Empirical Study on the Effect of Population Size on $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System 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*—In this paper, the effect of the population size on the performance of the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for dynamic optimization problems (DOPs) is investigated. DOPs are generated with the dynamic benchmark generator for permutation-encoded problems. In particular, the empirical study investigates: a) possible dependencies of the population size parameter with the dynamic properties of DOPs; b) the effect of the population size with the problem size of the DOP; and c) whether a larger population size with less algorithmic iterations performs better than a smaller population size with more algorithmic iterations given the same computational budget for each environmental change. Our study shows that the population size is sensitive to the magnitude of change of the DOP and less sensitive to the frequency of change and the problem size. It also shows that a longer duration in terms of algorithmic iterations results in a better performance.**

I. Introduction

Ant colony optimization (ACO) is a metaheuristic inspired by the foraging behaviour of real ant colonies [2], [4]. ACO algorithms have been successfully applied to many $\mathcal{NP}$-hard combinatorial problems such as the travelling salesman problem (TSP) [3] and vehicle routing problem (VRP) [8]. Although, there are many existing ACO variations in this paper, we consider one of the state-of-the-art variations, i.e., the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [20].

The construction of solutions from ants is biased by existing pheromone trails and heuristic information. Pheromone trails are updated according to the search experience and towards solution with good quality. This is similar to a learning reinforcement scheme. The behaviour and performance of $\mathcal{MM}$AS algorithm depends strongly on the number of ants used [5], [22]. When a given computational budget is available, e.g., the maximum number of function evaluations, a smaller number of ants will produce more algorithmic iterations whereas a larger number of ants less. Hence, the population size affects the duration of the learning reinforcement.

In [22], it was investigated that when fewer ants are used, the algorithm may converge quickly at early stages of the optimization but get stuck in the stagnation behaviour later on. When more ants are used, the algorithm performs broader search but may waste computational resources. For the TSP, it was found that a higher number of ants performs better at later stages of the optimization process. However, the effect of the population size parameter on the performance of $\mathcal{MM}$AS algorithm was only investigated for stationary optimization problems.

In this paper, we investigate the effect of the population size parameter on the $\mathcal{MM}$AS algorithm for dynamic optimization problems (DOPs), e.g., the dynamic TSP (DTSP) and dynamic VRP (DVRP). In particular, we are interested to investigate: a) the dependency of the population size with the dynamic properties of a DOP, i.e., magnitude and frequency; b) the effect of the population size parameter with the problem size; and c) whether a broader search with less learning reinforcement time leads to better performance than a limited search with more learning reinforcement time given the same computation budget between environmental changes in DOPs. Several dynamic test cases are generated using the dynamic benchmark generator for permutation-encoded problems (DBGP) [16] for our study.

The rest of the paper is organized as follows. Section II and Section III describe the DOPs generated and the ACO algorithm used for this study, respectively. Section IV discusses the importance of the population size parameter. Section V presents the experimental study and gives a discussion. Finally, Section VI concludes this paper.

II. Dynamic Environments

A. Dynamic Optimization Problems

1) Dynamic Travelling Salesman Problem (DTSP): The DTSP is modelled by a fully connected weighted graph $G = (N, A)$, where $N = \{v_1, \ldots, v_n\}$ is a set of n nodes (e.g., cities) and $A = \{(v_i, v_j) \mid v_i, v_j \in N, i \neq j\}$ is a set of arcs (i.e., links), where n represents the size of a problem instance. Each arc $(v_i, v_j) \in A$ is associated with a non-negative value $d_{ij} \in \mathbb{R}^+$, which represents the distance between cities v_i and v_j. The objective of the problem is to find the shortest

Hamiltonian cycle that starts from one node and visits each of the other cities once before returning to the starting city.

The distance matrix of the DTSP is subject to changes, which is defined as follows: $\mathbf{D}(t) = \{d_{ij}(t)\}_{n\times n}$, where t is the period of a dynamic change. A particular TSP solution $s=[s_1,\ldots,s_n]$ in the search space is specified by a permutation of the nodes (cities) and it is evaluated as follows:

$$f(s,t) = d_{s_n s_1}(t) + \sum_{i=1}^{n-1} d_{s_i s_{i+1}}(t). \quad (1)$$

2) Dynamic Vehicle Routing Problem (DVRP): The DVRP is modelled with the same weighted graph G as with the DTSP above but with an additional node v_0 that represents the central depot such that $N = \{v_1,\ldots,v_n\}\cup\{v_0\}$, where A and $\mathbf{D}(t)$ remain the same. In addition, each node (customer) $v_i \in N$ is associated with a quantity q_i of some goods that need to be delivered by K vehicles. The objective of the problem is to find the shortest routes for a fleet of K vehicles with capacity Q all starting from and ending at the depot satisfying the demands of all customers. For the central depot v_0, the quantity is always $q_0 = 0$.

A particular VRP solution s in the search space is specified by a permutation of nodes. More precisely, let R_j represent a route of the jth vehicle that service particular customers $R_j = [s_1^j,\ldots,s_z^j]$, where s_1^j is always the central depot, i.e., $s_1^j = v_0$. A complete solution is defined with the routes of all K vehicles, i.e., $s = [R_1, R_2, \ldots, R_K]$. The cost of a single route of the jth vehicle at time t is computed as follows:

$$Cost(R_j,t) = d_{s_z^j s_1^j}(t) + \sum_{i=1}^{z-1} d_{s_i^j s_{i+1}^j}(t), \quad (2)$$

subject to

$$\sum_{s_i^j \in R_j} q_i \le Q. \quad (3)$$

The complete solution s in the search is evaluated as follows:

$$f(s,t) = \sum_{j=1}^{K} Cost(R_j,t). \quad (4)$$

B. Dynamic Benchmark Generators

Over the years, several dynamic benchmark generators have been proposed for these problems that tend to model real-world scenarios, such as the DTSP with traffic factors [7], [15], [17], the DTSP with exchangeable cities [9], [10], the DVRP with traffic factors [12], [13], and the DVRP with stochastic demands [18]. These benchmark generators modify the fitness landscape, whenever a dynamic change occurs, and cause the optimum value to change.

In this paper, DBGP [1] is used [16], 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. The main advantage of using the DBGP rather than the other generators is that one can observe how close to the optimum an algorithm can perform when a dynamic change occurs. However, DBGP sacrifices the realistic modelling of application problems for the sake of benchmarking.

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

Algorithm 1 DBGP(m,f)

```
INPUT: m          % magnitude of change
INPUT: f          % frequency of change
Read initial benchmark problem of size n
t ← 0
ComputeDistances(D(t))
repeat
   DoOptimization(t) % e.g., with MMAS
   t ← t + 1
   if (f%t = 0) then
      num_of_swaps ← ⌈m × n⌉
      for (i = 1 to num_of_swaps) do
         V[i] ⊗ U[i]
      end for
      ComputeDistances(D(t))
   end if
until (optimization not terminated)
```

C. Construct Dynamic Test Environment

Considering the description of the problems above, each node $i \in N$ has a location defined by (x,y) and each arc $(v_i,v_j)\in A$ is associated with a non-negative distance $d_{ij}(t)$ at time t. DBGP generates the dynamic case as follows.

In every function evaluations f, a random vector $\vec{V}(T)$ is generated that contains exactly $\lceil mn\rceil$ nodes where $T = \lceil t/f\rceil$ is the index of the period of change, t is the evaluation 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 of $\vec{V}(T)$ is generated, denoted $\vec{U}(T)$, of the same size. In this way, exactly $\lceil mn\rceil$ pairwise swaps are performed and affect the distance matrix (e.g., $\mathbf{D}(t)$) using the two random vectors $(\vec{V}(T)\otimes\vec{U}(T))$, where "$\otimes$" denotes the swap operator. The pseudo code is presented in Algorithm 1.

III. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

A. Construct Solutions

One of the best performing ACO variations is the $\mathcal{MM}$AS [20]. A colony of ω 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 node to visit. The decision rule of the kth ant to move from node v_i to node v_j is defined as follows:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l\in\mathcal{N}_i^k}[\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j\in\mathcal{N}_i^k, \quad (5)$$

where τ_{ij} and η_{ij} are the existing pheromone trail and the heuristic information available a priori between nodes v_i and

v_j, respectively. The heuristic information is defined as $\eta_{ij} = 1/d_{ij}(t)$, where $d_{ij}(t)$ is defined as in Eq. (1). $\mathcal{N}_i^k$ is the neighbourhood of unvisited nodes available for ant k to select. The main difference between constructing a TSP or a VRP solution lies in the generation of $\mathcal{N}_i^k$. For the TSP, $\mathcal{N}_i^k$ is generated by the unvisited nodes incident to node i, whereas for the VRP, $\mathcal{N}_i^k$ is generated by the unvisited nodes incident to node i in addition with the depot node (i.e. $\{0\}$). α and β are the two parameters which determine the relative influence of τ_{ij} and η_{ij}, respectively.

B. Pheromone Update

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

$$\tau_{ij} \leftarrow (1-\rho)\,\tau_{ij}, \forall (v_i, v_j), \tag{6}$$

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:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall (v_i, v_j) \in T^{best}, \tag{7}$$

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 and the iteration best ant, respectively. The best-so-far ant is a special ant that may not necessarily belong to the current population of ants as the iteration best ant. Both update rules are used in an alternate way in the implementation [21].

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 $\tau_{min} = \tau_{max}/2n$.

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 occurs or when no improved solution is found for a given number of iterations, the pheromone trails are reinitialized.

C. Response to Dynamic Changes

$\mathcal{MM}$AS is able to use knowledge from previous environments via pheromone trails and can be applied directly to DOPs without any modifications [1], [14]. 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.

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], [9], [10], [14]. 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 [14].

IV. Effect of the Number of Ants

The population size has a significant impact to the performance of the $\mathcal{MM}$AS algorithm. For example, if for a given optimization problem only a certain computation budget, e.g., the maximum number of function evaluations, is available, then the number of ants is a very critical parameter. This is because it not only determines the number of iterations (e.g., less ants means more iterations), but also determines how broad the search is at each iteration (e.g., more ants means broader search). Hence, the number of ants needs to be tuned accordingly in order not to waste computation resources and degrade solution quality.

Up to now, the effect of the number of ants was only investigated for stationary optimization problems [22]. In particular, the number of ants used shows a trade-off between the early and later optimization process of the algorithm regarding the solution quality. At early stages of the optimization process fewer ants result to better performance, whereas at later stages more ants result to better performance. With fewer ants the algorithm seems to initially progress faster but leads to stagnation behaviour at later stages. More ants give better results only on later stages of the optimization process.

In this paper, we study the impact of the population size on the performance of the $\mathcal{MM}$AS algorithm for DOPs. These kind of problems in a nutshell are a series of stationary optimization problems that all need to be optimized. Therefore, it is straightforward that more challenges exist and the population size will have impact to the performance of the algorithm. This is because that not only determines the number of iterations and the broadness of the search as in stationary optimization problems, but also determines how quickly the change occurs (in terms of algorithmic iterations). For example, for a given DOP a predefined computation budged is available between each environmental change that is typically sychronized with the algorithm, i.e., every f function evaluations a change occurs [16]. Therefore, a larger population size means that the algorithm will perform less iterations for each environmental change.

In summary, the experimental study investigates: a) the dependency of the population size with the dynamic properties

TABLE I: Resulting population sizes (ω) of $\mathcal{MM}$AS for each problem instance scaled according to the problem size as defined in Eq. (8)

Problem Instance	n	Optimal Value	$\delta = 1$	$\delta = 2$	$\delta = 5$	$\delta = 10$	$\delta = 25$	$\delta = (n/2)$	$\delta = n$
kroA100.tsp	100	21282	100	50	20	10	4	2	1
kroA150.tsp	150	26524	150	75	30	15	6	2	1
kroA200.tsp	200	29368	200	100	40	20	8	2	1
F-n45-k4.vrp	45†	724	50	20	15	8	2	2	1
F-n72-k4.vrp	72†	237	100	50	20	10	4	2	1
F-n135-k7.vrp	135†	1162	150	75	30	15	6	2	1

†*Note* that the resulting values are higher for the VRP instances to match the scales of the TSP problem instances

of a DOP, i.e., magnitude and frequency; b) the effect of the population size parameter with the problem size, e.g., whether more ants are needed for larger problem instances; and c) whether a broader search with less learning reinforcement time leads to better performance than a limited search with more learning reinforcement time given the same computation budget between environmental changes in DOPs.

V. Experimental Study and Discussion

A. Experimental Setup

To investigate the effect of the population size of $\mathcal{MM}$AS in dynamic environments, three TSP stationary benchmark instances (i.e., kroA100.tsp, kroA150.tsp and kroA200.tsp) were obtained from TSPLIB[2] and three VRP stationary benchmark instances (i.e., F-n45-k4.vrp, F-n72-k4.vrp and F-n135-k7.vrp) were obtained from VRPLIB[3], and corresponding DOPs are generated using the DBGP generator (described in Section II) with f set to 600 and 6000 function evaluations, indicating quickly and slowly changing environments, respectively, and m set to 0.1, 0.25, 0.5 and 0.75, indicating slightly, to medium, to severely changing environments, respectively. The problem size and global optimum values of the benchmark problem instances are given in Table I. Totally, a series of 8 dynamic test cases (or DOPs) are constructed from each stationary benchmark instance for both DTSPs and DVRPs to systematically investigate the dependency (if any) of the population size with the m and f parameters.

The population size was set proportionally to the size of the problem instances as follows:

$$\omega = \lceil n/\delta \rceil, \tag{8}$$

where n is the problem size and δ defines the factor which the population will be determined. For all problem instances, δ was set to $\delta \in \{1, 2, 5, 10, 25, (n/2), n\}$. This means that when $\delta = 1$ the population size will be equal to the problem size, whereas when $\delta = n$ the population size will be equal to one. The resulting population sizes of $\mathcal{MM}$AS for all problem instances with the problem sizes (i.e., n) are given in Table I. The remaining parameters were set to typical values for DOPs as follows: $\alpha = 1$, $\beta = 5$ and $\rho = 0.8$.

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

[3]http://neo.lcc.uma.es/vrp/

B. Performance Measurement

For each DOP, 30 independent runs of the $\mathcal{MM}$AS were executed. For each run, 25 environments changes were allowed and the best so far ant after a dynamic change was recorded. The overall *offline error* [11] is defined as follows:

$$\bar{E}_{OFF} = \frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{R}\sum_{j=1}^{R} Err_{ij}\right), \tag{9}$$

where E is the total number of function evaluations, R is the number of runs, 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) of iteration i of run j. Note that this measurement is compatible with DBGP because the optimal value of each benchmark instance is known (given in Table I) and remains the same during the environmental changes.

C. Results and Discussion

The offline error results of the $\mathcal{MM}$AS algorithm on DOPs with different population sizes are presented in Table II. Statistical Kruskal–Wallis tests were applied, followed by posthoc paired comparisons using Mann–Whitney tests with the Bonferroni correction. The algorithms with bold values indicate that are significantly better than the other algorithms and insignificantly difference between them. In Figs. 1 and 2, the dynamic offline error for slowly changing environments against the function evaluations of the algorithms are plotted to better understand the effect of different number of ants. From the experimental results, the following observations can be drawn.

First, the performance of $\mathcal{MM}$AS with different population sizes generally depends on the dynamic properties of the DOP. This can be observed from Table II since different population sizes perform better in different dynamic test cases. More precisely, in all problem instances (except F-n45-k4.vrp) a larger population size has better performance in DOPs with $m = 0.1$ and $m = 0.25$ whereas a smaller population has better performance in DOPs with $m = 0.5$ and $m = 0.75$. This can be observed from Figs. 1 and 2, where in all cases with $m = 0.1$, the $\mathcal{MM}$AS with more ants, i.e., $\delta = [1, 5]$, has better performance. Differently, in all cases (except F-n45-k4.vrp) with $m = 0.75$, the $\mathcal{MM}$AS with less ants, i.e., $\delta = [10, 25]$, has better performance.

TABLE II: Experimental results regarding the offline error of $\mathcal{MM}$AS with different population sizes. Bold values indicate the best results.

Dynamic Travelling Salesman Problem												
Algorithms & DOPs	kroA100.tsp				kroA150.tsp				kroA200.tsp			
$f = 600$, $m \Rightarrow$	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 ($\delta = 1$)	1165.7	3014.4	4303.1	4742.0	3284.8	5876.0	7093.5	7394.3	5230.3	7828.4	9121.4	9341.3
$\mathcal{MM}$AS ($\delta = 2$)	929.2	2367.5	3546.6	3947.1	2443.1	4767.9	5993.1	6368.7	3708.5	6722.2	7983.0	8315.2
$\mathcal{MM}$AS ($\delta = 5$)	**827.2**	1921.2	2817.2	3183.5	1938.0	3830.2	4820.9	5232.3	2660.7	5374.2	6601.8	6999.0
$\mathcal{MM}$AS ($\delta = 10$)	**783.6**	**1661.0**	2452.4	2762.7	**1769.4**	3233.6	4153.7	4509.9	**2367.9**	4548.7	5668.0	6131.3
$\mathcal{MM}$AS ($\delta = 25$)	1172.2	1823.2	**2332.6**	**2561.7**	**1692.7**	**3012.7**	3739.4	3980.7	**2138.6**	**3966.6**	4914.0	5300.7
$\mathcal{MM}$AS ($\delta = (n/2)$)	1875.3	2169.9	2449.5	**2562.4**	2920.9	3319.2	**3617.5**	**3766.7**	3346.7	**3937.5**	**4431.7**	**4788.5**
$\mathcal{MM}$AS ($\delta = n$)	2340.0	2486.4	2638.1	2730.8	3455.3	3640.8	3868.6	3988.0	4153.9	4446.0	4815.5	4966.9
$f = 6000$, $m \Rightarrow$	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 ($\delta = 1$)	**227.4**	535.2	980.8	1211.4	659.8	1380.0	2190.4	2434.1	663.5	1957.4	3199.7	3678.4
$\mathcal{MM}$AS ($\delta = 2$)	**228.7**	**448.1**	794.5	979.5	**610.7**	1171.5	1737.9	1945.3	**622.0**	1432.3	2419.0	2771.6
$\mathcal{MM}$AS ($\delta = 5$)	404.5	491.0	693.9	817.5	**553.0**	**972.8**	1437.7	1595.8	**530.9**	**1099.8**	1879.2	2115.8
$\mathcal{MM}$AS ($\delta = 10$)	399.1	**472.8**	**628.8**	**716.9**	810.9	**1017.9**	**1320.3**	1413.6	802.4	**1157.4**	1597.4	1757.0
$\mathcal{MM}$AS ($\delta = 25$)	411.1	521.2	653.1	**704.6**	828.6	1087.9	**1293.8**	**1364.5**	773.5	**1111.2**	**1508.8**	**1634.1**
$\mathcal{MM}$AS ($\delta = (n/2)$)	807.6	847.8	888.8	909.4	1428.3	1554.1	1605.2	1612.1	1478.5	1660.3	1761.2	1816.4
$\mathcal{MM}$AS ($\delta = n$)	1290.7	1317.6	1317.8	1327.9	2147.3	2200.4	2206.9	2208.4	2510.5	2533.8	2597.0	2616.8
Dynamic Vehicle Routing Problem												
Algorithms & DOPs	F-n45-k4.vrp				F-n72-k4.vrp				F-n135-k7.vrp			
$f = 600$, $m \Rightarrow$	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 ($\delta = 1$)	**87.9**	**100.0**	**108.3**	**112.9**	**27.7**	40.4	46.4	50.4	146.6	200.4	229.8	241.6
$\mathcal{MM}$AS ($\delta = 2$)	99.7	103.9	**110.5**	**114.4**	**27.4**	37.8	42.8	46.5	**135.8**	**187.5**	217.9	227.2
$\mathcal{MM}$AS ($\delta = 5$)	98.8	107.5	**110.3**	116.9	**27.2**	**35.5**	**40.1**	43.2	**143.4**	**185.4**	208.2	**216.3**
$\mathcal{MM}$AS ($\delta = 10$)	105.1	111.4	115.7	120.5	29.2	**35.1**	**39.2**	**41.0**	145.6	**184.0**	**204.2**	**212.0**
$\mathcal{MM}$AS ($\delta = 25$)	122.0	125.9	128.0	130.1	34.8	37.4	**39.9**	**41.5**	169.7	193.9	209.8	**214.9**
$\mathcal{MM}$AS ($\delta = (n/2)$)	122.0	125.9	128.0	130.1	39.9	40.7	42.4	43.9	198.4	207.9	216.4	220.4
$\mathcal{MM}$AS ($\delta = n$)	129.3	131.5	131.2	132.5	44.8	44.7	45.8	46.3	203.8	206.9	211.1	**212.2**
$f = 6000$, $m \Rightarrow$	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 ($\delta = 1$)	**53.5**	**65.5**	**69.3**	**71.2**	**13.2**	**19.9**	22.6	24.8	**77.8**	109.4	127.7	130.3
$\mathcal{MM}$AS ($\delta = 2$)	64.2	**66.9**	72.2	73.8	**13.6**	**19.4**	**21.8**	23.1	**79.2**	**102.2**	121.7	123.1
$\mathcal{MM}$AS ($\delta = 5$)	65.2	70.5	73.1	74.1	17.6	**19.6**	**21.3**	**22.2**	**80.1**	**102.4**	**117.0**	**118.8**
$\mathcal{MM}$AS ($\delta = 10$)	74.6	77.4	77.3	78.9	19.0	**20.1**	**21.5**	**21.7**	95.8	108.0	**119.0**	**120.4**
$\mathcal{MM}$AS ($\delta = 25$)	86.7	87.8	87.2	87.8	22.7	22.5	23.1	23.2	114.7	121.0	127.3	128.5
$\mathcal{MM}$AS ($\delta = (n/2)$)	86.7	87.8	87.2	87.8	27.4	26.7	27.0	27.4	145.2	146.2	146.8	147.9
$\mathcal{MM}$AS ($\delta = n$)	91.3	93.1	92.6	93.6	32.7	31.6	32.0	32.0	152.0	153.1	154.3	153.0

Second, for the F-n45-k4.vrp problem instance (which is the smallest), a larger population size, i.e., $\delta = 1$, results in better performance for $\mathcal{MM}$AS. This is probably because the problem size is small and a broader search (e.g., with a larger number of ants) has higher probability to locate a good optimum. On the remaining problem instances, it can be observed that a similar number of ants performs best in DOPs of different sizes. For example, $\mathcal{MM}$AS with $\delta = n/2$ (meaning the number of ants is 2 for all problem sizes) has the best performance on the kroA150.tsp and kroA200.tsp problem instances for the same DOPs, i.e., $f = 600$ with $m = 0.5$ and $m = 0.75$.

Third, it can be observed from Table II that when a single ant is used with $\mathcal{MM}$AS, i.e., when $\delta = n$, the algorithm has the worst performance in most DOPs with $f = 6000$. This is probably because there is no collaboration between ants to exchange information. In contrast, even when a single ant is used it has better performance when compared with most cases of $\mathcal{MM}$AS that use more ants in most DOPs with $f = 600$. This is because with more ants the $\mathcal{MM}$AS will perform less iterations for each dynamic change as discussed previously. Hence, the time for learning reinforcement is not enough to express its effect.

VI. Conclusion

$\mathcal{MM}$AS is one of the state-of-the-art ACO algorithms. The number of ants has a great impact to the performance in stationary environments. In this paper, the impact of the number of ants is investigated for dynamic environments.

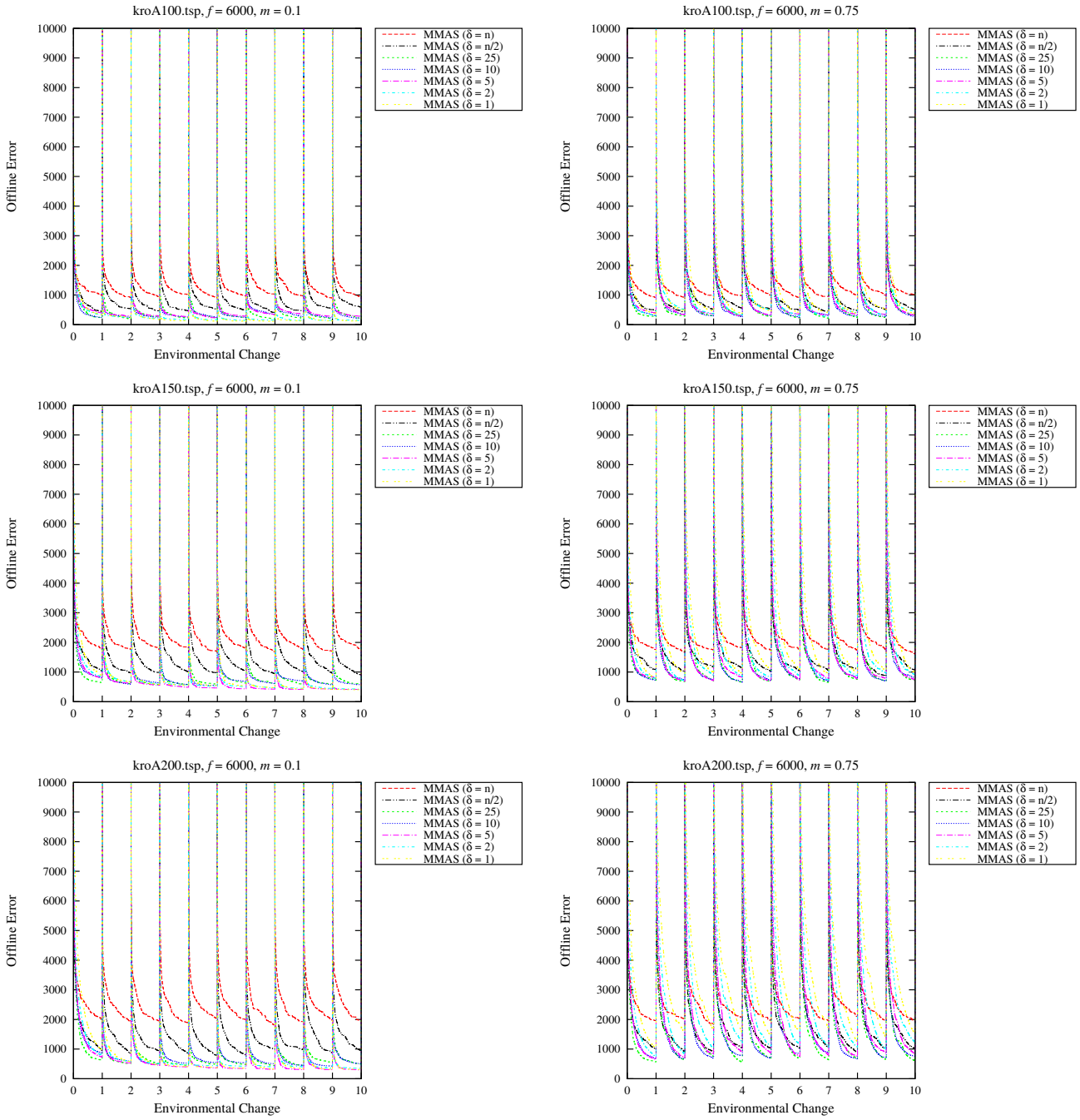

Fig. 1: Dynamic offline error of $\mathcal{MM}$AS with different population sizes for DTSPs with $f = 6000$ and $m = 0.1$ (left) and $m = 0.75$ (right)

Several dynamic test cases are generated using the DBGP and the following concluding remarks can be drawn.

First, the population size parameter of $\mathcal{MM}$AS is sensitive to the magnitude of change of the DOP: as the magnitude of change increases, the number of ants must decrease. Second, a high number of ants wastes computational resources in many DOPs, especially when they change quickly. Third, the number of ants is less sensitive on the problem size of the DOP. Finally, the learning reinforcement requires time to express its effect since fewer ants mean more algorithmic iterations.

In fact, the findings of this paper, i.e., fewer ants may perform better even in large problem sizes, are important and related to many real-world problems. This is because many objective functions for such problems may require a lot of time to compute [6], [19]. Hence, using a smaller population size may be appropriate in such situations to reduce the computation time and maintain the solution quality.

For future work, it would be interesting to investigate the

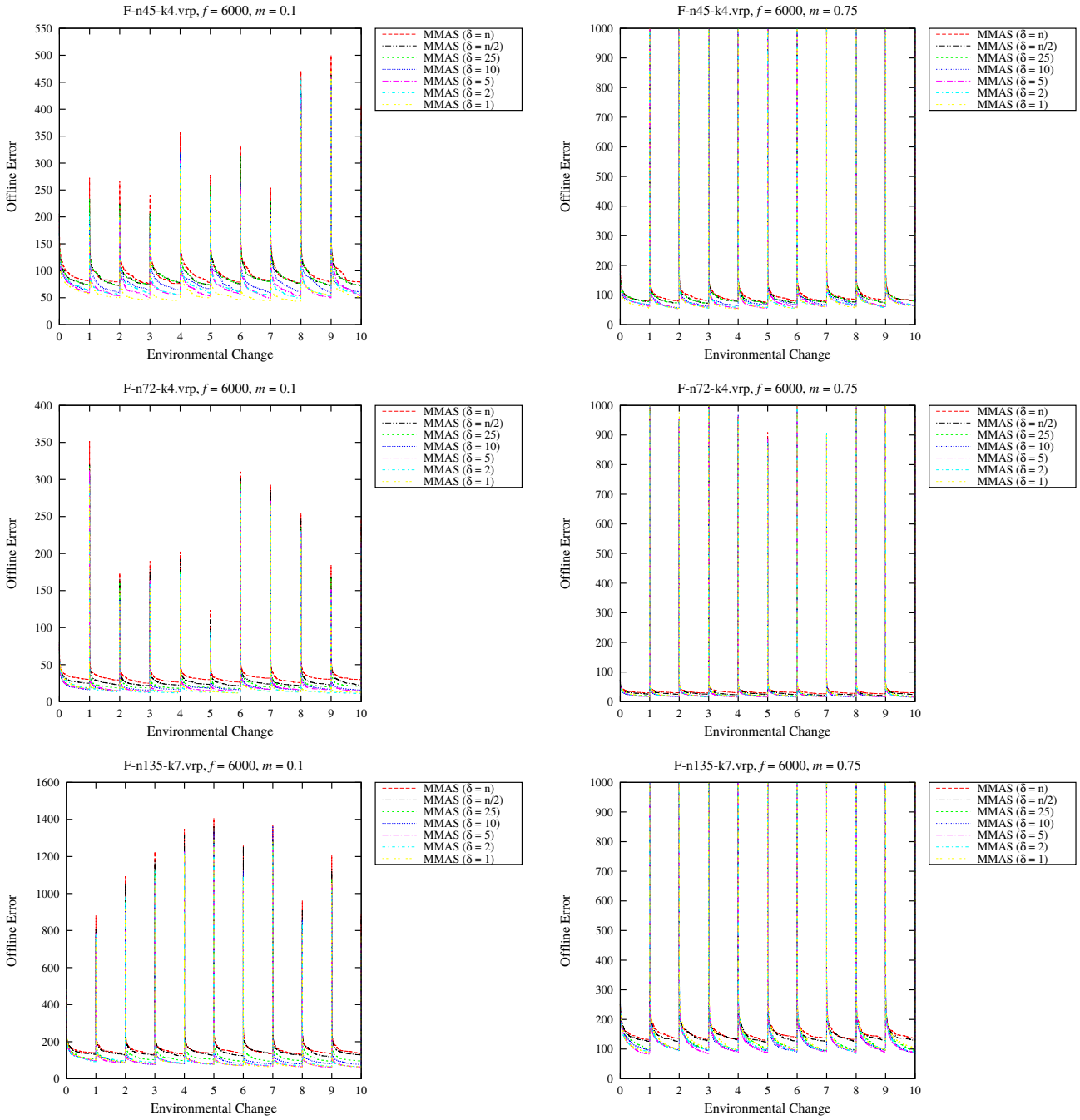

Fig. 2: Dynamic offline error of $\mathcal{MM}$AS with different population sizes for DVRPs with $f = 6000$ and $m = 0.1$ (left) and $m = 0.75$ (right)

impact of the population size with other evolutionary algorithms for dynamic environments. Another interesting work is to adapt the population size since for different optimization stages a different population size may be the best.

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 optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, vol. 2358, pp. 618–627.

[2] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. 1st European Conf. on Artificial Life*, 1991, pp. 134–142.

[3] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[4] 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.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[6] J. Eaton, S. Yang, and M. Mavrovouniotis, "Ant colony optimization with immigrants schemes for the dynamic railway junction rescheduling problem with multiple delays," *Soft Computing*, pp. 1–16, 2015.

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

[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] 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, Springer Berlin Heidelberg, 2001, vol. 2037, pp. 213–222.

[10] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," in *Ant Algorithms*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, vol. 2463, pp. 111–122.

[11] 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.

[12] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, vol. 7248, pp. 519–528.

[13] M. Mavrovouniotis and S. Yang, "Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem," in *Proc. 2012 IEEE Congr. on Evol. Comput.*, 2012, pp. 2645–2652.

[14] M. Mavrovouniotis and S. Yang, "Adapting the pheromone evaporation rate in dynamic routing problems," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, vol. 7835, pp. 606–615.

[15] M. Mavrovouniotis and S. Yang, "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.

[16] M. Mavrovouniotis, S. Yang, and X. Yao, "A benchmark generator for dynamic permutation-encoded problems," in *Parallel Problem Solving from Nature - PPSN XII*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, vol. 7492, pp. 508–517.

[17] 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, Springer Berlin Heidelberg, 2013, vol. 7824, pp. 179–188.

[18] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *Combinatorial Optimization*, vol. 10, pp. 327–343, 2005.

[19] 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.

[20] T. Stützle and H. Hoos, "$\mathcal{MAX}$-$\mathcal{MIN}$ ant system and local search for the traveling salesman problem," in *Proc. IEEE Int. Conf. on Evol. Comput.*, 1997, pp. 309–314.

[21] T. Stützle and H. H. Hoos, "$\mathcal{MAX}$-$\mathcal{MIN}$ ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889 – 914, 2000.

[22] 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.