Ant Colony Optimization with Self-Adaptive Evaporation Rate 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*—The performance of ant colony optimization (ACO) algorithms in tackling optimization problems strongly depends on different parameters. One of the most important parameters in ACO algorithms when addressing dynamic optimization problems (DOPs) is the pheromone evaporation rate. The role of pheromone evaporation in DOPs is to improve the adaptation capabilities of the algorithm. When a dynamic change occurs, the pheromone trails of the previous environment will not match the new environment especially if the changing environments are not similar. Therefore, pheromone evaporation helps to eliminate pheromone trails that may misguide ants without destroying any knowledge gained from previous environments. In this paper, a self-adaptive evaporation mechanism is proposed in which ants are responsible to select an appropriate evaporation rate while tracking the moving optimum in DOPs. Experimental results show the efficiency of the proposed self-adaptive evaporation mechanism on improving the performance of ACO algorithms for DOPs.**

I. Introduction

Ant colony optimization (ACO) algorithms have proved to be powerful methods to address challenging combinatorial optimization problems [3], [4], [8], [9], [22]. So far, most problems addressed by ACO have stationary environments where the problems remain fixed during the optimization process. However, many real-world problems are dynamic optimization problems (DOPs) where the objective function, decision variables, problem instance, constraints, and so on, may change during the execution of an algorithm [20], [26]. For DOPs, algorithms need to track the moving optimum rather than simply converging to a static optimum.

When a dynamic change occurs, it may take some time for an ACO algorithm to adapt to the new environment [1], [16]. This is because ACO algorithms are designed especially to converge into a single optimum [2], [3]. Therefore, once the population converges into an optimum, it will be difficult to escape from the optimum and track the changing optimum. A simple strategy to address this issue is to re-initialize the pheromone trails whenever a dynamic change occurs [16]. However, this strategy may be inefficient, especially in cases where the changing environments are similar [12]. This is because the knowledge gained from the previous optimization will not be available to speed up re-optimization. Over the years, several strategies have been proposed to enhance the performance of ACO algorithms for DOPs [20], including maintaining diversity strategies [5], [10], [16], increasing diversity via immigrants [15], [17], memory-based schemes [11], multi-colony schemes [19] and memetic algorithms [13].

The performance of an ACO algorithm depends on the selection of its parameters' values [6]. The algorithmic parameters are typically kept fixed during the execution of the algorithm to solve an optimization problem, either with stationary or dynamic environments. Modifying the parameters during the execution of the algorithm showed promising performance in stationary optimization problems. There are three main methods of modifying parameters during the execution of an algorithm: pre-scheduled, adaptive and self-adaptive (a comprehensive survey is available in [25]). Such methods have attracted less attention for DOPs.

The adaptation capabilities of ACO algorithms on DOPs rely on the evaporation rate where a constant amount of pheromone is deducted from all trails. The pheromone evaporation helps to eliminate the trails of solutions that may bias ants to search to non-promising areas of the search space. Since the optimal evaporation rate depends on the magnitude of change, an adaptive evaporation rate was recently proposed in [16]. More precisely, the evaporation rate is modified according to some rules that consider the search behaviour of the ACO algorithm. However, this method introduces two new parameters, e.g., the initial evaporate rate and the evaporation step size, that may affect the performance of ACO algorithms.

In this paper, a self-adaptive evaporation rate is proposed where the evaporation rate is integrated into the ACO search task. This way, there is no need to select a value for the initial evaporation rate or the evaporation step size as in the adaptive method. More precisely, ants are responsible to select the appropriate evaporation rate during the optimization process of a DOP. The dynamic benchmark generator proposed in [18] is used to generate a series of dynamic test cases and experiments are systematically conducted. The experiments show that ACO with a self-adaptive evaporation rate outperforms ACO with adaptive or fixed evaporation rates on most dynamic test cases.

The rest of the paper is organized as follows. Section II describes the DOPs generated by the benchmark generator. Section III describes one of the state-of-the-art ACO algorithm, which is used for the experiments. Section IV describes the the self-adaptive method used to tune the evaporation rate. Section V gives the experimental results of comparing an ACO

algorithm with fixed, adaptive, and self-adaptive evaporation rates, respectively. Finally, Section VI concludes this paper with discussions on future work.

II. Dynamic Test Environments

A. Travelling Salesman Problem (TSP)

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 and only once before returning to the starting city.

Formally, the TSP is defined as follows. Let ψ_{ij} denote the binary decision variables defined as follows:

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

where $\psi_{ij} \in \{0, 1\}$. Then, the objective of the TSP is defined as follows:

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

where n is the number of cities and d_{ij} is the distance between cities i and j.

B. Dynamic Benchmark Generators

Over the years, several dynamic benchmark generators have been proposed for the TSP that tend to model real-world scenarios, such as the dynamic TSP (DTSP) with traffic factors [5], [17], [19] and the DTSP with exchangeable cities [10], [11]. These benchmark generators modify the fitness landscape whenever a dynamic change occurs and cause the optimum value to change.

In this paper, the dynamic benchmark generator for permutation-encoded problems (DBGP) proposed in [18] is used, which can convert any stationary permutation-encoded benchmark problem instance into a DOP. The fitness landscape is not modified with DBGP, and thus, the optimum value (if known) remains the same. This is because DBGP shifts the population of the algorithm to search to a new location in the fitness landscape. The main advantage of using the DBGP rather than other generators is that one can observe how close to the optimum an algorithm can perform when a change occurs.

C. Constructing Dynamic Test Environments

The TSP is typically presented by a weighted graph. Let $G = (N, A)$ be a weighted graph where $N = \{1, \ldots, n\}$ is a set of n cities (nodes) and $A = \{(i, j) : i \neq j\}$ is a set of links (arcs). Each city i has a location defined by (x, y) and each link (i, j) 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 a dynamic test 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 $\vec{V}(T)$ city locations are swapped. Then, a randomly re-ordered vector $\vec{U}(T)$ is generated that contains only the cities of $\vec{V}(T)$. Therefore, exactly $m \times n$ pairwise swaps are performed in $\vec{D}$ using the two random vectors $(\vec{V}(T) \otimes \vec{U}(T))$, where "$\otimes$" denotes the swap operator.

III. ACO in Dynamic Environments

A. Description of ACO

ACO algorithms were initially developed to address routing problems modelled with weighted graphs [3], [9]. Within ACO, a population of μ ants construct feasible solutions on their forward mode, and update pheromone trails on their backward mode every iteration. Hence, after a few iterations high concentrations of pheromone will be generated into a trail that corresponds to a solution of a routing problem, e.g., the TSP.

In this paper, we consider one of the best performing ACO algorithm, i.e., the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [23], [24]. Each ant k constructs a feasible solution 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, d_{ij} is the distance between cities i and j, and $\mathcal{N}_i^k$ denotes the cities that ant k is allowed to select from city i. $\mathcal{N}_i^k$ is generated by the unvisited cities incident to city i.

The pheromone trails in $\mathcal{MM}$AS are updated first by applying evaporation as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\, \tau_{ij}, \forall (i, j), \tag{4}$$

where ρ is the evaporation rate which satisfies $0 < \rho \leq 1$, and τ_{ij} is the existing pheromone value. After evaporation, the best ant deposits pheromone as follows:

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

where $\Delta \tau_{ij}^{best} = 1/C^{best}$ is the amount of pheromone that the best ant deposits and C^{best} defines the solution quality of tour T^{best}. The best ant that is allowed to deposit pheromone may be either the best-so-far[1], in which case $C^{best} = C^{bs}$, or the iteration-best, in which case $C^{best} = C^{ib}$, where C^{bs} and C^{ib} are the solution quality of the best-so-far and iteration best ants, respectively. Both update rules are used in an alternate way in the implementation [24].

The lower and upper limits τ_{min} and τ_{max} of the pheromone values are imposed. The τ_{max} value is initially bounded by $1/(\rho C^{bs})$, where C^{bs} is initially the solution quality of an estimated optimal tour and later on is updated whenever a new best-so-far ant solution quality is found. The τ_{min} value is set to $\tau_{min} = \tau_{max}/a$, where a is a parameter.

Since only the best ant deposits pheromone, the population may quickly converge towards the best solution found in the

[1]The best-so-far ant may not necessarily belong to the current ant population.

first iteration. Therefore, the pheromone trails are occasionally re-initialized to the τ_{max} value to increase exploration. For example, whenever stagnation behaviour[2] occurs or when no improved solution is found for a given number of iterations, the pheromone trails are re-initialized.

B. Response to Dynamic Changes

ACO algorithms can transfer knowledge via the pheromone trails generated from the previous environment to the newly generated environment. If the dynamic change is severe, the pheromone trails of the previous environment may misguide the population to search into areas far from the optimum of the new environment; otherwise, if the dynamic change is small, the pheromone trails of the previous environment may provide valuable knowledge to speed up the re-optimization process to the new environment.

ACO algorithms face a serious challenge with stagnation behaviour when addressing DOPs. Once the population converges quickly into an optimum, then it is difficult for the population to escape from the old optimum in order to adapt to the new optimum when an environmental change occurs. More precisely, the pheromone trails generated around the optimum before the dynamic change may misguide ants not to track the moving optimum. Therefore, ACO algorithms may need some time to adapt to dynamic changes because of the stagnation behaviour.

The adaptation via pheromone evaporation may be a sufficient choice when the changing environments are similar; otherwise, a complete re-initialization of the pheromone trails after a dynamic change occurs may be a better choice [16]. However, such action is available only for DOPs where the frequency of change is known beforehand or DOPs where the dynamic changes can be detected. In our case, the dynamic changes can be detected by re-evaluating some stored solutions, which are used as detectors, in every iteration [14].

IV. Self-Adaptive Evaporation Rate

A. Adaptation via Pheromone Evaporation

Once an ACO algorithm converges into an optimum, it loses its adaptation capabilities. This is because the pheromone trails generated to the optimum before a dynamic change may misguide ants not to track the newly generated optimum after the dynamic change. Pheromone evaporation may help to eliminate unused pheromone trails and help the population escape from the previously converged optimum.

Although ACO has adaptation capabilities due to the pheromone evaporation, the time required to adapt to a new environment depends on the problem size and the magnitude of change [16]. The evaporation rate defined in Eq. 4 is fixed in conventional ACO algorithms. A low evaporation rate corresponds to slow adaptation, whereas a high evaporation rate corresponds to fast adaptation. Therefore, it is straightforward that a higher evaporation rate is more suitable in both quickly and severely changing environments, whereas a lower evaporation rate is more suitable in slightly changing environments, which has been empirically investigated in [16].

[2]A term used when all ants construct the same solution from early stages of the optimization process.

B. Adaptive Evaporation Rate

At different stages of the optimization process for different optimization problems and under different dynamic environments, the most appropriate evaporation rate varies. In [16], the evaporation rate has been adapted according to the status of the algorithm. In case the algorithm reaches the stagnation behaviour, the evaporation rate increases in order to eliminate the high intensity of pheromone trails in some areas and increase exploration; otherwise, it decreases to allow the population to use the knowledge transferred from previous environments.

According to the behaviour of the algorithm in terms of searching, the following pheromone evaporation rate update rule was proposed:

$$\rho(t) = \begin{cases} \rho(t-1) - \sigma, & \text{if } \bar{\lambda}(t) > 1, \\ \rho(t-1) + \sigma, & \text{otherwise}, \end{cases} \tag{6}$$

where $\bar{\lambda}(t)$ is the λ-branching factor [7] and σ is the step size of varying the evaporation rate ρ at iteration t. A high value of σ may quickly increase the evaporation rate to an extreme evaporation rate and destroy information whereas a small value may not have any effect on the performance of ACO (see Section V-C).

C. Self-Adaptive Evaporation Rate

The limitation of the adaptive method proposed in [16] is that the performance is dependent on the initial evaporation rate value $\rho(0)$ and the evaporation step size σ. In this paper, a self-adaptive method is proposed to address the aforementioned issues and enhance the performance of ACO for DOPs. The difference between adaptive and self-adaptive methods lies in that in the adaptive strategy the parameter is modified according to some rules that take into account the search behaviour of the algorithm whereas in the self-adaptive strategy the parameter is modified by the algorithm itself by integrating the parameter into its search task.

The key idea to self-adapt the evaporation rate is to split the value range of ρ into P discrete points that are generated randomly within the interval of $[0.0, 1.0]$. In addition, an extra pheromone table of the same size of the main pheromone table is maintained, which is attached with the discrete points [21]. The update policy of the extra pheromone table is the same with the one described in Section III.

More precisely, ants are responsible to select the ρ value when they construct their solutions at every iteration. When ants update their pheromone trails in the main pheromone table using the selected ρ as defined in Eqs. 4 and 5, the specific ant updates the pheromone trails in the extra pheromone table. A single ant is used to select the ρ value since it is a global algorithmic parameter. In case local algorithmic parameters, e.g., α and β, are considered, then each ant should select its own values.

V. Experimental Study

A. Experimental Setup

The proposed self-adaptive evaporation mechanism is integrated with the $\mathcal{MM}$AS algorithm, denoted as $\mathcal{MM}\text{AS}_S$

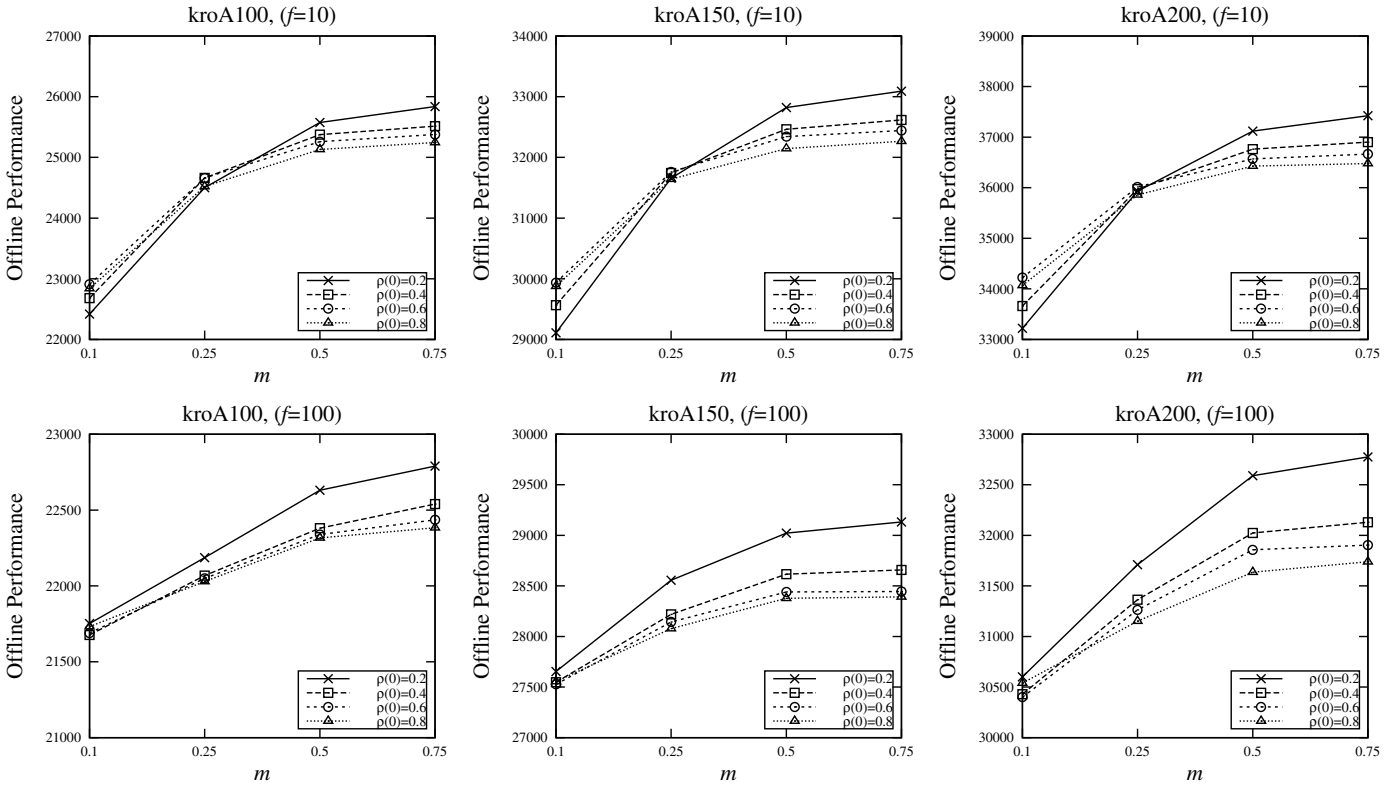

Fig. 1. Impact of the initial evaporation rate regarding the offline performance for $\mathcal{MM}\text{AS}_A$ on different DOPs.

in this paper, and is compared with an existing adaptive mechanism integrated to the same algorithm [16], denoted as $\mathcal{MM}\text{AS}_A$, and a traditional $\mathcal{MM}$AS with a fixed evaporation rate.

All common algorithmic parameters were set as follows: $\alpha = 1$, $\beta = 5$ and the population size $\mu = 50$. In order to investigate the effect of the initial evaporation rate and the evaporation step size on the performance of $\mathcal{MM}\text{AS}_A$, the parameters were set as $\rho(0) \in \{0.2, 0.4, 0.6, 0.8\}$ (see Section V-B) and $\sigma \in \{0.0001, 0.001, 0.005, 0.01\}$ (see Section V-C), respectively. For $\mathcal{MM}\text{AS}_S$, the number of discrete points was set to $P = 20$.

DOPs are generated from three stationary TSP benchmark instances taken from TSPLIB[3] using the DBGP[4] 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 DOPs are constructed from each stationary TSP benchmark instance.

For each ACO algorithm on a DOP, 30 independent runs were executed on the same set of random seeds. 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 [12] of an ACO on a DOP is defined as follows:

$$\bar{P}_{OFF} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R}P_{ij}^{*}\right), \tag{7}$$

where E is the number of iterations, R is the number of runs, and P_{ij}^{*} defines the tour cost of the best ant of iteration i of run j.

B. Effect of Parameter $\rho(0)$ in $\mathcal{MM}AS_A$

The offline performance results of $\mathcal{MM}\text{AS}_A$ [16] with different initial evaporation rate values are plotted in Fig. 1 for all DOPs. It can be seen that when $\rho(0) = 0.2$ the offline performance of the algorithm is better than when $\rho(0) > 0.2$ on DOPs with $f = 10$ and $m = 0.1$. In contrast, when $\rho(0) = 0.4$ the performance of the algorithm improves on DOPs with $f = 100$ and $m = 0.1$. When $\rho(0) > 0.4$, the performance of the algorithm achieves better results on the remaining DOPs. The results support our claim above that the performance of $\mathcal{MM}\text{AS}_A$ is dependent on the initial evaporation rate value.

In general, these results match the findings in [16] where the performance of the traditional $\mathcal{MM}$AS with a fixed evaporation rate depends on the ρ value. When the environment changes slightly (e.g., $m = 0.1$ or $m = 0.25$), a lower evaporation rate achieves better performance whereas when the environment changes severely (e.g., $m = 0.5$ or $m = 0.75$), a higher evaporation achieves better performance.

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

[4]Available from www.tech.dmu.ac.uk/mmavrovouniotis/Codes/DBGP.zip

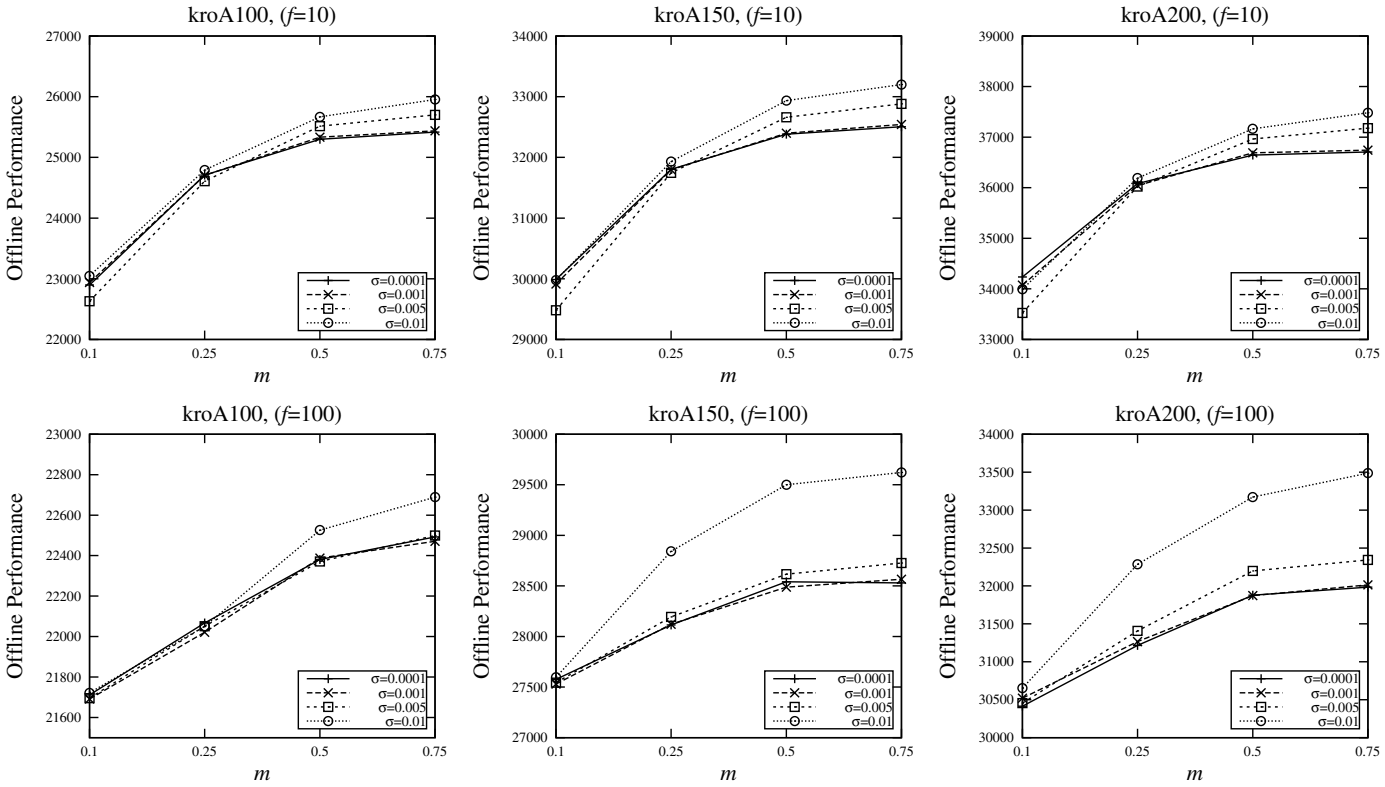

Fig. 2. Impact of the evaporation step size regarding the offline performance for $\mathcal{MM}\text{AS}_A$ on different DOPs.

TABLE I. EXPERIMENTAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF ACO ALGORITHMS. BOLD VALUES INDICATE THE BEST RESULTS

Travelling Salesman Problem												
Algorithms & DOPs	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
$\mathcal{MM}\text{AS}(\rho = 0.2)$	**22448**	24542	25486	25722	**29105**	31609	32686	32921	**33175**	35900	37017	37207
$\mathcal{MM}\text{AS}(\rho = 0.8)$	22802	24698	25334	25480	29745	31762	32422	32578	33896	35987	36687	36802
$\mathcal{MM}\text{AS}_A$	22939	24704	25334	25439	29914	31796	32399	32545	34074	36044	36691	36744
$\mathcal{MM}\text{AS}_S$	22746	**24367**	**24888**	**24991**	29804	**31341**	**31901**	**31965**	33835	**35567**	**36033**	**36223**
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}(\rho = 0.2)$	**21685**	22162	22567	22692	**27354**	28446	28860	28885	**30311**	31618	32444	32528
$\mathcal{MM}\text{AS}(\rho = 0.8)$	21706	22054	22395	22526	27493	28189	28569	28653	30478	32329	31981	32077
$\mathcal{MM}\text{AS}_A$	21691	22021	22388	22471	27530	28121	28488	28566	30520	31267	31872	32014
$\mathcal{MM}\text{AS}_S$	21688	**22124**	**22286**	**22320**	27623	**28033**	**28313**	**28411**	30526	**31185**	**31675**	**31641**

C. Effect of the σ Parameter in $\mathcal{MM}AS_A$

The offline performance results of $\mathcal{MM}\text{AS}_A$ [16] with different evaporation step size values are plotted in Fig. 2 for all DOPs. Note that the initial evaporation rate was set to $\rho(0) = 0.5$.

When the evaporation step size is lower (e.g., $\sigma \leq 0.001$), the offline performance of the algorithm is better than when the evaporation step size is higher (e.g., $\sigma \geq 0.005$) on DOPs with $f = 100$ and with $m = 0.5$ and $m = 0.75$. In contrast, the offline performance of the algorithm is competitive with different σ values in DOPs with $f = 100$ and with $m = 0.1$ and $m = 0.25$. When $\sigma \geq 0.005$ the offline performance of the algorithm is better than when σ is lower (e.g., $\sigma = 0.001$) in DOPs with $f = 10$ and $m = 0.1$ and $m = 0.25$ whereas when $\sigma \leq 0.001$ the offline performance of the algorithm is better than when σ is higher in DOPs with $f = 10$ and $m = 0.5$ and $m = 0.75$. This is natural because the initial evaporation rate is set to a high value, i.e., $\rho(0) = 0.5$, and a lower step will keep the evaporation rate to a high value. A higher evaporation rate achieves better performance in severely changing environments as described in Section V-B. Similarly, these results support our claim above that the performance of $\mathcal{MM}\text{AS}_A$ is also dependent on the evaporation step size value.

D. Self-adaptive vs Adaptive Evaporation Rate

The experimental results regarding the offline performance of the proposed $\mathcal{MM}\text{AS}_S$ algorithm compared with $\mathcal{MM}\text{AS}_A$ and $\mathcal{MM}$AS with a fixed evaporation rate for

TABLE II. STATISTICAL RESULTS REGARDING THE OFFLINE PERFORMANCE ACO ALGORITHMS

Travelling Salesman Problem												
Algorithms & DOPs	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
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}(\rho = 0.2)$	$-$	$+$	$+$	$+$	$-$	$+$	$+$	$+$	$-$	$+$	$+$	$+$
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}(\rho = 0.8)$	$\sim$	$+$	$+$	$+$	$\sim$	$+$	$+$	$+$	$\sim$	$+$	$+$	$+$
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}_A$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}(\rho = 0.2)$	$\sim$	$\sim$	$+$	$+$	$\sim$	$+$	$+$	$+$	$-$	$+$	$+$	$+$
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}(\rho = 0.8)$	$\sim$	$\sim$	$+$	$+$	$\sim$	$+$	$+$	$+$	$-$	$+$	$+$	$+$
$\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}_A$	$\sim$	$-$	$+$	$+$	$\sim$	$+$	$+$	$+$	$\sim$	$\sim$	$+$	$+$

all DOPs are presented in Table I. For $\mathcal{MM}\text{AS}_A$ the initial evaporation rate was set to $\rho(0) = 0.5$ and the evaporation step size was set to $\sigma = 0.001$ as in [16]. Two fixed evaporation rates were used for $\mathcal{MM}$AS, i.e., $\rho = 0.2$ and $\rho = 0.8$, indicating slow and fast adaptation, respectively, denoted as $\mathcal{MM}\text{AS}(\rho)$. 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 "$\sim$" 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. In Fig. 3, the dynamic offline performance for quickly and slowly changing environments against the algorithmic iterations of the algorithms are plotted to better understand the behaviour of the ACO algorithms. From the experimental results, several observations can be drawn.

$\mathcal{MM}\text{AS}_S$ significantly outperforms $\mathcal{MM}\text{AS}_A$ on most DOPs (except when $f = 100$ and $m = 0.1$); see the comparisons of $\mathcal{MM}\text{AS}_S \Leftrightarrow \mathcal{MM}\text{AS}_A$ in Table II. It can be observed from Fig. 3 that the proposed algorithm is able to maintain better performance than its competitor during the environmental changes.

When the environment changes slightly and slowly, the two algorithms are comparable. This is possibly because, when the environments are similar, a slow evaporation rate is required. However, the initial evaporation rate of $\Leftrightarrow \mathcal{MM}\text{AS}_A$ is set to $\rho(0) = 0.5$, which is considered as a high evaporation rate. Regarding $\mathcal{MM}\text{AS}_S$, different evaporation rate values, including large values, may be used until the ants converge to one (hopefully the optimum). This also supports the superior performance of $\mathcal{MM}\text{AS}_S$ against $\mathcal{MM}\text{AS}_A$ in severely changing environments. For example, $\mathcal{MM}\text{AS}_A$ performs always small constant adjustments to the evaporation according to the σ value used (i.e., $\sigma = 0.001$) whereas $\mathcal{MM}\text{AS}_S$ performs either small or large adjustments according to the different evaporation rate values selected.

E. Self-adaptive vs Fixed Evaporation Rate

Considering Tables I and II and Fig. 3, it can be observed that $\mathcal{MM}\text{AS}_S$ is significantly better than $\mathcal{MM}\text{AS}(\rho = 0.2)$ on most DOPs with $m = 0.25$, $m = 0.5$ and $m = 0.75$ whereas the latter is significantly better than the former on most DOPs with $m = 0.1$. This is because a slow evaporation rate is not the best choice when the environmental changes are severe. This can be observed from Table I, where $\mathcal{MM}\text{AS}(\rho = 0.8)$ achieves better performance than $\mathcal{MM}\text{AS}(\rho = 0.2)$ in severely changing environments whereas the former achieves better results than the latter in slightly changing environments. In fact, the results of $\mathcal{MM}\text{AS}(\rho = 0.2)$ and $\mathcal{MM}\text{AS}(\rho = 0.8)$ are the optimum fixed evaporation rates for environments with $m = 0.1$ and for environments with $m = 0.25$, $m = 0.5$ and $m = 0.75$, respectively.

In contrast, $\mathcal{MM}\text{AS}_S$ significantly outperforms $\mathcal{MM}\text{AS}(\rho = 0.8)$ on most DOPs. Although $\mathcal{MM}\text{AS}_S$ generally has good performance, it can be observed that it inherits the bad performance when fast evaporation rate (e.g., $\mathcal{MM}\text{AS}(\rho = 0.8)$) is used when the dynamic change is small. This is probably because most of the random numbers generated as discrete points represent a high value of evaporation rate. Therefore, previous knowledge is eliminated quickly from the pheromone trails that could be useful in changing environments that are similar (e.g., $m = 0.1$).

VI. CONCLUSIONS AND FUTURE WORK

The pheromone evaporation rate has a significant impact on the performance of ACO algorithms for DOPs. Different evaporation rates perform better on different DOPs. This paper introduces a self-adaptive evaporation rate where ants are responsible to select an appropriate value for the evaporation rate while tackling the problem. The evaporation rate was previously adapted and showed promising results in [16]. The main difference between the previous adaptive method and the self-adaptive method proposed in this paper lies in that in the former method the evaporation rate is modified according to some rules that consider the search behaviour of the ACO algorithm whereas in the latter method the evaporation rate is integrated into the ACO search task [25]. In the experiments, the ACO with a self-adaptive evaporation rate was compared with two ACO algorithms: one with the adaptive rate and another with a fixed evaporation rate. From the experimental results, several concluding remarks can be drawn.

First, the initial evaporation rate and evaporation step size of the adaptive evaporation method [16] affect the performance of the ACO algorithm. Second, the advantage of the self-adaptive evaporation method against the adaptive evaporation rate is that two parameters within the adaptive evaporation scheme are eliminated. Furthermore, the proposed self-adaptive evaporation method performs better than the existing

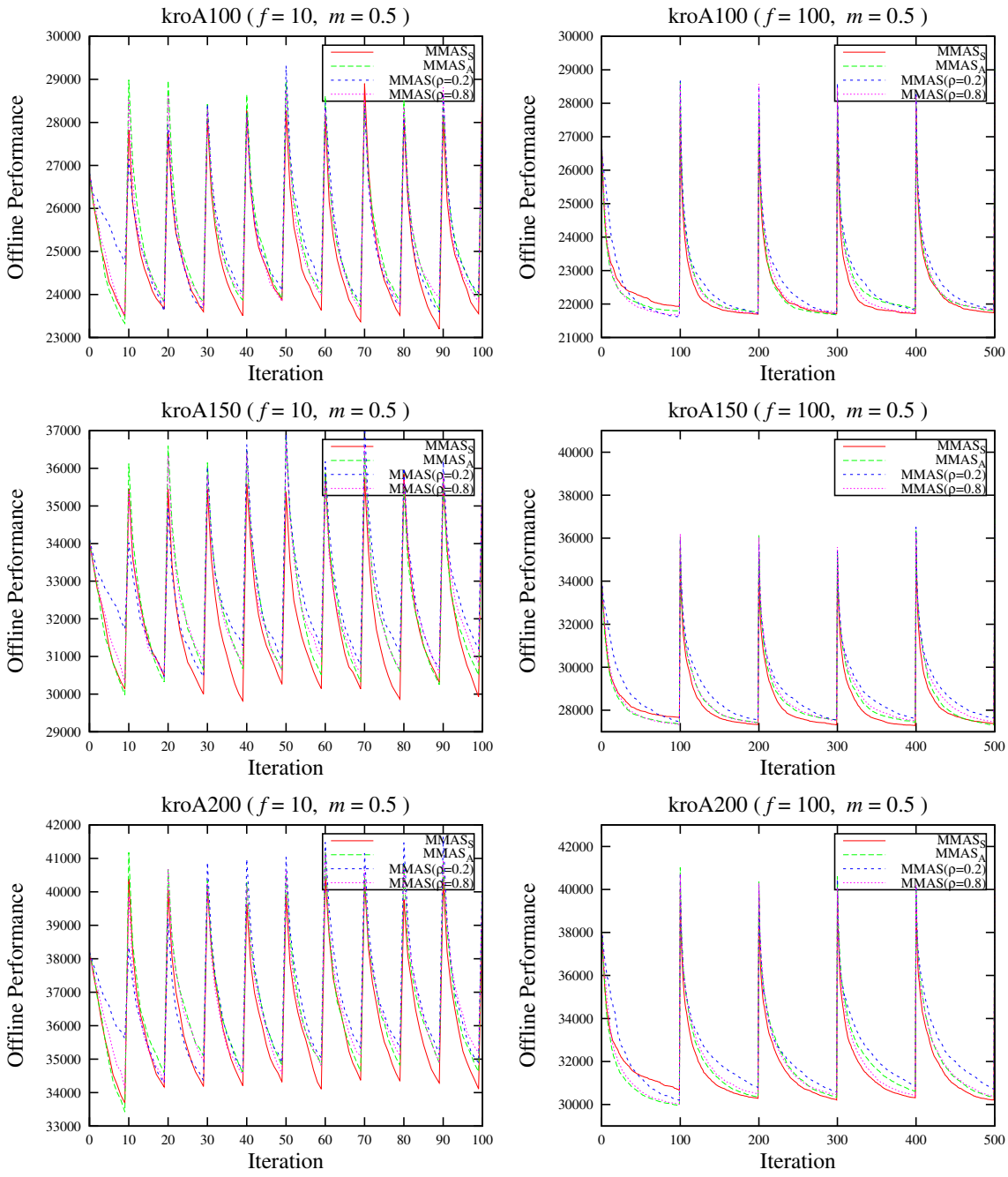

Fig. 3. Dynamic offline performance of ACO algorithms in different changing environments.

adaptive evaporation method in most dynamic test cases. This is because the adaptive method performs constant adjustments to the evaporation rate whereas the self-adaptive method performs adjustments according to the value selected probabilistically by ants. Third, a fixed evaporation rate seems a better choice when the environment changes slightly. This is because the adaptive or self-adaptive evaporation rate may reach high evaporation rates and destroy previous knowledge from the pheromone trails. In contrast, the self-adaptive evaporation rate seems a better choice when the environment changes severely.

For future work, it would be interesting to also self-adapt local ACO parameters (e.g., α and β) instead of global ACO parameters (e.g., ρ) when addressing DOPs using ACO algorithms. Furthermore, a better calibration of the discrete points of the self-adaptive method may improve the consistency on different dynamic test cases, e.g., achieve better performance on DOPs with $m = 0.1$.

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] D. Favaretto, E. Moretti, and P. Pellegrini, "On the explorative behavior of maxmin ant system," in *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, ser. Lecture Notes in Computer Science, T. Stützle, M. Birattari, and H. Hoos, Eds. Springer Berlin Heidelberg, 2009, vol. 5752, pp. 115–119.

[7] 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.

[8] L. M. Gambardella, E. D. Taillard, and C. Agazzi, "Macs-vrptw: A multicolony ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*, 1999, pp. 63–76.

[9] 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.

[10] 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.

[11] ——, "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.

[12] 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.

[13] M. Mavrovouniotis and S. Yang, "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, no. 7, pp. 1405–1425, 2011.

[14] ——, "Memory-based immigrants for ant colony optimization in changing environments," in *EvoApplications2011: Applications of Evolutionary Computation*, ser. LNCS, vol. 6624. Springer-Verlag, 2011, pp. 324–333.

[15] ——, "Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem," in *EvoApplications2012: Applications of Evolutionary Computation*, ser. LNCS, vol. 7248. Springer-Verlag, 2012, pp. 519–528.

[16] ——, "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.

[17] ——, "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.

[18] 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.

[19] 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.

[20] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[21] M. Randall, "Near parameter free ant colony optimisation," in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Sttzle, Eds. Springer Berlin Heidelberg, 2004, vol. 3172, pp. 374–381.

[22] 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.

[23] T. Stützle and H. Hoos, "The max-min ant system and local search for the traveling salesman problem," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*. IEEE Press, 1997, pp. 309–314.

[24] ——, "Max-min ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889 – 914, 2000.

[25] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, and M. Dorigo, "Parameter adaptation in ant colony optimization," in *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Springer Berlin Heidelberg, 2012, pp. 191–215.

[26] S. Yang, Y. Jiang, and T. T. Nguyen, "Metaheuristics for dynamic combinatorial optimization problems," *IMA Journal of Management Mathematics*, vol. 24, no. 4, pp. 451–480, 2013.