Memory-Based Immigrants for Ant Colony Optimization in Changing Environments

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 that they can adapt to dynamic optimization problems (DOPs) when they are enhanced to maintain diversity. DOPs are important due to their similarities to many real-world applications. Several approaches have been integrated with ACO to improve their performance in DOPs, where memory-based approaches and immigrants schemes have shown good results on different variations of the dynamic travelling salesman problem (DTSP). In this paper, we consider a novel variation of DTSP where traffic jams occur in a cyclic pattern. This means that old environments will re-appear in the future. A hybrid method that combines memory and immigrants schemes is proposed into ACO to address this kind of DTSPs. The memory-based approach is useful to directly move the population to promising areas in the new environment by using solutions stored in the memory. The immigrants scheme is useful to maintain the diversity within the population. The experimental results based on different test cases of the DTSP show that the memory-based immigrants scheme enhances the performance of ACO in cyclic dynamic environments.

1 Introduction

In nature, ant colonies have proved that they have a distributed optimization behaviour when they search for food from their nest to food sources. Ants communicate with their pheromone trails and cooperate to optimize the travel between their nest and food sources. Inspired from this behaviour, ant colony optimization (ACO) algorithms have been developed to solve different optimization problems in real-world applications [2, 3]. Traditionally, researchers have focused on stationary optimization problems, where the environment remains fixed during the execution of the algorithm.

However, many real-world applications have dynamic environments, where the optimum needs to be tracked over time [12]. Theoretically, ACO algorithms can adapt to dynamic changes since they are inspired from nature, which is a continuous adaptation process [10]. In practice, they can adapt by transferring knowledge from past environments [1]. The challenge of such algorithms is how quickly they can react to dynamic changes in order to maintain the high quality of output instead of premature convergence. Developing strategies for ACO algorithms to deal with premature convergence and address DOPs has attracted a lot of attention, which include local and global restart strategies [7], memory-based approaches [6, 8], pheromone manipulation schemes to maintain diversity [4], and immigrants schemes to increase diversity [11].

These approaches have been applied to the dynamic travelling salesman problem (DTSP). Among them, the memory and immigrants schemes have proved to be beneficial for the DTSP where cities are replaced. The memory-based approach, known as population-based ACO (P-ACO) [6], maintains a population-list (memory), which stores the best ant of every iteration, and is used to generate the pheromone trails. When the change affects the solutions stored in the memory, they are repaired heuristically [8]. Immigrants schemes enable the algorithm to maintain the diversity of the population, by introducing new individuals into the population-list [5, 11].

In this paper, a hybrid memory-based immigrants scheme is proposed where immigrant ants are generated using a memory that stores the best solutions found in previous environments, called memory-based immigrants ACO (MI-ACO). The algorithm is a variation of the P-ACO, where memory-based immigrants replace the worst ants in the population-list, and is applied to the DTSP. The environmental changes are applied in such a way as to represent traffic jams over 24 hours. For example, during rush hour times, the traffic factor is high whereas during evening times it is low. The key idea of MIACO is to use the best ant from the memory as the base to generate immigrants. As a result, valuable knowledge is transferred to the pheromone trails that influence ants to move directly to a previous environment which is similar with the new one.

The rest of the paper is organized as follows. Section 2 describes the problem we try to solve, i.e., the DTSP with a cyclic environment. Section 3 describes the standard ACO (S-ACO) and P-ACO algorithms for the DTSP. Section 4 describes our proposed approach where we incorporate memory-based immigrants to P-ACO. Section 5 describes the experiments carried out by comparing MI-ACO with S-ACO and P-ACO. Finally, Section 6 concludes this paper with directions for future work.

2 DTSP with Cyclic Traffic Jams

The TSP is a well-known *NP*-hard 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.

The TSP becomes more challenging and realistic if it is subject to a dynamic environment. There are different variations of the DTSP such as changing the topology of cities by replacing cities [6, 7, 11], and changing the distances between cities by adding traffic factors to the links between cities [4]. In this paper, we

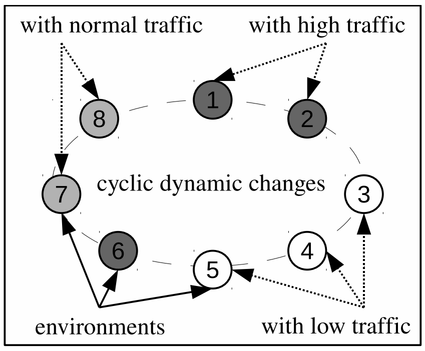

Fig. 1. Illustration of a cyclic dynamic environment with 8 base states. Each node represents a different environment where white, light grey and dark grey, represents low, medium and high traffic jams, respectively

generate a different variation of the DTSP with traffic factor, in which the dynamic changes occur with a cyclic pattern, as illustrated in Fig. 1. In other words, previous environments will appear again in the future. Such environments are more realistic since they represent a 24-hour traffic jam situation. For example during rush hour periods the traffic is high whereas during evening hours it is normal. The dynamics of the proposed DTSP are generated as described below.

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. 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 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.

A cyclic environment is constructed by generating different dynamic cases with traffic factor as the base states, representing DTSP environments with either low, normal or high traffic. For example a dynamic case with high traffic is constructed by assigning values closer to F_U a higher probability to be generated. Then, the environment cycles among these base states in a fixed logical ring.

3 ACO for the DTSP

3.1 Standard ACO

The S-ACO algorithm consists of a population of μ ants and it is based on the best performing ACO, i.e., Max-Min AS (MMAS) [13]. Initially, all ants are placed on a randomly selected city for a TSP 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, an ant k chooses the next city j while being on city 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} is the existing pheromone trail between cities i and j, η_{ij} is the heuristic information available a priori, which is defined as $1/D_{ij}$, where D_{ij} is the distance travelled (including F_{ij}) between cities i and j, N_i^k denotes the neighbourhood of cities of ant k when being on city 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, the 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. However, before adding any pheromone, a constant amount of pheromone is deducted 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. Reducing the pheromone values enables the population to forget bad decisions made in previous iterations [3]. This is important for ACO in order to adapt effectively to a new environment. After evaporation, the best ant deposits pheromone to the corresponding trails of its tour 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 cost of the tour T^{best}. Note that 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 [13].

3.2 Population-Based ACO

The P-ACO algorithm is the memory-based version of an ACO algorithm [8]. It differs from the S-ACO algorithm described above, since it follows a different framework. The algorithm maintains a memory (population-list) of ants, 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. P-ACO uses a more aggressive pheromone mechanism to forget bad solutions from previous environments than the pheromone evaporation used in the S-ACO.

On every iteration the population-list is updated using a strategy based on the *Age* of the ants. For the first K iterations, the iteration-best ant deposits a constant amount of pheromone using Eq. (2), where $\Delta\tau_{ij}^{best} = (\tau_{max} - \tau_{init})/K$. Here, τ_{max} and τ_{init} denote the maximum and initial pheromone amount, respectively. This positive update procedure is performed whenever the 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 thus, a negative update to its pheromone trails is done.

The population-list is a long-term memory, denoted k_{long}, since it contains ants from previous environments that survive in more than one iteration. Therefore, when a dynamic change occurs, the solutions stored in k_{long} are re-evaluated or repaired since their phenotype or genotype will be affected, and the pheromone trails are updated accordingly.

4 Memory-Based Immigrants ACO for the DTSP

Memory-based immigrants have been found beneficial for Genetic Algorithms (GAs) in DOPs, especially with a cyclic environment [14, 15]. Useful solutions are stored into a memory and used in the future since old environments will re-appear. When addressing DTSPs, S-ACO algorithms cannot adapt well to the environmental changes once the ants reach stagnation behaviour. The algorithm loses its adaption capability since it does not maintain diversity within the population. On the other hand, P-ACO is developed especially for DTSPs, but the stagnation behaviour remains unsolved since identical ants may be stored in the memory and generate high intensity of pheromone to a single trail.

However, considering that P-ACO is a memory-based approach, it may be beneficial in cyclic environments since it may guide the population into an old environment that is similar to the new one using good solutions from k_{long}. Also, it may be beneficial in slowly and slightly changing environments in order to have time to store good solutions in k_{long} that can be used later on. Similarly, the S-ACO algorithm may be beneficial in slowly changing environments because the population in S-ACO needs sufficient time to adapt to the new environment. The time needed depends on the magnitude of change. For a small magnitude, the population will adapt quickly since the previous environment will be similar with the new one.

Other immigrants schemes have been successfully applied to P-ACO algorithms to solve the DTSP [11]. Immigrant ants are generated to the population-list to maintain a certain level of diversity in the population and enhance its dynamic performance. However, a short-term memory, denoted k_{short}, is used instead of k_{long}, where the ants of the current iteration replace the ants of the previous iteration. Moreover, a number of immigrants are generated and replace the worst ants in k_{short} on every iteration. The advantages of using k_{short} are closely related to the survival of ants in a dynamic environment, where no ant can survive in more than one iteration. The proposed MIACO is another variation of the framework described above.

The only difference is that the MIACO consists of both k_{short} and k_{long}, where the first type of memory is updated and used as described above, and the second type of memory is updated by replacing the closest ant in the memory with the best-so-far ant whenever there is a dynamic change. The metric to define how close is ant i to ant j is defined as $M_{ij} = 1 - \frac{CE_{ij}}{n}$, where CE_{ij} is defined as the number of common edges between the ants and n is the number of cities. A value M_{ij} closer to 0 means that the ants are similar. Note that the update strategy of k_{long} in MIACO is different from P-ACO regarding which ant to replace and when to replace it. However, when a dynamic change occurs the ants in k_{long} are re-evaluated in order to be valid with the new environment as in the P-ACO algorithm.

Every iteration the best ant from k_{long} is selected in order to generate the memory-based immigrants, using inversions based on the inver-over operator [9], and replace the worst ants in k_{short}. MIACO inherits the advantages of both the memory scheme to guide the population directly to an old environment already visited and the immigrants scheme to maintain diversity. It is very important to store different solutions in the k_{long} which represent different environments that might be useful in the future. The key idea behind MIACO is to provide guided diversity into the pheromone trails in order to avoid the disruption of the optimization process.

5 Simulation Experiments

5.1 Experimental Setup

In the experiments, we compare the proposed MIACO with P-ACO and a S-ACO, which are described in Section 3. All the algorithms have been applied to the `eil76`, `kroA200`, and `att532` problem instances, obtained from TSPLIB[3].

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 [6, 11]. For all algorithms, $\mu = 25$ ants are used, $\alpha = 1$ and $\beta = 5$. For S-ACO, $q_0 = 0.0$, and $\rho = 0.2$. For P-ACO, $q_0 = 0.9$, $\tau_{max} = 1.0$, and the size of k_{long} is 3. For MIACO, $q_0 = 0.0$, and the size of k_{long} and k_{short} is 4 and 10, respectively. Moreover, 4 immigrant ants are used to replace the worst ants in k_{short}. For each algorithm on a DTSP instance, $N = 30$ independent runs were executed on the same cyclic environmental changes. The algorithms were executed for $G = 1000$ iterations and the overall offline performance is calculated as follows:

$$\boldsymbol{P}_{offline} = \frac{1}{G} \sum_{i=1}^{G} (\frac{1}{N} \sum_{j=1}^{N} P^{*}_{ij}) \qquad (3)$$

where P^{*}_{ij} defines the tour cost of the best ant since the last dynamic change of iteration i of run j [10]. Our implementation closely follows the guidelines of the ACOTSP[4] framework.

The value of f was set to 20 and 100, which indicate fast and slowly changing environments, respectively. The value of m was set to 0.10, 0.25, 0.50, and 0.75, which indicate the degree of environmental changes from small, to medium, to

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

Table 1. Comparison of algorithms regarding the results of the offline performance

Alg. & Inst.	`eil76`							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75
S-ACO	399.7	449.6	529.2	788.9	381.2	420.3	490.0	738.4
P-ACO	392.7	445.0	529.4	778.7	384.7	427.9	499.8	742.6
MIACO	393.8	440.0	521.6	771.4	385.8	424.5	495.3	737.0
Alg. & Inst.	`kroA200`							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75
S-ACO	26279.2	28756.2	35838.2	51150.5	23673.1	26010.7	32468.6	46579.3
P-ACO	24113.8	28096.6	35274.6	50378.2	23457.4	26359.0	32436.6	46612.2
MIACO	23813.7	27427.8	34183.7	48436.8	23290.0	25892.7	31691.3	45152.8
Alg. & Inst.	`att532`							
	$f = 20$				$f = 100$			
$m \Rightarrow$	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75
S-ACO	48987.0	53527.5	65462.0	88449.1	45489.7	49523.1	59879.9	81343.8
P-ACO	47677.2	53835.9	66229.8	90773.7	45371.8	50125.3	60810.9	82603.9
MIACO	46558.6	51885.1	63347.5	85608.3	44223.2	48324.8	58256.8	78643.0

Table 2. Statistical tests of comparing algorithms regarding the offline performance, where "−" or "+" means that the first algorithm is significantly better or the second algorithm is significantly better, respectively, and "∼" indicates no significance

Alg. & Inst.	`eil76`				`kroA200`				`att532`			
$f = 20, m \Rightarrow$	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75
S-ACO ⇔ P-ACO	+	+	∼	+	+	+	+	+	+	−	−	−
MIACO ⇔ P-ACO	+	−	−	−	−	−	−	−	−	−	−	−
MIACO ⇔ S-ACO	−	−	−	−	−	−	−	−	−	−	−	−
$f = 100, m \Rightarrow$	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75
S-ACO ⇔ P-ACO	−	−	−	−	+	−	∼	∼	∼	−	−	−
MIACO ⇔ P-ACO	+	−	−	−	−	−	−	−	−	−	−	−
MIACO ⇔ S-ACO	+	+	+	∼	−	−	−	−	−	−	−	−

large, respectively. Each environment has 4 cyclic base states and $F_L = 0$ 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 TSP instance, as described in Section 2, to systematically analyze the adaptation and searching capability of each algorithm on the DTSP.

5.2 Experimental Results and Analysis

The experimental results regarding the offline performance of the algorithms are presented in Table 1 and the corresponding statistical results of two-tailed t-test with 58 degrees of freedom at a 0.05 level of significance are presented in Table

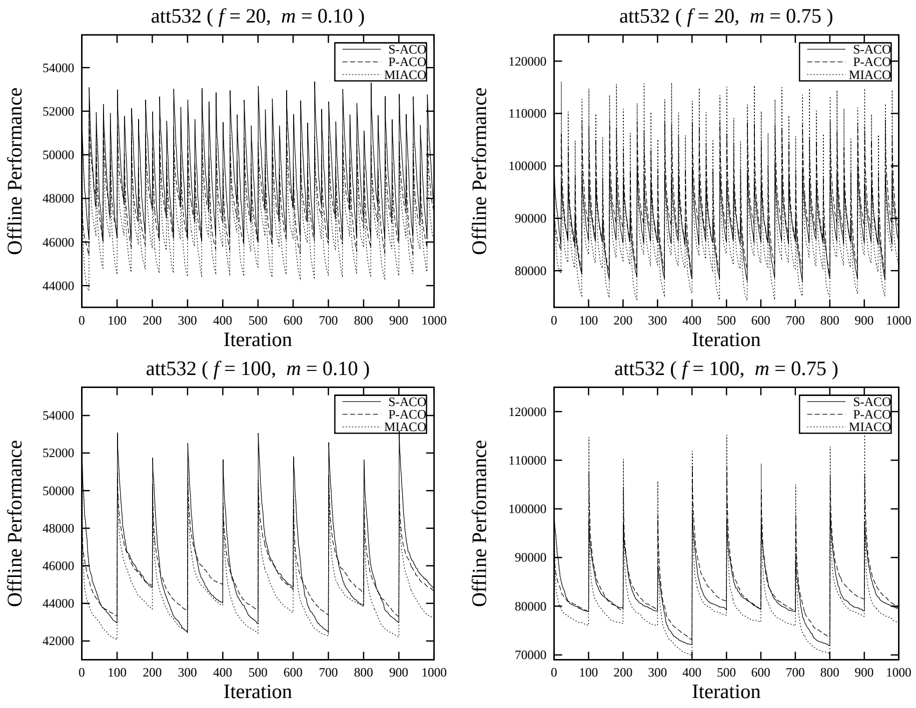

Fig. 2. Offline performance of algorithms for different dynamic test problems

2. Moreover, to better understand the dynamic behaviour of the algorithms, the results of the largest problem instance, i.e., `att532`, are plotted in Fig. 2 with $f = 20$, $m = 0.10$ and $m = 0.75$, and $f = 100$, $m = 0.10$ and $m = 0.75$, respectively. From the experimental results, several observations can be made by comparing the behaviour of the algorithms.

First, P-ACO outperforms S-ACO in almost all fast changing environments whereas it is beaten in almost all slowly changing environments; see the results of S-ACO $\Leftrightarrow$ P-ACO in Table 2. This validates our expectation that the S-ACO algorithm needs sufficient time to recover and converge to a new optimum when a dynamic change occurs, which can be observed from Fig. 2. This is because it uses only pheromone evaporation to eliminate pheromone trails that are not useful to the new environment. On the other hand, P-ACO has better performance in fast changing environments because previous pheromone trails are removed directly. Moreover, P-ACO is comparable with S-ACO in cases where $m = 0.10$ because the solutions stored in the memory from the previous environment are still fit to the new environment, since the environments are similar.

Second, the proposed MIACO outperforms P-ACO in almost all dynamic test environments as expected; see the results of MIACO $\Leftrightarrow$ P-ACO in Table 2. However, on some dynamic problem instances when $m = 0.10$, i.e., `eil76`, MIACO is beaten by P-ACO. This is because MIACO generates high levels of diversity which may not be beneficial when the environment changes slightly. However, on the largest problem instances, i.e., `kroA200` and `att532`, MIACO is significantly better in all test cases, where diversity is needed.

Third, the proposed MIACO outperforms S-ACO in all fast changing dynamic test cases; see the results of MIACO $\Leftrightarrow$ S-ACO in Table 2. However, in some slowly changing environments, i.e., `eil76`, which is the smallest problem instance, S-ACO outperforms MIACO. This is natural since it is easier for the population in S-ACO to eliminate unused pheromone trails from previous environments and become more adaptive. As the magnitude and problem size increases MIACO is significantly better than S-ACO as expected.

Finally, memory-based schemes are useful in cyclic dynamic environments since they are able to move the population directly to a previously visited environment. MIACO stores the best solutions for all cyclic states in its k_{long}, whereas P-ACO stores solutions only from the previous state since its k_{long} is updated on every iteration. Therefore, P-ACO is beneficial in cases where the changing environments are similar and the use of the solutions stored in the memory only from the previous environment are useful which can be observed from Table 2. On the other hand, MIACO can guide the population directly to any old environment visited that will re-appear in the future which can be observed from Fig. 2.

6 Conclusions

Memory-based immigrants have been successfully applied to GAs to address different DOPs [14, 15]. In this paper, we incorporate memory-based immigrants into ACO, denoted MIACO, for the DTSP under cyclic environmental changes. The immigrant ants are generated using the best ant of the memory and replace the worse ones in the population. It combines the merits of memory and immigrants schemes, where the first one is able to move the population into a previously visited environment directly, and the second one is able to maintain the diversity of solutions in order to adapt well in DOPs.

Comparing MIACO with a traditional S-ACO and P-ACO, a variation designed for DOPs, on different test cases of DTSPs, the following concluding remarks can be drawn. First, memory-based immigrants are advantageous for ACO algorithms in cyclic dynamic environments, since MIACO is significantly better than S-ACO and P-ACO in almost all dynamic test cases. Second, increasing the diversity of ACO is not always beneficial in DTSPs. Second, P-ACO is comparable with S-ACO in most slightly changing environments. Finally, P-ACO is significantly better than S-ACO in fast changing environments, while it is significantly worse in slowly changing environments.

In fact, MIACO may be also beneficial in random dynamic environments as the elitsm-based immigrants ACO (EIACO) [11], since both algorithms transfer knowledge from previous environments, and EIACO may be also beneficial in some cyclic dynamic environments. Generally, transferring the knowledge found in previous environments to the pheromone trails, helps ACO algorithms to adapt well in DOPs. Therefore, for future work, it would be interesting to apply MIACO on DTSPs with random dynamic environments or with different dynamic changes, e.g., replacing cities, and compare it with other peer ACOs.

Acknowledgement

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. on Syst., Man and Cybern., Part B: Cybern. 26(1), 29–41 (1996)
3. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, London (2004)
4. Eyckelhof, C. J., Snoek, M.: Ant Systems for a Dynamic TSP. In: ANTS 2002: Proc. of the 3rd Int. Workshop on Ant Algorithms, pp. 88–99 (2002)
5. Grefenestette, J. J.: Genetic algorithms for changing environments. In: Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)
6. Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: Proc. of the 3rd Int. Workshop on Ant Algorithms, LNCS 2463, pp. 111–122 (2002)
7. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: EvoWorkshops 2001: Appl. of Evol. Comput., pp. 213-222 (2001)
8. Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Proc. of the 2001 Genetic and Evol. Comput. Conf., pp. 860–867 (2001)
9. Guo, T., Michalewicz, Z.: Inver-over operator for the TSP. In: Proc. of the 5th Int. Conf. on Parallel Problem Solving from Nature, pp. 803–812 (1998)
10. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. on Evol. Comput. 9(3), 303–317 (2005)
11. Mavrovouniotis, M., Yang, S.: Ant colony optimization with immigrants schemes for dynamic environments. In: Proc. of the 11th Int. Conf. on Parallel Problem Solving from Nature, pp. 371–380 (2010)
12. 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), pp. 135–151 (2007)
13. Stützle, T., Hoos, H.: The MAX-MIN ant system and local search for the traveling salesman problem. In: Proc. of the 1997 IEEE Int. Conf. on Evol. Comput., pp. 309–314 (1997)
14. Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proc. of the 2005 Genetic and Evol. Comput. Conf., vol. 2, pp. 1115–1122 (2005)
15. Yang, S.: Genetic algorithms with memory and elitism based immigrants in dynamic environments. Evol. Comput. 16(3), 385–416 (2008)