such that, $\tau_{ij} \leftarrow (1-\rho)\,\tau_{ij}$, $\forall\ (i,j)$, where $0 < \rho \leq 1$ is the rate of evaporation. The pheromone trail values are kept to the interval $[\tau_{min}, \tau_{max}]$ and they are re-initialized to τ_{max} every time the algorithm shows a stagnation behaviour, where all ants follow the same path, or when no improved tour has been found for several iterations [12].

The S-ACO algorithm faces a serious challenge when it is applied to DTSPs. The pheromone trails of the previous environment will not make sense to the new one. From the initial iterations the population of ants will eventually converge to a solution, and, thus, high intensity of pheromone trails will be generated into a single path. The pheromone trails will influence ants to the current path even after a dynamic change. The pheromone evaporation is the only mechanism used to eliminate the pheromone trails generated previously, and help ants to adapt to the new environment. Therefore, S-ACO requires a sufficient amount of time in order to recover when a dynamic change occurs.

3.2 Population-Based ACO

The population-based ACO (P-ACO) algorithm is the memory based version of ACO [5]. It differs from the S-ACO algorithm described above since it follows a different framework. The algorithm maintains a population of ants (solutions), called population-list (memory), which is used to update pheromone trails without any evaporation.

The initial phase and the first iterations of the P-ACO algorithm work in the same way as with the S-ACO algorithm. The pheromone trails are initialized with an equal amount of pheromone and the population-list of size K is empty. For the first K iterations, the iteration-best ant deposits a constant amount of pheromone, using Eq. (2) with $\Delta\tau_{ij}^{best} = (\tau_{max} - \tau_{init})/K$. Here, τ_{max} and τ_{init} denote the maximum and initial pheromone amount, respectively. This positive update procedure is performed whenever an ant enters the population-list. On iteration $K + 1$, the ant that has entered the population-list first, i.e., the oldest ant, needs to be removed in order to make room for the new one, and its pheromone trails are reduced by $\Delta\tau_{ij}^{best}$, which equals to the amount added when it entered the population-list before.

The population-list is a long-term memory, denoted as k_{long}, since it may contain ants from previous environments that survive in more than one iteration. Therefore, when a change occurs, the ants stored in the population-list are re-evaluated in order to be consistent with the new environment where different traffic factors F_{ij} are introduced. The pheromone trails are updated accordingly using the ants currently stored in the population-list.

The P-ACO algorithm has a more aggressive mechanism to eliminate previously generated trails since the corresponding pheromone of the ants that are replaced by other ants in the population-list are removed directly. However, they face the same challenge with S-ACO because identical ants may be stored in the population-list and dominate the search space with a high intensity of pheromone to a single trail. However, the P-ACO algorithm may be beneficial when a new environment is similar to an old one because the solutions stored in the population-list from the previous environments will guide ants towards promising areas in the search space.

4 Immigrants Based on Environmental Information

Many immigrants schemes have been found beneficial in genetic algorithms (GAs) for binary-encoded DOPs [4, 13, 14]. Therefore, to handle the problems described in S-ACO and P-ACO algorithms when addressing DTSPs, immigrants schemes are integrated with ACO since they maintain a certain level of diversity during the execution and transfer knowledge from previous environments [8, 9]. The immigrants are integrated within P-ACO as follows. A short-term memory is used, denoted as k_{short}, where the ants survive only for one iteration. All the ants of the current iteration replace the old ones, instead of only replacing the oldest one as in P-ACO. Then, a predefined portion of the worst ants are

Algorithm 1 Generate Frequency Of Cities

```
1: input k_short      // short term memory
2: input n            // size of k_short
3: for k = 1 to n do
4:    for i = 1 to l do
5:       city_one_id = ant[k].tour[i]
6:       city_two_id = ant[k].tour[i + 1]
7:       frequency_of_cities[city_one_id][city_two_id]+=1
8:       frequency_of_cities[city_two_id][city_one_id]+=1
9:    end for
10: end for
11: return frequency_of_cities
```

replaced by immigrant ants in k_{short}. When ants are removed from k_{short}, a negative update is made to their pheromone trails and when new ants are added to k_{short}, a positive update is made to their pheromone trails as in P-ACO.

Different immigrants schemes were integrated with P-ACO, such as the traditional immigrants [8], where immigrant ants are generated randomly, the elitism-based immigrants [8], where immigrant ants are generate using the best ant from k_{short}, and the memory-based immigrants [9], where immigrant ants are generated using the best ant from k_{long}. The information obtained from the elitism- and memory-based immigrants to transfer knowledge is based on individual information (one ant). The proposed EIIACO algorithm generates immigrants using environmental information (population of ants) to transfer knowledge from the previous environment to a new one. EIIACO follows the same framework with other ACO algorithms based on immigrants schemes, as described above, but differs in the way immigrant ants are generated.

Environmental information-based immigrants are generated using all the ants stored in k_{short} of the previous environment. Within EIIACO, a probabilistic distribution based on the frequency of cities is extracted, representing information of the previous environment, which is used as the base to generate immigrant ants. The frequency vector of each city c_i, i.e, $\boldsymbol{D}_{c_i}$, is constructed by taking the ants of k_{short} as a dataset and locating city c_i from them. The successor and predecessor cities, i.e., c_{i-1} and c_{i+1}, respectively, of city c_i are obtained and update $\boldsymbol{D}_{c_i}$ accordingly. For example, one is added to the corresponding position $i-1$ and $i+1$ in $\boldsymbol{D}_{c_i}$. The process is repeated for all cities and a table $S = (\boldsymbol{D}_{c_1}, \ldots, \boldsymbol{D}_{c_l})$ is generated (where l is the number of cities) as represented in Algorithm 1.

An environmental information-based immigrant ant, i.e., $A_{eii} = (c_1, \ldots, c_l)$, is generated as follows. First, randomly select the start city c_1; then, the probabilistic distribution of $\boldsymbol{D}_{c_{i-1}} = (d_1, \ldots, d_l)$ is used to select the next city c_i probabilistically as follows:

$$p_i = \frac{d_i}{\sum_{j \in \boldsymbol{D}_{c_{i-1}}} d_j}, \text{if } i \in \boldsymbol{D}_{c_{i-1}}, \tag{3}$$

Algorithm 2 Generate Environmental Information-Based Immigrant

```
1: input l                    // number of cities
2: input k                    // immigrant ant identifier
3: input frequency_of_cities  // see Algorithm 1
4: step = 1                   // counter for construction step
5: ant[k].tour[step] = random[1, l]
6: while step < l do
7:    step+=1
8:    current_city = ant[k].tour[step − 1]
9:    sum_probabilities = 0.0
10:   for j = 1 to l do
11:      if ant[k].visited[j] then
12:         probability[j] = 0.0
13:      else
14:         probability[j] = frequency[current_city][j]
15:         sum_probabilities += probability[j]
16:      end if
17:   end for
18:   if sum_probabilities = 0.0 then
19:      selected = random[1, l]
20:      while ant[k].visited[selected] do
21:         selected = random[1, l]
22:      end while
23:      ant[k].tour[step] = selected
24:   else
25:      r = random[0, sum_probabilities]
26:      selected = 1
27:      p = probability[selected]
28:      while p < r do
29:         selected+=1
30:         p += probability[selected]
31:      end while
32:      ant[k].tour[step] = selected
33:   end if
34: end while
35: ant[k].tour[l + 1] = ant[k].tour[1]
36: return ant[k]             // generated immigrant ant
```

where d_i is the frequency number where city c_i appears before or after city c_{i-1}. Note that all cities currently selected and stored in A_{eii} have a probability of 0.0 to be selected since they are already visited. In the case where the sum of $p_i = 0.0$, which means that all cities in $\boldsymbol{D}_{c_{i-1}}$ are visited, a random city j that has not been visited yet is selected. This process is repeated until all cities are used in order to generate a valid immigrant ant based on the environmental information, as represented in Algorithm 2. During lines 7–17, the probabilistic distribution is generated, during lines 18–24, the next city is selected randomly from the unvisited cities, and during lines 25–33, the next city is selected probabilistically

from the frequency of cities generated from Algorithm 1. Note that in line 35 the first city stored in the ant is added to the end since the TSP tour is cyclic.

5 Simulation Experiments

5.1 Experimental Setup

In the experiments, we compare the proposed EIIACO algorithm with the S-ACO and P-ACO algorithms, which are described in Section 3. Our implementation follows the guidelines of the ACOTSP[3] framework. All the algorithms have been applied to the `eil76`, `kroA100`, and `kroA200` problem instances, obtained from TSPLIB[4]. Most of the parameters have been optimized and obtained from our preliminary experiments while others have been inspired from the literature [5, 8, 9]. For all algorithms, $\mu = 25$ ants are used, $\alpha = 1$, $\beta = 5$ and $q_0 = 0.0$ (except P-ACO where $q_0 = 0.9$). Moreover, for S-ACO, $\rho = 0.2$. For P-ACO, $\tau_{max} = 1.0$, and the size of k_{long} is 3. For EIIACO, the size of k_{short} is 10 and four immigrant ants are generated. For each algorithm on a DTSP instance, $N = 30$ independent runs were executed on the same dynamic changes. The algorithms were executed for $G = 1000$ iterations and the overall offline performance of an algorithm is calculated as follows:

$$\bar{P}_{offline} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N}P_{ij}^{*}\right), \tag{4}$$

where P_{ij}^{*} defines the tour cost of the best ant since the last dynamic change of iteration i of run j [7].

The value of f was set to 20 and 100, which indicates fast and slowly changing environments, respectively. The probability of m was set to 0.1, 0.25, 0.5, and 0.75, which indicates the degree of environmental changes from small, to medium, to large, respectively. The intervals of the traffic factor were set to $F_L = 0$ and $F_U = 5$. As a result, 8 dynamic environments, i.e., 2 values of $f\times$ 4 values of m, were generated from each stationary TSP instance, to systematically analyze the adaptation and searching capability of each algorithm in the DTSP.

5.2 Experimental Results and Analysis

The experimental results regarding the overall offline performance of the algorithms for DTSPs are presented in Table 1. The corresponding statistical results of two-tailed t-test with 58 degree of freedom at a 0.05 level of significance are presented in Table 2, where "$s+$" or "$s-$" means that the first or the second algorithm is significantly better, respectively, and "+" or "−" means that the first or the second algorithm is insignificantly better, respectively. Moreover, to

[3] http://www.aco-metaheuristic.org/aco-code/
[4] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Table 1. Experimental results of algorithms regarding the offline performance

Alg. & Inst.	eil76							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
S-ACO	403.1	441.1	541.7	752.4	378.5	426.5	505.6	711.5
P-ACO	396.9	437.2	538.5	750.9	381.5	432.3	511.8	723.8
EIIACO	392.5	432.0	532.8	739.3	379.3	428.5	506.4	710.6
Alg. & Inst.	kroA100							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
S-ACO	18806.3	21106.1	26000.6	35812.8	17726.2	19351.3	23672.0	34232.5
P-ACO	18043.5	20767.0	25777.5	35544.5	17891.0	19656.8	24089.5	34633.1
EIIACO	18041.3	20710.7	25447.4	34963.1	17789.5	19520.1	23800.9	34106.4
Alg. & Inst.	kroA200							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
S-ACO	25979.4	29058.4	36383.0	50344.3	23519.0	26037.0	33583.6	45338.0
P-ACO	24621.4	28525.8	35924.7	49620.5	23529.9	26284.5	33761.4	45314.5
EIIACO	24686.9	28132.4	35164.3	48190.4	23423.4	26136.9	33302.5	44388.0

Table 2. Statistical tests of comparing algorithms regarding the offline performance

Alg. & Inst.	eil76				kroA100				kroA200			
$f = 20, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
S-ACO $\Leftrightarrow$ P-ACO	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
EIIACO $\Leftrightarrow$ P-ACO	$s+$	$s+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$
EIIACO $\Leftrightarrow$ S-ACO	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$f = 100, m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
S-ACO $\Leftrightarrow$ P-ACO	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$+$	$s+$	$s+$	$-$
EIIACO $\Leftrightarrow$ P-ACO	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIIACO $\Leftrightarrow$ S-ACO	$s-$	$s-$	$-$	$+$	$s-$	$s-$	$s-$	$s+$	$s+$	$s-$	$s+$	$s+$

better understand the dynamic behaviour of the algorithms, the offline performance of the first 500 iterations is plotted in Fig. 1 for fast and slowly changing environments with $m = 0.1$ and $m = 0.75$, respectively. From the experimental results several observations can be made and are analyzed as follows.

First, S-ACO significantly outperforms P-ACO in almost all slowly changing environments whereas it is beaten in fast changing environments; see the results of S-ACO $\Leftrightarrow$ P-ACO in Table 2. This result validates that the S-ACO algorithm can adapt to dynamic changes, but it needs sufficient time to recover and locate the new optimum. The S-ACO algorithm uses pheromone evaporation in order to eliminate pheromone trails that are not useful for the new environment and helps the population of ants to forget the previous solution where they have converged to. On the other hand, P-ACO uses more aggressive method to elim-

inate previous pheromone trails, which guides the population of ants to keep up with the changing environments, even if they change fast. From Fig. 1, it can be observed that P-ACO converges faster than S-ACO, which helps in fast changing environments but not in slowly changing environments. As we have discussed previously, P-ACO has a high risk to maintain identical ants in the population-list and may get trapped in a local optimum solution.

Second, EIIACO outperforms P-ACO in almost all dynamic test environments; see the results of EIIACO $\Leftrightarrow$ P-ACO in Table 2. However, P-ACO is competitive with EIIACO when the environment is slightly changing, e.g., in the `kroA200` with $f = 20$ and $m = 0.1$. This is because the solutions stored in the population-list of P-ACO may be fit only when the previous environment has many similarities with the new one. In cases where the environmental changes are medium to significant, the environmental information-based immigrants transfer more knowledge to the pheromone trails of the next iteration and increase the diversity. In slowly changing environments, EIIACO outperforms P-ACO in all dynamic test cases, either with small or large magnitude of changes. This is because EIIACO has enough time to gain knowledge from the previous environment.

Third, EIIACO outperforms S-ACO in all fast changing environments whereas it is competitive in some slowly changing environments; see the results of EIIACO $\Leftrightarrow$ S-ACO in Table 2. On the smallest problem instance, i.e., `eil76`, S-ACO is significantly better than EIIACO because the diversity provided from the immigrants scheme may not be helpful. Furthermore, it is easier for the population in S-ACO to forget previous solutions and become more adaptive. This validates our expectation for S-ACO, where the time needed to adapt depends on the size of the problem and the magnitude of change. As the problem size and magnitude of change increases, EIIACO is significantly better than S-ACO because the population in S-ACO needs more time to adapt in more complex problem instances; see Fig. 1. Moreover, it can be observed that when the magnitude of change is small, S-ACO converges slowly to a better solution, whereas when the magnitude of change is large, EIIACO converges quickly to a better solution.

Finally, in order to investigate the effect of the environmental information-based immigrants scheme in the population diversity of ACO, we calculate the mean population diversity of all iterations as follows:

$$\bar{D}iv = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{\mu(\mu-1)}\sum_{p=1}^{\mu}\sum_{q\neq p}^{\mu}M_{pq}\right)\right), \tag{5}$$

where G is the number of iterations, N is the number of runs, μ is the size of the population, $M_{pq} = 1 - \frac{CE(p,q)}{l}$ is the metric that defines the difference between ant p and ant q, where $CE(p,q)$ is the common edges between the ants and l is the number of cities. A value of M_{pq} closer to 0 means that the two ants are similar. The total diversity results for all dynamic test cases are presented in Fig. 2. It can be observed that S-ACO maintains the highest diversity. Especially, in fast changing environments, S-ACO has an extremely high level of diversity, and this

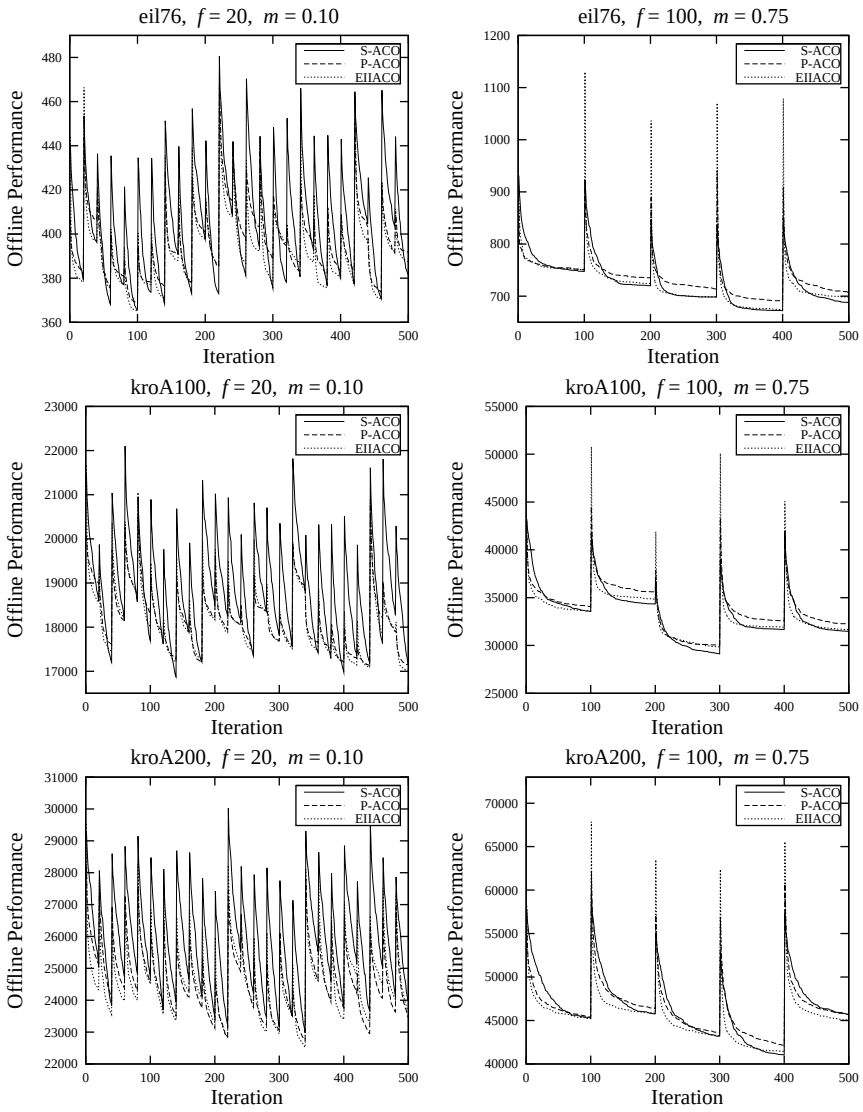

Fig. 1. Dynamic behaviour of investigated algorithms on different DTSPs

shows that it does not have sufficient time to converge. The P-ACO algorithm has the lowest diversity level, which shows the negative effect when identical ants are stored in the population-list. EIIACO maintains higher diversity than P-ACO and much lower diversity than S-ACO. This shows that the diversity generated from the proposed scheme is guided. Moreover, it shows that ACO algorithms that maintain higher diversity levels than others do not always achieve better performance for the DTSP; see Table 1 and Fig. 2.

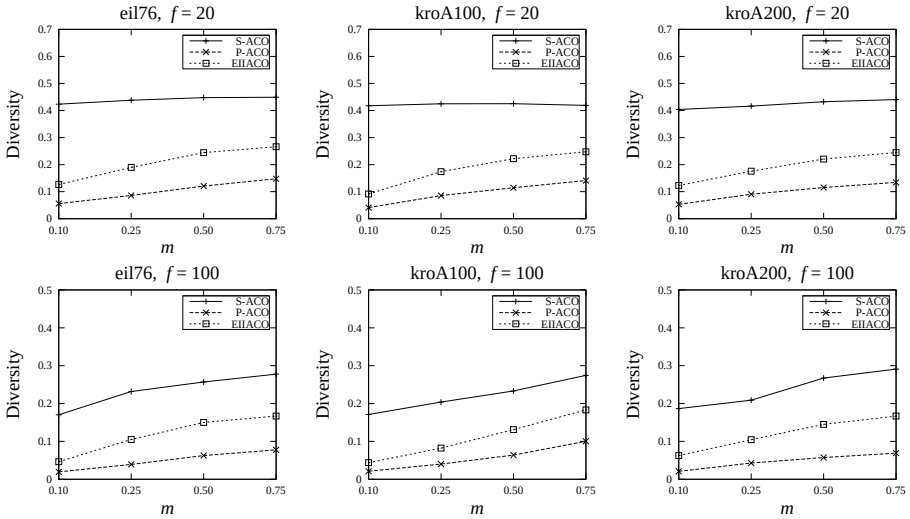

Fig. 2. Overall population diversity of algorithms for different dynamic test problems

6 Conclusions and Future Work

Several immigrants schemes based on individual information, e.g., elitist-based immigrants, have been integrated with ACO algorithms to address different DTSPs in the literature [8, 9]. In this paper, an immigrants scheme based on environmental information, i.e., the frequency of cities appearing next to each other, is proposed for ACO to address the DTSP with traffic factors. A number of generated immigrants replace the worst ants in the current population in order to maintain diversity and transfer knowledge to the pheromone trails for the ants that will construct solutions on the next iteration.

From the experimental results of comparing the proposed EIIACO algorithm with S-ACO and P-ACO algorithms on different cases of DTSPs, the following concluding remarks can be drawn. First, ACO algorithms can be benefited by transferring knowledge from previous environments to the pheromone trails using immigrants schemes for DTSPs. Second, S-ACO has good performance in slowly and slightly changing environments and it is comparable with EIIACO, especially on small problem instances. Third, EIIACO outperforms other ACO algorithms in fast changing environments, while it is comparable with P-ACO in some slightly changing environments. Fourth, EIIACO is significantly better than P-ACO in almost all slowly changing environments. Finally, guided diversity is usually better than random diversity.

For future work, it will be interesting to compare or hybridize EIIACO with other immigrants schemes, which are based on individual information [8, 9], and investigate the interaction between the two types of schemes. As another interesting future work, EIIACO can be applied in more challenging optimization problems, e.g., vehicle routing problems [11].

7 Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/2.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford University Press, New York (1999)
2. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst., Man and Cybern., Part B: Cybern. 26(1), 29–41 (1996)
3. Eyckelhof, C.J., Snoek, M.: Ant system for a dynamic TSP. In: ANTS 2002: Proc. 3rd Int. Workshop on Ant Algorithms, pp. 88–99 (2002)
4. Grefenestette, J.J.: Genetic algorithms for changing environments. In: Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)
5. Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 111–122. Springer, Heidelberg (2002)
6. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
7. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. Evol. Comput. 9(3), 303–317 (2005)
8. Mavrovouniotis, M., Yang, S.: Ant colony optimization with immigrants schemes for dynamic environments. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 371–380. Springer, Heidelberg (2010)
9. Mavrovouniotis, M., Yang, S.: Memory-based immigrants for ant colony optimization in changing environments. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A., Merelo, J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G. (eds.) EvoApplications 2011. LNCS, vol. 6624, pp. 324–333. Springer, Heidelberg (2011)
10. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. Algorithmica 54(2), 243–255 (2009)
11. Rizzoli, A. E., Montemanni, R., Lucibello, E., Gambardella, L. M.: Ant colony optimization for real-world vehicle routing problems – from theory to applications. Swarm Intelli. 1(2), 135–151 (2007)
12. Stützle, T., Hoos, H.: The MAX-MIN ant system and local search for the traveling salesman problem. In: Proc. 1997 IEEE Int. Conf. on Evol. Comput., pp. 309–314 (1997)
13. Yang, S.: Genetic algorithms with memory and elitism based immigrants in dynamic environments. Evol. Comput. 16(3), 385–416 (2008)
14. Yu, X., Tang, K., Yao, X.: An immigrants scheme based on environmental information for genetic algorithms in changing environments. In: Proc. 2008 IEEE Cong. of Evol. Comput., pp. 1141–1147 (2008)
15. Yu, X., Tang, K., Chen, T., Yao, X.: Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. Memetic Comput. 1(1), 3–24 (2009)