An Ant Colony Optimization Based Memetic Algorithm for the Dynamic Travelling Salesman Problem

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

Felipe Martins Müller*
Technological Center
Department of Applied Computing
Federal University of Santa Maria
97105–900, Santa Maria, Brazil
felipe@inf.ufsm.br

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

ABSTRACT

Ant colony optimization (ACO) algorithms have proved to be able to adapt for solving dynamic optimization problems (DOPs). The integration of local search algorithms has also proved to significantly improve the output of ACO algorithms. However, almost all previous works consider stationary environments. In this paper, the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, one of the best ACO variations, is integrated with the unstringing and stringing (US) local search operator for the dynamic travelling salesman problem (DTSP). The best solution constructed by ACO is passed to the US operator for local search improvements. The proposed memetic algorithm aims to combine the adaptation capabilities of ACO for DOPs and the superior performance of the US operator on the static travelling salesman problem in order to tackle the DTSP. The experiments show that the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System is able to provide good initial solutions to US and the proposed algorithm outperforms other peer ACO-based memetic algorithms on different DTSPs.

Categories and Subject Descriptors

G.2.1 [**Combinatorics**]: Combinatorial algorithms

Keywords

Ant colony optimization, Memetic computing, Local search, Dynamic Travelling salesman problem

1. INTRODUCTION

Ant colony optimization (ACO) algorithms have proved that they are powerful tools to provide near-optimal solutions for solving different combinatorial optimization problems, e.g., the travelling salesman problem (TSP) [6]. Traditionally, researchers have drawn their attention on stationary optimization problems, where the environment remains fixed during the execution of an algorithm. However, many real-world applications are subject to dynamic environments. Dynamic optimization problems (DOPs) are challenging since the aim of an algorithm is not only to find the optimum of the problem quickly, but to efficiently track the moving optimum when changes occur [16]. A change in a DOP may involve factors like the objective function, input variables, problem instance, constraints, and so on.

ACO algorithms have been originally designed for stationary optimization problems [5], e.g., to converge fast into an optimum or near-optimum solution, and may face a serious challenge to tackle DOPs. This is because, after a change, the pheromone trails of the previous environment may bias the population to search into an old optimum, making it difficult to track the moving optimum. As a result, ACO will not adapt well to dyanmic changes once the population converges into an optimum. Considering that DOPs can be taken as a series of stationary problem instances, a simple way to tackle them is to reinitialize the pheromone trails and consider every dynamic change as the arrival of a new problem instance which needs to be solved from scratch d_{ij} is the distance [13, 18]. However, this restart strategy is generally not efficient.

In contrast, once enhanced properly, ACO algorithms are able to adapt to dynamic changes since they are inspired from nature, which is a continuously changing process [1, 16]. Several 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 [13, 18]; maintaining diversity during the execution [7, 19]; and memory-based schemes [14, 15].

The integration of local search operations with ACO or other metaheuristics, which leads to the so-called memetic algorithms, has proved that the performance of these algorithms can be further improved regarding the solution quality [17, 22, 24]. This is because local search algorithms can better explore locally a neighbourhood in the search space than ACO. However, local search algorithms generate strong exploitation at the global level since they are optimizing to-

*Visiting professor at the Centre for Computational Intelligence (CCI), De Montfort University, Leicester, UK

GECCO '15, July 11 - 16, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754651

wards the optimum. Therefore, the use of ACO provides the exploration and the knowledge transfer needed to adapt to DOPs, whereas a local search algorithm improves the output. In this paper, we integrate an advanced local search operator, called unstringing and stringing (US) [12], which was proposed for the TSP, within ACO to address dynamic TSPs (DTSPs). The dynamic benchmark generator for permutation problems (DBGP) [20] is used to generate DTSPs in order to systematically analyze the investigated algorithms.

The rest of the paper is organized as follows. Section 2 describes the TSP and the construction of DTSP using the DBGP. Section 3 describes one of the best variations of ACO used for the experiments. Section 4 describes the proposed memetic framework for ACO algorithms. Section 5 gives the experimental results, including the statistical tests, and analysis. Finally, Section 6 concludes this paper with discussions on relevant future work.

2. TRAVELLING SALESMAN PROBLEMS

2.1 Problem Description

The TSP can be described as follows: given a collection of cities, the objective is to find the Hamiltonian cycle that starts from one city and visits each of the other cities once before returning to the starting city. Typically, the problem is modelled by a fully connected weighted graph $G = (N, A)$, where $N = \{v_0, \ldots, v_n\}$ is a set of nodes and $A = \{(v_i, v_j) \in N : i \neq j\}$ is a set of arcs. Each arc (v_i, v_j) is associated with a non-negative value d_{ij} which represents the distance between cities v_i and v_j.

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

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

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

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

where n is the number of cities and d_{ij} is the distance between cities v_i and v_j.

2.2 Dynamic Benchmark Generators

Over the years, several dynamic benchmark generators have been proposed for the DTSP that tend to model real-world scenarios, such as the DTSP with traffic factors [7, 19] and the DTSP with exchangeable cities [13, 14, 17]. These benchmark generators modify the fitness landscape, whenever a dynamic change occurs, and cause the optimum value to change.

In this paper, the recently proposed dynamic benchmark generator for permutation-encoded problems (DBGP)[1] is used [20], which can convert any stationary permutation-encoded benchmark problem instance to a DOP. The fitness landscape is not changed 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 rather than modifying the fitness landscape as other benchmark generators do.

[1]Available from: http://www.tech.dmu.ac.uk/∼syang/Codes/DBGP.zip

The main advantage of using the DBGP over other generators is that one can observe how close to the optimum an algorithm can perform when a dynamic change occurs. One can argue that since a DOP can be considered as a series of static problem instances, a direct way is to solve each problem instance to optimality, which may be non-trivial due to the $\mathcal{NP}$-hardness of most combinatorial optimization problems, especially for large-size problem instances [11]. It may be possible for small-size problem instances, but then the usefulness of benchmarking will be reduced.

The drawback of DBGP is that it sacrifices the realistic modelling of application problems for the sake of benchmarking. However, when comparing algorithms on the DTSP with traffic factors, the optimum value is unknown. Although such DOPs satisfy the modelling of a real-world scenario, using them leads to other problems. For example, one can see that one algorithm performs better than the others, but it may be the case that all algorithms converge far away from the actual optimum. Hence, with DBGP, apart from the comparison benefits described previously, it is also possible to assess the difficulty of a DOP. Particularly, in case all the algorithms perform far away from the optimum on a specific DOP, and closer to the optimum on another DOP, it gives an indication that the former DOP is more difficult to solve than the latter one.

2.3 Dynamic Test Environments

Considering the TSP description above, each city $i \in N$ has a location defined by (x, y) and each arc $(v_i, v_j) \in A$ is associated with a non-negative distance d_{ij}. Usually, the distance matrix of a problem instance is defined as $\mathbf{D} = (d_{ij})_{n \times n}$. DBGP generates the dynamic case as follows.

Every f iterations a random vector $\vec{V}(T)$ is generated that contains exactly $m \times n$ cities where $T = \lceil t/f \rceil$ is the index of the period of change, t is the iteration count of the algorithm, f determines the frequency of change, n is the size of the problem instance, and $m \in [0.0, 1.0]$ determines the magnitude of change. Then, a randomly re-ordered vector $\vec{U}(T)$ is generated that contains only the cities of $\vec{V}(T)$. In this way, exactly $m \times n$ pairwise swaps are performed in $\mathbf{D}$ using the two random vectors $(\vec{V}(T) \otimes \vec{U}(T))$, where "$\otimes$" denotes the swap operator.

3. ANT COLONY OPTIMIZATION

3.1 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

The ACO metaheuristic consists of a population of μ ants that construct solutions and share their information among each other via their pheromone trails. The first ACO algorithm, i.e., the Ant System (AS) [3], was developed for solving the TSP. Many variations of the AS have been applied to address difficult optimization problems [6].

One of the best performing AS variations is the $\mathcal{MAX}$-$\mathcal{MIN}$ AS ($\mathcal{MM}$AS) [21]. Ants read pheromones in order to construct their solutions and write pheromones to store their solutions. Each ant k uses a probabilistic rule to choose the next city to visit. The decision rule of the kth ant to move from city v_i to city v_j is defined as follows:

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

where τ_{ij} and η_{ij} are the existing pheromone trail and the heuristic information available a priori between cities v_i and v_j, respectively. The heuristic information is defined as $\eta_{ij} = 1/d_{ij}$ where d_{ij} is defined as in Eq. (2). $\mathcal{N}_i^k$ is the neighbourhood of unvisited cities for ant k adjacent to city v_i. α and β are the two parameters which determine the relative influence of τ_{ij} and η_{ij}, respectively.

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

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

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:

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

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, 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 ant[2] and the iteration best ant, respectively. Both update rules are used in an alternate way in the implementation [22].

The lower and upper limits τ_{min} and τ_{max} of the pheromone trail values are imposed. The τ_{max} value is 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 constant parameter.

Since only the best ant is allowed to deposit pheromone, the population may quickly converge towards the best solution found in the first iteration. Therefore, the pheromone trails are occasionally reinitialized to the τ_{max} value to increase exploration. For example, whenever the stagnation behaviour[3] occurs or when no improved solution is found for a given number of iterations, the pheromone trails are reinitialized.

3.2 Response to Dynamic Changes

ACO algorithms are able to use knowledge from previous environments via their pheromone trails and can be applied directly to DOPs without any modifications [1, 18]. For example, when the changing environments are similar, the pheromone trails of the previous environment may provide knowledge to speed up the optimization process to the new environment. However, the algorithm needs to be flexible enough to accept the knowledge transferred from the pheromone trails, or eliminate the pheromone trails, in order to adapt well to the new environment. In particular, pheromone evaporation enables the algorithm to forget bad decisions made in previous iterations. When a dynamic change occurs, evaporation eliminates the pheromone trails of the previous environment from areas that are generated on the old optimum and helps ants to explore for the new optimum.

[2]The best-so-far ant is a special ant that may not necessarily belong to the current population of ants.

[3]Detected using λ-branching [10] that calculates the statistics regarding the distribution of the current pheromone trails.

Algorithm 1 ACO-based Memetic Framework

```
1: InitializePheromoneTrails
2: while (termination condition not satisfied) do
3:    ConstructSolutions
4:    UpdateBestAnts
5:    if (newBestAntFound is true) then
6:       ApplyLocalSearch
7:    end if
8:    UpdatePheromone
9: end while
```

In case the changing environments are different, then pheromone reinitialization may be a better choice rather than transferring the knowledge from previous pheromone trails [1, 13, 14, 18]. A detection mechanism is required to reinitialize the pheromone trails whenever a dynamic change occurs. The detection mechanism for the DTSPs generated by DBGP is straightforward. A single solution is required to be stored and re-evaluated every iteration. If there is a change to the tour cost, it indicates that a dynamic change has occurred [18].

4. ACO-BASED MEMETIC ALGORITHM

4.1 Memetic Framework

Generally, in ACO based memetic algorithms, when all ants construct solutions, the best-so-far ant after a dynamic change is selected to undergo local search improvements in dynamic environments. For example, in the memetic-ACO (M-ACO) proposed in [17], an adaptive version of the inver-over operator [23] is applied to the best ant for a predefined number of steps. The resulting ant replaces the selected best ant whenever the solution is improved.

The inver-over operator used in M-ACO does not consider any heuristic information as other operators (e.g., 2-opt or 3-opt [4]), hence it is not used for comparison in the experiments. Usually, heuristic based local search operators are applied to solutions until no further improvement is available. Therefore, there is no need to apply such operators to the best-so-far ant after a dynamic change in every iteration. Since local search operators are computational expensive, especially when the problem size is large, the computation time can be reduced when they are applied only when a new best-so-far ant is found (e.g., when the best ant of the current iteration is better than the best ant found from all previous iterations since the last dynamic change). The overall framework of proposed ACO-based memetic algorithms is given in Algorithm 1.

4.2 Local Search Operator

In this paper, we consider a more advanced local search operator previously proposed for the TSP, i.e., US [12]. This specific operator has been originally used for the stationary TSP with promising results when compared with other local search algorithms. In order to take advantage of the adaptation capabilities of ACO, the US operator considers the best solution constructed by ACO.

The US operator basically removes (unstringing) and inserts (stringing) cities from a tour into such position that improves the overall tour cost. It can work equally well to symmetric and asymmetric problems [8, 9]. In this paper, we consider only symmetric TSPs. Hence, to simplify the

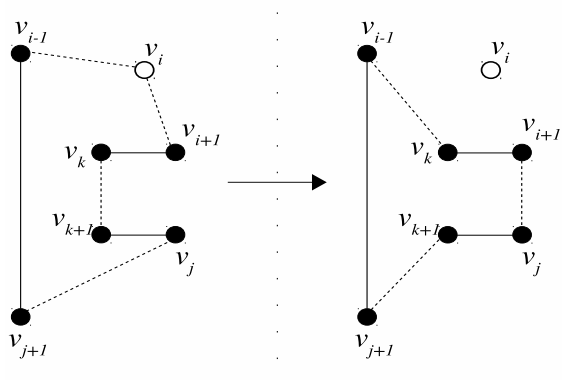

Figure 1: Illustration of Type I removal of v_i.

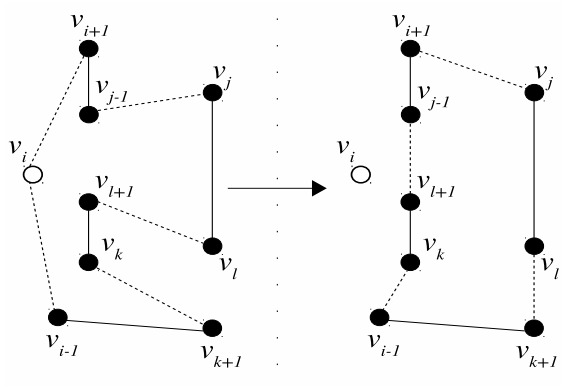

Figure 2: Illustration of Type II removal of v_i.

description of the US operator, we assume only this case, i.e., symmetric TSPs.

Suppose that we wish to insert v_x between any two vertices v_i and v_j. The main feature of the insertion procedure is that when a city v_x is inserted, it is not necessarily placed between two consecutive cities. However, after the insertion, these cities become adjacent to v_x. For a given orientation of a tour, consider v_k a city in the subtour from v_j to v_i and v_l a city in the subtour from v_i to v_j. We also consider for any vertex v_h on the tour, v_{h+1} its successor and v_{h-1} its predecessor. Since the potential number of choices for v_i, v_j and v_x could be large, we limit the search within a neighbourhood of a given size q (the suggested value is $q = 4$) as in most local search operators. Basically, the neighbours of a city v_j are the closest (e.g., the least distance) q successors and predecessors among all cities already included in the tour. The selection of the best place to insert a city in the tour is now constrained to the neighbourhood of each city involved in the alternative under consideration as well as the alternatives tested to the removal procedures.

More precisely, the unstringing procedure suggests two possible types of removals of v_i from the tour as shown in Figures 1 and 2, respectively.

- Type I Removal: Assume that v_j belongs to the neighbourhood of v_{i+1} and v_k belongs to the neighbourhood of v_{i-1}, with v_k being part of the subtour $(v_{i+1}, \ldots, v_{j-1})$. The removal of city v_i results in the deletion of arcs (v_{i-1}, v_i), (v_i, v_{i+1}), (v_k, v_{k+1}) and (v_j, v_{j+1}); and the insertion of arcs (v_{i-1}, v_k), (v_{i+1}, v_j) and (v_{k+1}, v_{j+1}). Also, the subtours $(v_{i+1}, \ldots, v_k)$ and $(v_{k+1}, \ldots, v_j)$ are reversed.

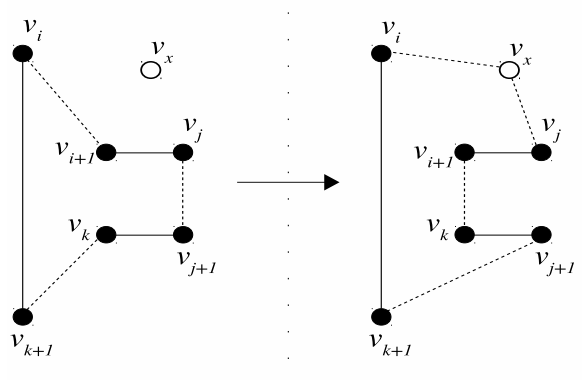

Figure 3: Illustration of Type I insertion of v_x between v_i and v_j.

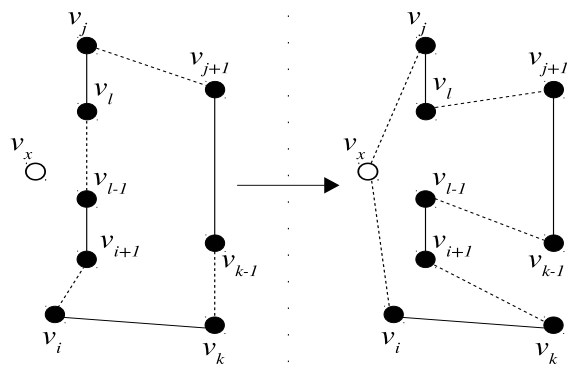

Figure 4: Illustration of Type II insertion of v_x between v_i and v_j.

- Type II Removal: Assume that v_j belongs to the neighbourhood of v_{i+1}, v_k belongs to the neighbourhood of v_{i-1}, with v_k being part of the subtour $(v_{j+1}, \ldots, v_{i-2})$ and v_l belongs to the neighbourhood of v_{k+1}, with v_l being part of the subtour $(v_j, \ldots, v_{k-1})$. The removal of city v_i results in the deletion of arcs (v_{i-1}, v_i), (v_i, v_{i+1}), (v_{j-1}, v_j), (v_k, v_{k+1}) and (v_l, v_{l+1}); and the insertion of arcs (v_{i-1}, v_k), (v_{l+1}, v_{j-1}), (v_{i+1}, v_j) and (v_l, v_{k+1}). As above, the subtours $(v_{i+1}, \ldots, v_{j-1})$ and $(v_{l+1}, \ldots, v_k)$ are reversed.

The stringing procedure is basically the reverse of the unstringing procedure. Since there are two types of removals there are two corresponding types of insertions, as shown in Figures 3 and 4, respectively.

- Type I Insertion: Assume that $v_k \neq v_i$ and $v_k \neq v_j$. The insertion of v_x results in the deletion of arcs (v_i, v_{i+1}), (v_j, v_{j+1}) and (v_k, v_{k+1}); and the insertion of arcs (v_i, v_x), (v_x, v_j), (v_{i+1}, v_k) and (v_{j+1}, v_{k+1}). Also, the subtours $(v_{i+1}, \ldots, v_j)$ and $(v_{j+1}, \ldots, v_k)$ are reversed.

- Type II Insertion: Assume that $v_k \neq v_j$, $v_k \neq v_{j+1}$, $v_l \neq v_i$, and $v_l \neq v_{i+1}$. The insertion of v_x results in

the deletion of arcs (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}) and (v_{k-1}, v_k); and the insertion of arcs (v_i, v_x), (v_x, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) and (v_{i+1}, v_k). As above, the subtours $(v_{i+1}, \ldots, v_{l-1})$ and $(v_l, \ldots, v_j)$ are reversed.

In summary, the unstringing and stringing procedures described above are applied to the tour provided by ACO (i.e., the best-so-far ant) for local search improvements. The resulting solution is then used to deposit pheromone trails using Eq. (5). Note that the same pheromone update policy with $\mathcal{MM}$AS is used.

5. EXPERIMENTAL STUDY

5.1 Experimental Setup

In the experiments, we investigate the effects of having ACO providing its solutions rather than using randomly generated solutions for local search improvements, and compare the proposed ACO with the US operator over the existing integrations of ACO with the 2-opt and 3-opt operators. In particular, the performance of the following algorithms is investigated:

- $\mathcal{MM}$AS: no local search improvement is applied [21].
- $\mathcal{MM}$AS+2opt: the 2-opt operator is applied whenever a new best-so-far ant is found until there is no further improvement [22].
- $\mathcal{MM}$AS+3opt: the 3-opt operator is applied as in the $\mathcal{MM}$AS+2opt [22].
- $\mathcal{MM}$AS+US: the US operator is applied as in $\mathcal{MM}$AS+2opt and $\mathcal{MM}$AS+3opt for a fair comparison.
- $\mathcal{MM}$AS$_R$+US: the same algorithm as $\mathcal{MM}$AS+US but whenever an environmental change is detected the pheromone trails are reinitialized.
- Random+US: the US operator applied on random solutions rather than solutions generated by the ants.

All algorithmic parameters were set to commonly used values: $\alpha = 1$, $\beta = 5$, $\rho = 0.2$ and the number of ants was set to $\mu = 50$, except in $\mathcal{MM}$AS$_R$+US where $\mu = 49$ since one solution was used as a detector to detect dynamic changes.

DTSPs are generated from three stationary benchmark instances obtained from TSPLIB[4] using the DBGP generator described in Section 2. The frequency of change f was set to change every 10 and 100 algorithmic iterations indicating quickly and slowly changing environments, respectively, and the magnitude of change m was set to 0.1, 0.25, 0.5 and 0.75, indicating slightly, to medium, to severely changing environments, respectively. Totally, a series of 8 DTSPs were constructed from each stationary instance. For each ACO algorithm on a DTSP, 30 independent runs were executed on the same set of random seed numbers. For each run, 1000 iterations were allowed and an observation (i.e., the value of the best-so-far ant after a dynamic change) was recorded every iteration.

[4]Available from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

The modified *offline error* [2] was used to evaluate the overall performance of ACO algorithms, which is defined as:

$$\bar{E}_{offline} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R} Err_{ij}\right), \quad (6)$$

where R is the number of runs, E is the number of iterations, and Err_{ij} is the best-so-far error value (i.e., the difference between the tour cost of the best-so-far ant and the optimum value for the fitness landscape) after a change in observation i of run j. Note that this measurement is compatible with DBGP because the optimal value (shown in Table 1) of each benchmark instance is known and remains the same during the environmental changes.

Moreover, the population diversity [19] was recorded as:

$$\bar{T}_{DIV} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R} DIV_{ij}\right), \quad (7)$$

where R and E are defined in Eq. (6) and DIV_{ij} defines the diversity of the population in observation i of run j. For the DTSP, DIV_{ij} can be calculated as follows:

$$DIV_{ij} = \frac{1}{\mu(\mu-1)}\sum_{p=1}^{\mu}\sum_{q\neq p}^{\mu}\left(1 - \frac{c_{E_{pq}}}{n}\right), \quad (8)$$

where μ is the size of population, $c_{E_{pq}}$ is defined as the number of common edges between the solutions of ants p and q, and n is the number of cities.

5.2 Experimental Results and Analysis

The experimental results regarding the offline error of investigated algorithms for all DTSPs are presented in Table 1. The corresponding statistical results are presented in Table 2, where Kruskal–Wallis tests were applied, followed by posthoc paired comparisons using Mann–Whitney tests with the Bonferroni correction. In Table 2, 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. In Figure 5, the dynamic offline error for slowly changing environments against the algorithmic iterations of $\mathcal{MM}$AS+2opt, $\mathcal{MM}$AS+3opt, $\mathcal{MM}$AS+US and $\mathcal{MM}$AS$_R$+US are plotted to better understand the behaviour of the ACO-based memetic algorithms. Figure 6 presents the population diversity of ACO algorithms for all DTSPs. From the experimental results, several observations can be drawn.

First, it can be observed that the performance of $\mathcal{MM}$AS is improved dramatically when integrated with local search algorithms for all DTSPs; see Table 1. This is natural because $\mathcal{MM}$AS performs global optimization and cannot be accurate to the output provided, whereas 2-opt, 3-opt and US perform local optimization and can locate the optimum in a given neighbourhood. $\mathcal{MM}$AS provides a good initial solution for local search algorithms since it may locate the neighbourhood that contains the global optimum or a near-optimum solution. This can be supported by the fact that $\mathcal{MM}$AS+US significantly outperforms Random+US in most DTSPs; see the comparisons of $\mathcal{MM}$AS+US ⇔ Random+US in Table 2. Basically, Random+US is a simple restart strategy whereas $\mathcal{MM}$AS+US is a guided restart strategy, respectively, of the US local search procedure that restarts the US operator for several times (iterations). Their

Table 1: Offline error of ACO algorithms for different DTSPs where bold values indicate best results

Algorithms & DTSPs	kroA100(Optimum=21282)				kroA150(Optimum=26524)				kroA200(Optimum=29368)			
$f = 10$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}$AS	1183	3107	3915	4123	2737	4846	5804	6003	3689	6196	7135	7393
$\mathcal{MM}$AS+2opt	341	1084	1253	1307	1122	1732	1891	1927	1400	2038	2205	2251
$\mathcal{MM}$AS+3opt	135	374	444	443	525	722	755	770	611	861	911	927
$\mathcal{MM}$AS+US	**100**	227	247	236	**524**	593	602	586	**586**	729	748	756
$\mathcal{MM}$AS$_R$+US	220	**225**	**223**	**229**	597	**591**	**583**	**582**	728	**727**	**728**	**723**
Random+US	272	286	302	286	621	622	616	605	832	821	839	836
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}$AS	419	844	1176	1329	1049	1810	2261	2353	1108	2052	2847	2939
$\mathcal{MM}$AS+2opt	190	408	662	692	716	1154	1344	1384	695	1237	1608	1641
$\mathcal{MM}$AS+3opt	100	201	334	366	446	627	708	730	450	705	838	885
$\mathcal{MM}$AS+US	**65**	**145**	189	213	**434**	**550**	558	555	**455**	**655**	725	752
$\mathcal{MM}$AS$_R$+US	164	146	**168**	**169**	573	574	**548**	**554**	688	666	**714**	**672**
Random+US	290	294	297	261	618	612	629	604	810	854	808	831

Table 2: Statistical comparisons of ACO algorithms for different DTSPs.

Algorithms & DTSPs	kroA100				kroA150				kroA200			
$f = 10$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS	+	+	+	+	+	+	+	+	+	+	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS+2opt	+	+	+	+	+	+	+	+	+	+	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS+3opt	+	+	+	+	$\sim$	+	+	+	$\sim$	+	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS$_R$+US	+	$\sim$	$-$	$\sim$	+	$\sim$	$\sim$	$\sim$	+	$\sim$	$-$	$-$
$\mathcal{MM}$AS+US $\Leftrightarrow$ Random+US	+	+	+	+	+	+	$\sim$	$\sim$	+	+	+	+
$f = 100$, $m \Rightarrow$	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS	+	+	+	+	+	+	+	+	+	+	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS+2opt	+	+	+	+	+	+	+	+	+	+	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS+3opt	+	+	+	+	$\sim$	+	+	+	$\sim$	$\sim$	+	+
$\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS$_R$+US	+	$\sim$	$\sim$	$-$	+	$\sim$	$\sim$	$\sim$	+	$\sim$	$\sim$	$-$
$\mathcal{MM}$AS+US $\Leftrightarrow$ Random+US	+	+	+	$\sim$	+	+	+	$\sim$	+	+	+	$\sim$

difference lies in that the pheromone trails in $\mathcal{MM}$AS+US provide a communication mechanism between the several restart points. Therefore, the search process is guided towards the promising areas of the search space.

Second, $\mathcal{MM}$AS+US performs better than $\mathcal{MM}$AS$_R$+US on DTSPs with $m = 0.1$ and $m = 0.25$; and they are comparable on DTSPs with $m = 0.5$ and $m = 0.75$. This is because when the changing environments are not similar $\mathcal{MM}$AS cannot adapt well as expected. The pheromone trails of the previous environment may not fit for the new environment hence the solution constructed by $\mathcal{MM}$AS and passed to US may lead the search process to a poor local optimum. This can be supported by the superior performance of $\mathcal{MM}$AS$_R$+US against $\mathcal{MM}$AS+US on DTSPs with $m = 0.75$; see the comparisons of $\mathcal{MM}$AS+US $\Leftrightarrow$ $\mathcal{MM}$AS$_R$+US in Table 2. In contrast, $\mathcal{MM}$AS+US significantly outperforms $\mathcal{MM}$AS$_R$+US in slightly changing environments, e.g., $m = 0.1$, where the adaptation via pheromone trails works as expected. $\mathcal{MM}$AS$_R$+US inherits the communication mechanism of $\mathcal{MM}$AS+US between the restart points as described above, but it is destroyed whenever a dynamic change occurs.

Third, $\mathcal{MM}$AS+US performs significantly better than $\mathcal{MM}$AS+2opt and $\mathcal{MM}$AS+3opt in all DTSPs. The strength of US procedure relies on the fact that it was designed to preserve or reverse subtours. Since ACO tends to find promising subtours (e.g., from the areas with high intensity of pheromone trails); when the US operator is applied to these solutions the promising subtours tend to be preserved whereas the remaining subtours are reversed. If we observe the US operator closely; taking the right choice for the cities and neighbourhood, we can see that it embodies 2-opt and 3-opt moves, but it outperforms both of them. In fact, US was also found to perform better than the Lin-Kernighan algorithm [12], which is one of the best performing algorithm for the TSP.

Fourth, the diversity is lower when local search operators are used (except on $\mathcal{MM}$AS$_R$+US where pheromone trails are reinitialized). This can be observed from Figure 6, where $\mathcal{MM}$AS maintains slightly higher diversity than $\mathcal{MM}$AS+2opt, $\mathcal{MM}$AS+3opt and $\mathcal{MM}$AS+US in almost all DTSPs. In contrast, Random+US always maintains higher diversity since random solutions are generated for the US operator. This shows that higher diversity may sometimes help in DTSPs with severely changing environments but may sometimes disturb the optimization in DTSPs with slightly changing environments.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we integrate an advanced local search operator (i.e., US) with the $\mathcal{MM}$AS for dynamic environments. The aim of the integration is to take the advantage of the adaptation capabilities of $\mathcal{MM}$AS and the solution improvement of the US operator. The performance of this specific memetic algorithm is investigated on different DTSPs gen-

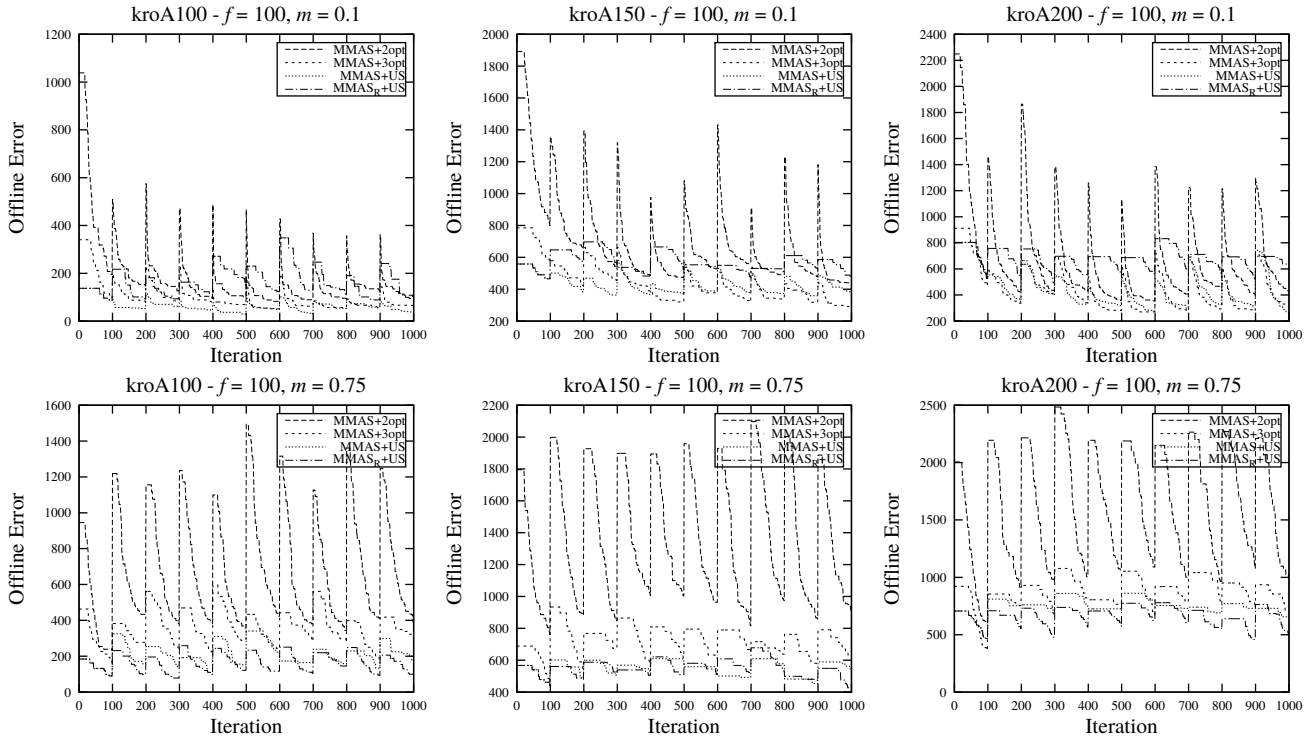

Figure 5: Dynamic offline error of ACO-based MAs in slowly changing DTSPs with $m = 0.1$ (top) and $m = 0.75$ (bottom).

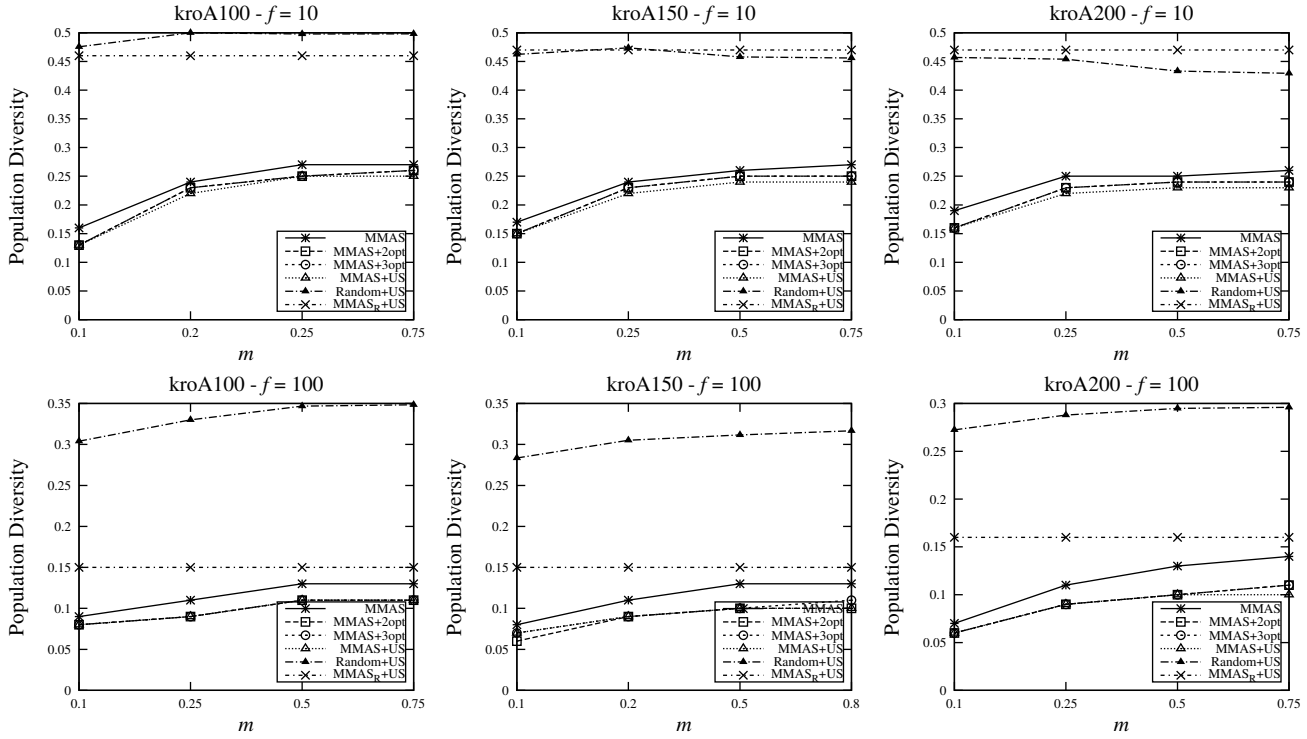

Figure 6: Population diversity of ACO algorithms for DTSPs with $f = 10$ (top) and $f = 100$ (bottom).

erated by the DBGP generator against other ACO-based memetic algorithms.

From the experimental results, the following conclusions can be drawn. First, local search operators improve signifi-

cantly the performance of $\mathcal{MM}$AS in DTSPs. Second, the integration of $\mathcal{MM}$AS with US outperforms other memetic algorithms in all DTSP cases. Third, $\mathcal{MM}$AS provides good initial solutions to US in cases where adaptation is convenient (e.g., slightly changing environments); otherwise, a restart of $\mathcal{MM}$AS provides better initial solutions (e.g., severely changing environments).

A straightforward future work is to investigate the performance of $\mathcal{MM}$AS+US in asymmetric DTSPs since it is claimed that it performs equally well for stationary symmetric and asymmetric TSPs [8, 9]. Another future work is to apply the algorithm into more challenging DOPs that model real-world scenarios (e.g., DTSPs with traffic factors).

7. ACKNOWLEDGMENTS

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1. Felipe M. Müller was partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior (CAPES), Brazil under grant BEX 2380/14-5.

8. REFERENCES

[1] D. Angus and T. Hendtlass. Ant colony optimisation applied to a dynamically changing problem. In T. Hendtlass and M. Ali, editors, *Developments in Applied Artificial Intelligence*, volume 2358 of *LNCS*, pages 618–627. Springer Berlin Heidelberg, 2002.

[2] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computing*, Natural Computing Series, pages 239–262. Springer Berlin Heidelberg, 2003.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. Vaerla and P. Bourgine, editors, *Proceedings of the European Conference on Artificial Life*, pages 134–142. Elsevier Publishing, 1991.

[4] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[5] 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*, 26(1):29–41, 1996.

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

[7] C. Eyckelhof and M. Snoek. Ant systems for a dynamic TSP. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms*, volume 2463 of *LNCS*, pages 88–99. Springer Berlin Heidelberg, 2002.

[8] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29(3):267–275, 1995.

[9] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43(2-3):79–89, 1996.

[10] L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *International Conference on Machine Learning*, pages 252–260, 1995.

[11] M. Garey and D. Johnson. *Computer and intractability: A guide to the theory of NP–completeness*. Freeman, San Francisco, 1979.

[12] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[13] M. Guntsch and M. Middendorf. Pheromone modification strategies for ant algorithms applied to dynamic TSP. In E. Boers, editor, *Applications of Evolutionary Computing*, volume 2037 of *LNCS*, pages 213–222. Springer Berlin Heidelberg, 2001.

[14] M. Guntsch and M. Middendorf. Applying population based ACO to dynamic optimization problems. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms*, volume 2463 of *LNCS*, pages 111–122. Springer Berlin Heidelberg, 2002.

[15] 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*, pages 860–867, 2001.

[16] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments–a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[17] M. Mavrovouniotis and S. Yang. A memetic ant colony optimization algorithm for the dynamic travelling salesman problem. *Soft Computing*, 15(7):1405–1425, 2011.

[18] M. Mavrovouniotis and S. Yang. Adapting the pheromone evaporation rate in dynamic routing problems. In A. Esparcia-Alcázar, editor, *Applications of Evolutionary Computation*, volume 7835 of *LNCS*, pages 606–615. Springer Berlin Heidelberg, 2013.

[19] M. Mavrovouniotis and S. Yang. Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing*, 13(10):4023–4037, 2013.

[20] M. Mavrovouniotis, S. Yang, and X. Yao. A benchmark generator for dynamic permutation-encoded problems. In C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature – PPSN XII*, volume 7492 of *LNCS*, pages 508–517. Springer Berlin Heidelberg, 2012.

[21] T. Stützle and H. Hoos. MAX–MIN ant system and local search for the traveling salesman problem. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pages 309–314, 1997.

[22] T. Stützle and H. H. Hoos. MAX–MIN ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[23] G. Tao and Z. Michalewicz. Inver-over operator for the TSP. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, volume 1498 of *LNCS*, pages 803–812. Springer Berlin Heidelberg, 1998.

[24] N. Ulder, E. Aarts, H.-J. Bandelt, P. van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *LNCS*, pages 109–116. Springer Berlin Heidelberg, 1991.