Pheromone Modification Strategy for the Dynamic Travelling Salesman Problem with Weight Changes

Michalis Mavrovouniotis
School of Science and Technology
Nottingham Trent University
Nottingham NG11 8NS, U.K.
email: michalis.mavrovouniotis@ntu.ac.uk

Mien Van
School of Science and Technology
Nottingham Trent University
Nottingham NG11 8NS, U.K.
email: mien.van@ntu.ac.uk

Shengxiang Yang
School of Computer Science and Informatics
De Montfort University
Leicester LE1 9BH, U.K.
email: syang@dmu.ac.uk

***Abstract*—Ant colony optimization (ACO) algorithms have proved to be able to adapt in problems that change dynamically. One of the key issues for ACO when a change occurs is that the pheromone trails generated in the previous environment will not be compatible with the new environment. Therefore, the optimization process may be biased from the pheromone trails of the previous environment and fail to search for the newly generated global optimum. In this paper, we consider the dynamic travelling salesman problem (DTSP) in which the weights of the arcs are modified. A pheromone strategy that utilizes change-related information and regulates heuristically the pheromone trails of the affected arcs is proposed. From the experimental results the heuristic-based pheromone strategy performs statistically significant better in most DTSP test cases than other peer ACO algorithms.**

I. Introduction

Ant colony optimization (ACO) has proved that is a powerful metaheuristic to provide optimal (or near-optimal) solutions for solving different combinatorial optimization problems (e.g., the travelling salesman problem (TSP) [3]). Traditionally, researchers have drawn their attention on static optimization problems, where the environment remains fixed during the execution of ACO. However, many real-world applications are subject to dynamic changes. Such problems are known as dynamic optimization problems (DOPs). DOPs are challenging since the aim of an optimization algorithm is not only to find the optimum of the problem quickly, but also to efficiently track the moving optimum during the changing environments [8]. A change in a DOP may involve factors like the objective function, input variables, problem instance and constraints.

ACO algorithms have been originally designed to address static optimization problems [2] (e.g., to converge fast into an optimum or near-optimum solution) and may face a serious challenge when addressing DOPs. This is because, after a change, the pheromone trails of the previous environment may bias the colony to search into an old optimum, making it difficult to track the moving optimum. As a result, ACO will not adapt well to dynamic changes once the colony converges into an optimum. Considering that DOPs can be taken as a series of static problem instances, a simple way to tackle them is to reinitialize all the pheromone trails with an equal amount and consider every dynamic change as the arrival of a new problem instance which needs to be solved from scratch [5], [11]. However, this restart strategy is generally not very efficient because all the information gained from previously optimized environments is removed.

In contrast, once ACO algorithms are enhanced properly they are able to adapt to dynamic changes [1], [8]. Several pheromone strategies have been proposed and integrated with ACO to shorten the re-optimization time and maintain a high quality of the output efficiently, simultaneously. These strategies can be categorized as: increasing diversity after a change [5], [7], [11]; maintaining diversity during the execution [4], [12]; multi-colony schemes [13], [15]; memetic algorithms [9], [10] and memory-based schemes [6] (refer to [14] for a recent comprehensive survey).

From the existing pheromone strategies only few of them utilize change-related information such as the location of dynamic changes [5], [7]. For example, the pheromone trails associated with the arcs incident to a newly added city are regulated accordingly for the dynamic TSP (DTSP) where the topology changes. However, pheromone strategies that utilize change-related information for DTSPs with weight changes (e.g., the weight of the arcs may increase or decrease but the topology remains the same) do no exist. In this paper, a pheromone strategy that regulates heuristically the pheromone trails for the DTSP with weight change is proposed. The key idea of the strategy is to utilize the existing heuristic information of the arcs affected by the dynamic change and regulate their pheromone trails accordingly. For example, an arc may have very high concentration of pheromone trails before the change but its weight may increase after a change. However, due to its high intensity of pheromone trails, it may be still very attractive and may result in poor solution quality. Therefore, penalizing the pheromone trails of the particular arc will make it less attractive for the colony.

The rest of the paper is organized as follows. Section II describes the generation of a dynamic environment using the TSP as the base problem. Specifically, the DTSP with weight changes is described. Section III describes the ACO metaheuristic and how to respond to dynamic changes. Section IV describes the proposed pheromone strategy which utilizes change-related and heuristic information. Section V gives the experimental results and analysis. Finally, Section VI concludes this paper with discussions on future work.

II. DTSP WITH WEIGHT CHANGES

A. Problem Formulation

Typically, a TSP instance is modelled by a fully connected weighted graph $G = (N, A)$, where $N = \{1, \ldots, n\}$ is a set of n nodes and $A = \{(i, j) \mid i, j \in N, i \neq j\}$ is a set of arcs. For the classic TSP, nodes and arcs represent the cities and the links between them. Each arc $(i, j) \in A$ is associated with a non- negative value $w_{ij} \in \mathbb{R}^+$, which for the classic TSP represents the distance or travel time between cities i and j.

B. Generating Dynamic Environments

The TSP becomes more realistic and challenging when it has a dynamic environment [12]. Specifically, the DTSP consists of weight matrix which is subject to changes and it can be defined as follows:

$$\mathbf{W}(T) = \{w_{ij}(T)\}_{n \times n}, \tag{1}$$

where T is the period of a dynamic change defined as follows:

$$T = \lceil t/f \rceil, \tag{2}$$

where t is the algorithmic iteration count and f is the frequency of change. Note that the introduced dynamic changes are synchronized with the optimization process of the algorithm. Hence, the parameter f is expressed in algorithmic iterations.

The DTSP with weight changes affects the weight matrix (i.e., $\mathbf{W}(T)$ in Eq. (1)) directly. A dynamic test case with this type of changes can be generated by assigning an increasing or decreasing factor value to the arc connecting cities i and j as follows:

$$w_{ij}(T+1) = \begin{cases} w_{ij}(0) + \mathcal{R}_{ij}, & \text{if arc } (i,j) \in A_S(T), \\ w_{ij}(T), & \text{otherwise}, \end{cases} \tag{3}$$

$w_{ij}(0)$ is the initial weight of the arc connecting cities i and j (i.e., from the static problem instance when $T = 0$), $\mathcal{R}_{ij}$ is a normally distributed random number (with zero mean and standard deviation set to $0.2 \times w_{ij}(0)$ [16]) that defines the modified factor value of the arc, $A_S(T) \subset A$ defines the set of arcs randomly selected for the change at that period and T defines the environmental period index as defined in Eq. (2).

The size of the set A is defined by the number of arcs (i.e., $n(n-1)$). Hence, the size of $A_S(T)$ is defined by the magnitude of change (i.e., $m \in [0, 1]$) and the size of A. For example, at period T, exactly $\lceil mn(n-1) \rceil$ arcs will be selected to change their weights. The higher the value of m, the more arcs will be selected for changes.

A particular solution $\pi = [\pi_1, \ldots, \pi_n]$ in the search space is specified by a permutation of the city indices, and for the DTSP, it is evaluated as follows:

$$\phi(\pi, t) = w_{\pi_n \pi_1}(T) + \sum_{i=1}^{n-1} w_{\pi_i \pi_{i+1}}(T). \tag{4}$$

III. ANT COLONY OPTIMIZATION

A. Constructing Solutions

Ants read pheromones to construct solutions and write pheromones to mark their constructed solutions. With probability q_0 ($q_0 \in [0, 1]$) each kth ant chooses the next city j with the highest probability as follows:

$$j = \arg\max_{l \in \mathcal{N}_i^k} \{[\tau_{il}]^\alpha [\eta_{il}]^\beta\}, \tag{5}$$

and with probability $(1 - q_0)$ the kth ant uses a probabilistic rule to choose city j from city i as follows:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in \mathcal{N}_i^k, \tag{6}$$

where τ_{ij} and η_{ij} are the existing pheromone trail and the heuristic information available a priori between cities i and j, respectively. The heuristic information is calculated as $\eta_{ij} = 1/w_{ij}(T)$ where $w_{ij}(T)$ is defined as in Eq. (3). $\mathcal{N}_i^k$ is the set of unvisited cities for ant k adjacent to city i. α and β are the two parameters which determine the relative influence of τ_{ij} and η_{ij}, respectively.

B. Updating Pheromones

In this paper, the pheromone update policy from [7] is used. This is because the pheromone policy is simplified and the effect of the pheromone strategy can be investigated. At the beginning the pheromone trails are initialized as follows:

$$\tau_0 \leftarrow \frac{1}{(n-1)}, \forall (i, j) \in A. \tag{7}$$

Then, the pheromone trails are updated by first applying evaporation as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\, \tau_{ij}, \forall (i, j), \tag{8}$$

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

$$\tau_{ij} \leftarrow \tau_{ij} + l\rho, \forall (i, j) \in \pi^{best}, \tag{9}$$

where π^{best} is solution of the iteration best ant (i.e., $\pi^{best} = \pi^{ib}$) and l is a constant. In addition, the best-so-far ant[1] is allowed to deposit pheromone (i.e., $\pi^{best} = \pi^{bs}$).

C. Responding to Dynamic Changes

A straightforward method of the aforementioned ACO to address dynamic changes is to adapt via pheromone evaporation. More precisely, pheromone evaporation will help the colony to forget previously poor decisions by eliminating unused pheromone trails. Another method to respond to dynamic changes is to perform explicit action whenever a change occurs such as to reset all pheromone trails back to τ_0 [11] as follows:

$$\tau_{ij} \leftarrow \tau_0, \forall (i, j), \tag{10}$$

[1]Best-so-far ant is a special ant that may not necessarily belong in the current colony

Algorithm 1 ACO

```
1: t ← 0
2: InitializePheromoneTrails(τ_0)
3: while (termination condition not satisfied) do
4:    ConstructSolutions
5:    π^ib ← FindIterationBest
6:    if (φ(π^ib, t) < φ(π^bs, t)) then
7:       π^bs ← π^ib
8:    end if
9:    PheromoneUpdate
10:   if (dynamic change occurs && respond == true) then
11:      ExplicitAction
12:   end if
13:   t ← t + 1
14: end while
15: OUTPUT: π^bs    %best-so-far solution
```

where τ_0 is defined as in Eq. (7), and consider every dynamic change as the arrival of a new problem instance that needs to be optimized from scratch. The former method may be useful when changes are small to medium whereas the latter method makes more sense when the changes are severe [11]. Algorithm 1 outlines the main components of the described ACO in dynamic environments. When the "respond" parameter in line 10 is set to *true* then ACO will perform an ExplicitAction (e.g., pheromone re-initialization as in Eq. (10)).

However, both methods are limited because they do not take into account the location of the dynamic change. In other words, they do not utilize any change-related or heuristic information of the problem. Such information may help to shorten the re-optimization process when dynamic changes occur.

IV. Heuristic-Based Pheromone Strategy

Considering the current available information and communication technologies (e.g., global position system, cloud computing, etc.) the period and location of changes can be easily made available for the optimization algorithm to explore and utilize. Specifically, when a dynamic change occurs, the heuristic information (i.e., η_{ij}) that ACO utilizes when constructing solutions using Eq. (6) are modified due to the changes on the weights of the arcs in Eq. (3). However, the pheromone trails (i.e., τ_{ij}) of the affected arcs typically remain unchanged (and regulated by the pheromone evaporation [11]) or re-initialized to τ_0 [5], [7].

A better strategy would be to modify the pheromone trails according to the change of the heuristic information. This corresponds to the modified degree of each arc (i, j) calculated as follows:

$$\delta_{ij} = \frac{\eta_{ij}(T-1) - \eta_{ij}(T)}{\max\left(\eta_{ij}(T), \eta_{ij}(T-1)\right)}, \tag{11}$$

where η_{ij} is defined in Eq. (6). Basically, is the normalized difference of the heuristic information of the previous environment $T-1$ and the current environment T for each arc (i, j).

The key idea of the heuristic-based pheromone strategy is to regulate the current pheromone trails (i.e., τ_{ij}) according to the information gathered from the heuristic information of ACO in Eq. (11). More precisely, if the weight of arc (i, j) is decreased (i.e., a negative δ_{ij} value); then the pheromone trails on the arc must be increased accordingly. Otherwise, if the weight of arc (i, j) is increased (i.e., a positive δ_{ij} value); then the pheromone trails on the arc must be decreased accordingly. In this way, the arcs with increasing weights will become less attractive to ants by reducing the pheromone trails and vice versa. Therefore, every arc (i, j) is regulated by a factor defined as follows:

$$\gamma_{ij} = \begin{cases} 1 + \mid \delta_{ij} \mid, & \text{if } \delta_{ij} > 0, \\ 1 - \delta_{ij}, & \text{if } \delta_{ij} < 0, \end{cases} \tag{12}$$

where δ_{ij} is defined as in Eq. (11).

Using the aforementioned factors, the current pheromone trails associated to the modified arcs at time T (i.e., $A_S(T)$) are regulated as follows:

$$\tau_{ij} \leftarrow \tau_0 + \frac{\gamma_{ij}(\tau_{ij} - \tau_0)}{\psi}, \forall (i, j) \in A_S(T), \tag{13}$$

where τ_0 is the initial pheromone trail defined in Eq. (7), γ_{ij} is the factor that regulates arc (i, j) defined in Eq. (12) and ψ is a constant parameter. Recall that this pheromone modification is used as an ExplicitAction in Algorithm 1 – line 11. The way the pheromone trails are regulated by only considering the accumulated pheromone trails (i.e., the difference of the current with the initial pheromone trails). This is to avoid reducing the pheromone trails below the initial pheromone value.

V. Experimental Study

A. Experimental Setup

The benchmark generator described in Section II can convert any static TSP problem instance into a DTSP. Three static TSP benchmark instances (i.e., i.e., `kroA100`, `kroA150`, and `kroA200`,) were obtained from TSPLIB[2] to generate different test cases of DTSP with weight changes. More precisely, the frequency of change was set to $f = 50$, $f = 100$ and $f = 500$ iterations indicating environmental changes from high to low frequencies, respectively, and the magnitude of change was set to $m = 0.1$, $m = 0.25$, $m = 0.5$, and $m = 0.75$, indicating the degree of environmental changes from small, to medium, and to large, respectively. As a result, twelve DTSP test cases (i.e., three values of f × four values of m) were generated from each problem instance to systematically analyze the proposed pheromone strategy. For each DTSP test case, 25 environmental changes were allowed.

An observation of the best-so-far solution after a dynamic change was recorded every algorithmic iteration and used to evaluate the performance for 30 independent executions

[2]A library that consists of TSP problem instances with their optimal solutions, which is available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

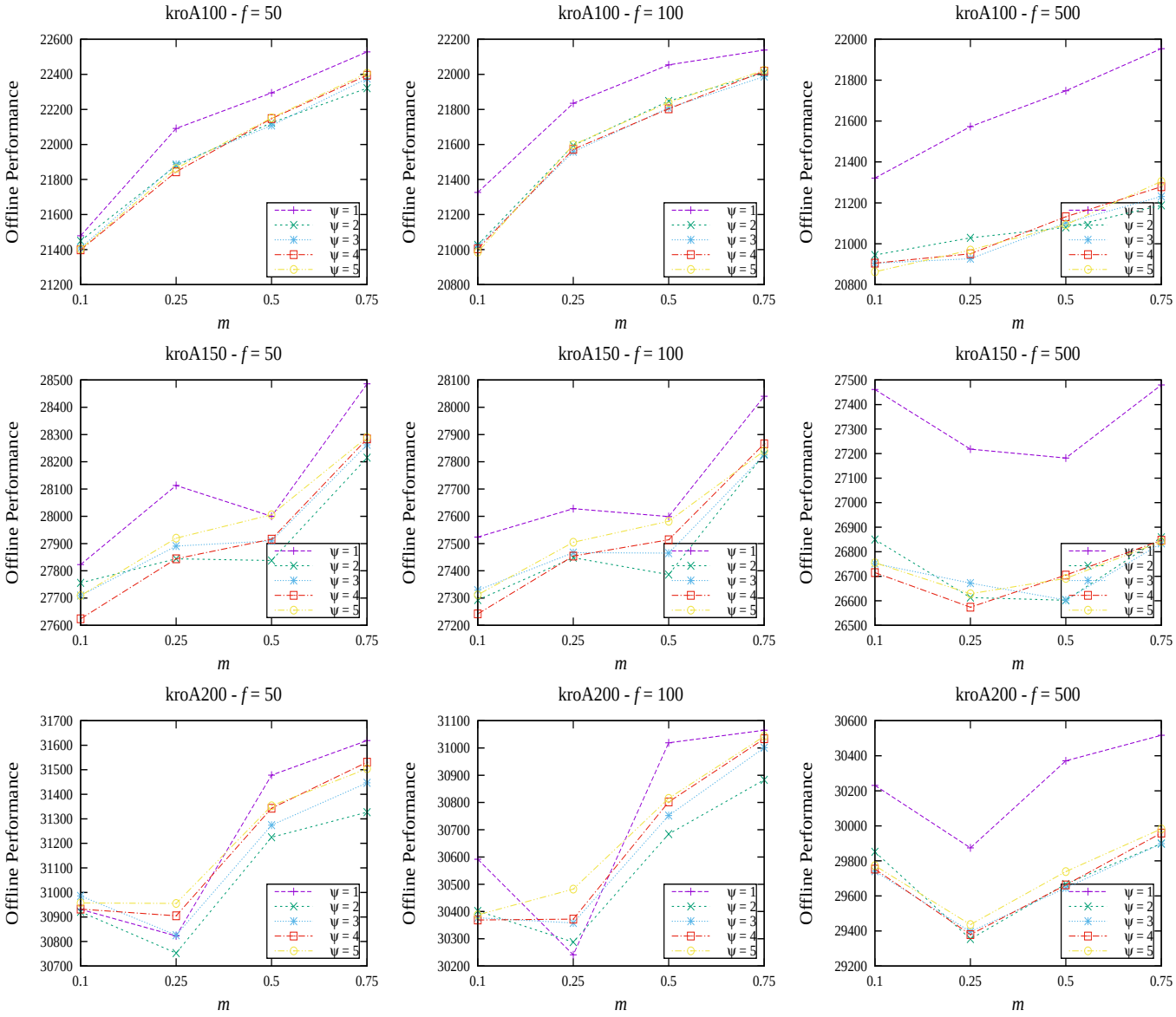

Fig. 1. Experimental results on varying the ψ parameter for different DTSPs.

(with a different random seed for an algorithm and the same random seed for the dynamic environment on each execution). Therefore, the overall *offline performance* [8] is defined as follows:

$$\bar{P}_{OFF} = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{1}{E}\sum_{j=1}^{E}P_{ij}^{*}\right), \tag{14}$$

where K is the total number of iterations, E is the number of independent executions and P_{ij}^{*} is the best-so-far solution cost (after a change) of iteration i of execution j. For a fair comparison, all the algorithms performed the same number of iterations, and for each iteration the same number of ants (corresponding to function evaluations) is allowed. The experiments were performed under a Linux System with an Intel Core i7–6700 4.0GHz processor with 8MB cache and 16GB RAM.

In order to investigate the effect of the proposed heuristic-based pheromone strategy the following ACO variants were used in the experiments:

- ACO: An algorithm with a standard pheromone policy as described in Section III. When a dynamic change occurs no explicit actions are performed. Out of date pheromone trails are regulated by the pheromone evaporation. In Algorithm 1 the respond parameter is set to $false$
- ACO+`global`: Same algorithm as ACO above but when a dynamic change occurs all pheromone trails are re-initialized back to their initial value (i.e., τ_0). In Algorithm 1 the respond parameter is set to $true$.
- ACO+`partial`: Same algorithm as ACO+`global`

TABLE I
EXPERIMENTAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF ACO ALGORITHMS

Algorithms & DTSPs	kroA100				kroA150				kroA200			
$f = 50$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO	21468	21973	22249	22348	27744	27955	27887	28376	30917	30799	31290	31470
ACO+global	21540	22307	22384	22554	27929	28235	28399	28464	31320	31292	31646	31762
ACO+partial	21447	22073	22318	22579	27794	28069	28289	28578	31155	31228	31868	31862
ACO+heuristic	21450	21879	22124	22321	27756	27844	27838	28215	30921	30753	31225	31327
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO	21170	21793	22044	21985	27416	27513	27536	28012	30501	30264	30846	31003
ACO+global	21198	21854	21943	22090	27463	27704	27831	27899	30775	30713	31046	31172
ACO+partial	21046	21646	21919	22139	27355	27595	27764	28068	30614	30672	31318	31310
ACO+heuristic	21027	21594	21848	22009	27293	27448	27386	27828	30402	30288	30684	30883
$f = 500$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO	21325	21709	21963	21972	27346	27213	27235	27444	30243	29855	30279	30435
ACO+global	20700	21128	21146	21267	26612	26713	26754	26822	29848	29555	29849	29968
ACO+partial	20830	20973	21201	21340	26703	26658	26854	26975	29715	29511	30034	30193
ACO+heuristic	20945	21029	21080	21188	26850	26613	26602	26859	29852	29354	29668	29899

above but when a dynamic change occurs only the pheromone trails associated with the modified arcs are re-initialized back to their initial pheromone value (i.e., τ_0). In Algorithm 1 the respond parameter is set to $true$.

- ACO+heuristic: This is the ACO with the proposed pheromone strategy. It is the same algorithm as ACO+partial regarding which pheromone trails will be modified but differ on how the pheromone trails are modified (i.e., they are regulated heuristically based on Eq. (13)). In Algorithm 1 the respond parameter is set to $true$.

The colony size consists of 10 ants for all ACO variants. The common parameters used were set to typical values as follows: $\alpha = 1$, $\beta = 5$, $\rho = 0.05$, $l = 0.25$ and $q_0 = 0.5$ whereas the effect of the ψ parameter was investigated (see the following subsection).

B. Experimental Results on the Effect of ψ Parameter

The ψ parameter defined in Eq. (13) determines the relative influence of change-related information. Since it is a key parameter it was further investigated with values $\psi \in \{1, 2, 3, 4, 5\}$. The offline performance of ACO+heuristic with different ψ values is given in Fig. 1 for all dynamic test cases.

From Fig. 1 it can be observed that when the influence of change-related information is strong (i.e., $\psi = 1$) the offline performance is poor in almost all dynamic test cases. This is probably because the optimization process gets trapped in a poor local optimum due to the strong knowledge imposed after the dynamic change. When $\psi > 1$ the offline performance is improved significantly with $\psi = 2$ having acceptable performance in most DTSP test cases for all three problem instances. For the remaining experiments the ψ parameter is set to $\psi = 2$ for the ACO+heuristic variation.

C. Experimental Results and Analysis

The experimental results regarding the offline performance of the aforementioned ACO variants for each dynamic test case are presented in Table I. The corresponding statistical results of the proposed ACO+heuristic against the competing variants are presented in Table II, where pairwise comparisons are performed using Mann–Whitney statistical test. In Table II, the results are shown as "+", "−" and "∼" when the first algorithm is significantly better than the second one, when the second algorithm is significantly better than the first one and when the two algorithms are not significantly different, respectively. Moreover, the dynamic behaviour of ACO variants in terms of offline performance for the first 10 environmental changes with $f = 50$ and $m = 0.25$, with $f = 100$ and $m = 0.25$, with $f = 500$ and $m = 0.25$, are presented in Fig. 2, Fig. 3 and Fig. 4, respectively. From the experimental results, several observations can be made by comparing the behaviour of ACO+heuristic with the competing ACO variants.

First, ACO+heuristic is competitive with a conventional ACO when $f = 50$, significantly better in most DTSP test cases when $f = 100$ and significantly better in all DTSP test cases when $f = 500$; see the comparisons of ACO+heuristic $\Leftrightarrow$ ACO in Table II. This observations shows that the proposed heuristic requires some time to express its effect. When not enough time is available the colony does not have enough time to utilize the change-related

TABLE II
STATISTICAL RESULTS REGARDING THE OFFLINE PERFORMANCE OF ACO ALGORITHMS

Algorithms & DTSPs	kroA100				kroA150				kroA200			
$f = 50$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO+heuristic $\Leftrightarrow$ ACO	$\sim$	$\sim$	$+$	$\sim$	$\sim$	$+$	$\sim$	$+$	$\sim$	$\sim$	$\sim$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+global	$+$	$+$	$+$	$-$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+partial	$\sim$	$+$	$+$	$+$	$\sim$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO+heuristic $\Leftrightarrow$ ACO	$+$	$+$	$+$	$\sim$	$\sim$	$\sim$	$+$	$+$	$\sim$	$\sim$	$+$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+global	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+partial	$\sim$	$\sim$	$+$	$+$	$\sim$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
$f = 500$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
ACO+heuristic $\Leftrightarrow$ ACO	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+global	$-$	$+$	$+$	$+$	$-$	$+$	$+$	$\sim$	$\sim$	$+$	$+$	$+$
ACO+heuristic $\Leftrightarrow$ ACO+partial	$-$	$\sim$	$+$	$+$	$\sim$	$\sim$	$+$	$+$	$\sim$	$+$	$+$	$+$

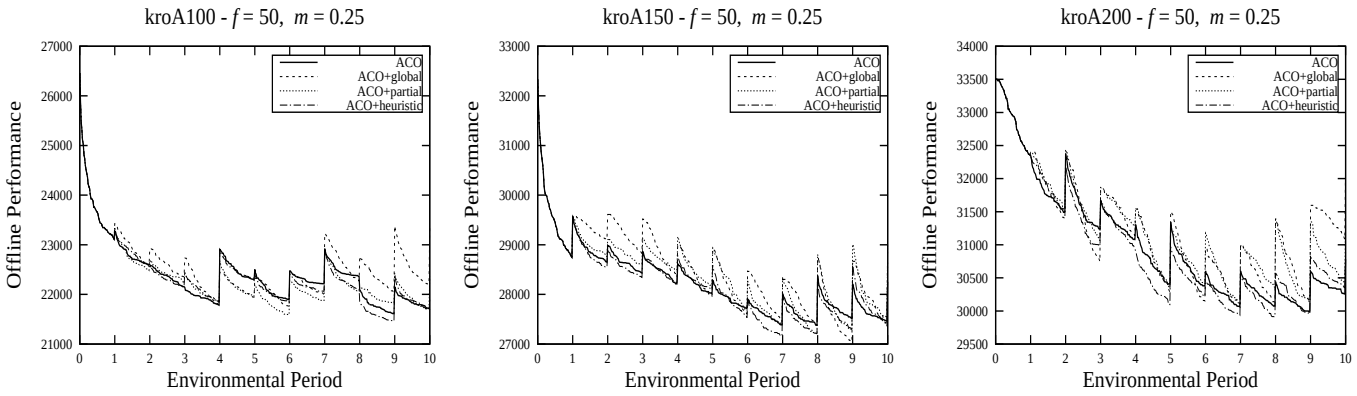

Fig. 2. Dynamic offline performance of ACO variants for the first 10 environmental changes for DTSPs with $f = 50$ and $m = 0.25$.

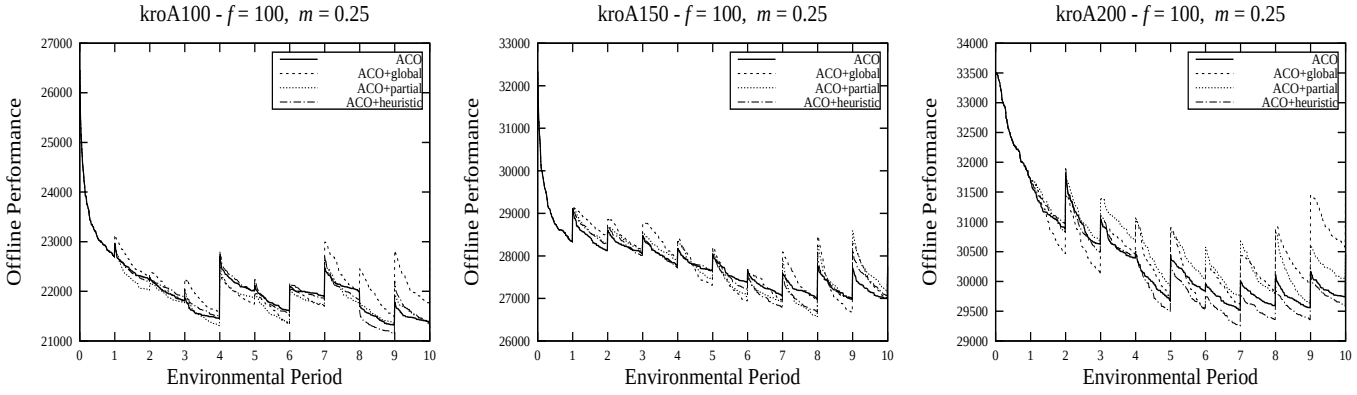

Fig. 3. Dynamic offline performance of ACO variants for the first 10 environmental changes for DTSPs with $f = 100$ and $m = 0.25$.

information imposed to the pheromone trails. Furthermore, it can be observed from Fig. 4 that ACO gets trapped in a local optimum solution (approximately) after the fifth dynamic change when $f = 500$.

Second, ACO+heuristic is significantly better than ACO+global in almost all DTSP test cases; see the comparisons of ACO+heuristic $\Leftrightarrow$ ACO+global in Table II. This observation supports the limitation of ACO+global of

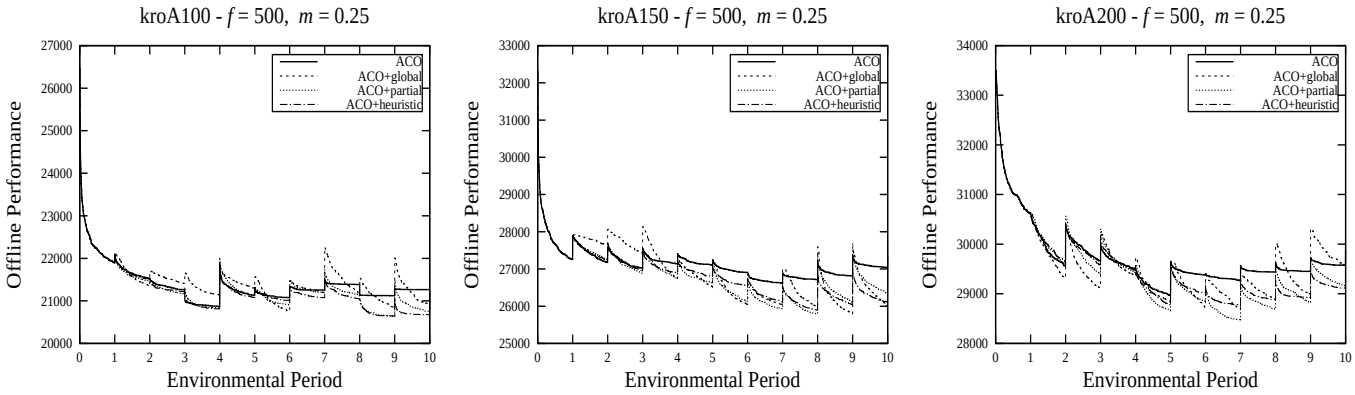

Fig. 4. Dynamic offline performance of ACO variants for the first 10 environmental changes for DTSPs with $f = 500$ and $m = 0.25$.

not taking into account the location of the dynamic changes. The poor performance of ACO+`global` is expected because all the previously gained knowledge of the algorithm is erased after a dynamic change. In fact, the offline performance results in Table I of ACO+`global` against ACO+`partial` shows the positive effect of utilizing change-related information. The difference of these two algorithms is that the former reinitializes the pheromone trails of all arcs to τ_0 whereas the latter reinitializes only the pheromone trails of the arcs affected by the changes to τ_0.

Third, ACO+`heuristic` is significantly better than ACO+`partial` in almost all DTSP test cases; see the comparisons of ACO+`heuristic` $\Leftrightarrow$ ACO+`partial` in Table II. Although the latter algorithm takes into account the location of the dynamic changes it does not regulate the pheromone trails heuristically as the former one. It can be observed from Fig 2, Fig. 3 and Fig. 4 that the proposed ACO+`heuristic` maintains better offline performance in most dynamic changes. This shows that regulating the pheromone trails according to the change of the heuristic information has a positive effect in most DTSP test cases.

VI. Conclusions

The pheromone trails generated from ACO in one environment may not be compatible when the environment changes. Utilizing change-related and heuristic information, older pheromone trails can be adapted effectively to the new environment. In this paper the DTSP with weight changes is used as the base to generate dynamic test cases. A pheromone strategy based on the heuristic information of the affected arcs is proposed to enhance the adaptation capabilities of ACO.

From our experimental studies the following concluding remarks can be drawn. First, pheromone strategies utilizing change-related information perform significantly better than conventional pheromone strategies. Second, regulating the pheromone trails according the heuristic information improves the performance of ACO in changing environments. Third, if too much information is transferred from the existing heuristic information it may start the optimization process from (or near) a poor local optimum and get stuck there.

Since the utilization of change-related and heuristic information improves the performance of ACO, for future work it would be interesting to design pheromone strategies based on fuzzy logic. In this way, the pheromone strategy would be able to better capture more DTSP test cases and further improve the performance of ACO in dynamic environments.

References

[1] D. Angus and T. Hendtlass, "Ant colony optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Hendtlass and M. Ali, Eds. Springer Berlin Heidelberg, 2002, vol. 2358, pp. 618–627.

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

[3] M. Dorigo and T. Stützle, *Ant colony optimization*. Cambridge, MA: MIT Press, 2004.

[4] C. Eyckelhof and M. Snoek, "Ant systems for a dynamic TSP," in *Ant Algorithms*, ser. Lecture Notes in Computer Science, M. Dorigo, G. Di Caro, and M. Sampels, Eds. Springer Berlin Heidelberg, 2002, vol. 2463, pp. 88–99.

[5] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic TSP," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, E. J. W. Boers, Ed. Springer Berlin Heidelberg, 2001, vol. 2037, pp. 213–222.

[6] ——, "Applying population based ACO to dynamic optimization problems," in *Ant Algorithms*, ser. Lecture Notes in Computer Science, M. Dorigo, G. Di Caro, and M. Sampels, Eds. Springer Berlin Heidelberg, 2002, vol. 2463, pp. 111–122.

[7] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic TSP," in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 2001, pp. 860–867.

[8] 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.

[9] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for the dynamic travelling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, 2017.

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

[11] ——, "Adapting the pheromone evaporation rate in dynamic routing problems," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, A. Esparcia-Alcázar, Ed. Springer Berlin Heidelberg, 2013, vol. 7835, pp. 606–615.

[12] ——, "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.

[13] M. Mavrovouniotis, S. Yang, and X. Yao, "Multi-colony ant algorithms for the dynamic travelling salesman problem," in *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, Dec 2014, pp. 9–16.

[14] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1 – 17, 2017.

[15] L. Melo, F. Pereira, and E. Costa, "Multi-caste ant colony algorithm for the dynamic traveling salesperson problem," in *Adaptive and Natural Computing Algorithms*, ser. Lecture Notes in Computer Science, M. Tomassini, A. Antonioni, F. Daolio, and P. Buesser, Eds. Springer Berlin Heidelberg, 2013, vol. 7824, pp. 179–188.

[16] R. Tinós, D. Whitley, and A. Howe, "Use of explicit memory in the dynamic traveling salesman problem," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2014, pp. 999–1006.