①

2016

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem

I.D.I.D. Ariyasingha [a,*], T.G.I. Fernando [b]

[a] Department of Mathematics and Computer Science, Faculty of Natural Sciences, Open University of Sri Lanka, Sri Lanka
[b] Department of Computer Science, Faculty of Applied Science, University of Sri Jayewardenepura, Sri Lanka

ARTICLE INFO

Article history:
Received 2 February 2014
Received in revised form
18 February 2015
Accepted 20 February 2015

Keywords:
Ant colony optimization
Multi-objective problem
Non-dominated solution
Pareto optimal front
Performance indicator
Traveling salesman problem

ABSTRACT

Most real world combinatorial optimization problems are difficult to solve with multiple objectives which have to be optimized simultaneously. Over the last few years, researches have been proposed several ant colony optimization algorithms to solve multiple objectives. The aim of this paper is to review the recently proposed multi-objective ant colony optimization (MOACO) algorithms and compare their performances on two, three and four objectives with different numbers of ants and numbers of iterations. Moreover, a detailed analysis is performed for these MOACO algorithms by applying them on several multi-objective benchmark instances of the traveling salesman problem. The results of the analysis have shown that most of the considered MOACO algorithms obtained better performances for more than two objectives and their performance depends slightly on the number of objectives, number of iterations and number of ants used.

1. Introduction

Ant Colony Optimization (ACO) was introduced by Dorigo and Stutzle in the early 1990s which is based on the behavior of natural ant colonies in particular, the foraging behavior of real ant species [1]. The indirect communication of real ants in the colony uses pheromone trail laying on the ground to find the shortest path between their food source and the nest. This procedure of real ant species in the colony is exploited by artificial ants. Moreover, ACOs are becoming popular approaches for solving combinatorial optimization (CO) problems such as the traveling salesman problem, job shop scheduling problem and quadratic assignment problem. Recently, ACO algorithms have been proposed to solve multi-objective problems (MOACO algorithms). Most of these algorithms find pareto optimal solutions and this characteristic makes ACO very attractive to solve multi-objective optimization problems.

The aim of this study is to review the recent MOACO, including pareto strength ant colony optimization (PSACO), and study the performances of those algorithms by comparing them. These MOACO algorithms have been applied to the travelling salesman problem (TSP) and six TSP problem instances have been considered to solve two, three and four objectives by changing the number of ants and number of iterations. A detailed analysis has been developed to analyze each of the MOACO algorithms by considering some of the performance indicators.

The remainder of this paper is structured as follows: some preliminaries about the multi-objective optimization problem, travelling salesman problem and ACO algorithms are reviewed in Section 2. Section 3 introduces the recent MOACO algorithms. The experimentation with the adaptation of MOACO algorithms to the traveling salesman problem, performance indicators, problem instances and parameter settings are presented in Section 4. In Section 5, the experimental results of the study are analyzed. Section 6 provides some concluding remarks.

2. Preliminaries

2.1. Multi objective optimization problem

Many real world problems consist more than one objective functions which are to be minimized or maximized simultaneously [2]. Single objective optimization problems find only one solution. However, multi-objective optimization problems find a set of optimal solutions. Generally, the multi-objective optimization can be presented as follows:

$$\left.\begin{aligned} & y=f(x)=[f_1(x),f_2(x),\dots,f_m(x)],\\ & e(x)=[e_1(x),e_2(x),\dots,e_k(x)]\geq 0,\\ & x=(x_1,x_2,\dots,x_n)\in X\\ & y=(y_1,y_2,\dots,y_m)\in Y \end{aligned}\right\} \quad (1)$$

* Corresponding author.
E-mail address: ireshaw82@gmail.com (I.D.I.D. Ariyasingha).

http://dx.doi.org/10.1016/j.swevo.2015.02.003
2210-6502/© 2015 Elsevier B.V. All rights reserved.

where X denotes the decision space of a set of decision variables n and the objective space is denoted by Y of a set of objective functions m with k restrictions.

2.2. Traveling salesman problem

The traveling salesman problem (TSP) is an extensively studied combinatorial optimization problem by computer scientists and mathematicians. In TSP, a salesman starts from his home city and returns to the starting city by visiting each city exactly once to finding the shortest path between a given set of cities [1]. To represent the TSP, a complete weighted graph $G=(N,E)$ can be used, where a set of nodes N represents the cities and E is the set of arcs which has fully connected the nodes. A value d_{ij} is assigned for each arc $(i,j) \in E$ to represents the distance between nodes i and j. The distances between the cities are independent in the symmetric TSPs for every pair of nodes, that is, $d_{ij}=d_{ji}$. In the asymmetric TSP (ATSP), distances between the cities are not independent for at least one pair of nodes, which means, $d_{ij} \neq d_{ji}$.

2.3. Ant colony optimization algorithm

Ant colony optimization (ACO) is a meta-heuristic which has been emerged recently, for solving hard combinatorial optimization problems [1]. ACO is based on the characteristics of the real ant colonies. Ants in the colony find the shortest path for gathering food by frequently travelling between their nest and food source. When ants move between nest to the food source, they deposit a chemical called *pheromone trails* on the path which can be followed by other ants to find the shortest path to the food source. If no more pheromone is laid down the pheromone trail evaporates over time. This indirect communication behavior of real ants is based on the artificial ants.

Artificial ants find solutions by transiting from one node to another. Also, they use special data structure which stores in the memory to keep their previous actions and it is used when ants move from one node to another. Each path has constant amount of pheromone when an ant starts its journey from the first node. After the ant completes its tour by visiting the first node to the last node, the pheromone trail of all paths are updated. If the completed path by the ant is a good path, then the pheromone trail of that path will be high and vice versa. Also pheromone trails are evaporated in each path before applying the new pheromone trail. Furthermore, when the ant moves, it considers *heuristic information* which measures the quality of the given problem.

2.3.1. Ant system

Dorigo et al. [3] proposed the first ant colony optimization algorithm, Ant System (AS) for solving stochastic combinatorial optimization problems. This approach was applied to the classical traveling salesman problem and also to the asymmetric traveling salesman problem, the quadratic assignment problem and the job shop scheduling problem. All the pheromone values are associated with edges and the initial pheromone value of each edge set is equal to the given value τ_0. The heuristic information η set to $1/d_{ij}$, where d_{ij} represents the distance between the city i and j. Initially, m ants are placed into the randomly selected cities. Thereafter, every ant k moves from city i to city j using the probability given in the following equation:

$$P_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in N_i^k}[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The relative importance of the pheromone trail and the heuristic information are represented by the parameters α and β, respectively. N_i^k is the feasible neighborhood of ant k in city i. After n iterations all the ants have completed a tour, the pheromone trails are updated. First the pheromone trail is evaporated and then pheromones are deposited on arcs that ants have visited as in the following equation:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

The pheromone evaporation rate denotes by $\rho (0 \le \rho < 1)$ and the amount of pheromones deposited by ant k on arc (i,j) denotes by $\Delta\tau_{ij}^{k}$. In AS $\Delta\tau_{ij}^{k}$ is defined as follows:

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k} \tag{4}$$

where Q is a constant and L_k being the total length of the tour of the k-th ant.

2.3.2. Ant colony system

Dorigo and Gambardella [4] introduced the ant colony system (ACS) which is based on the ant system (AS) algorithm and it has improved the efficiency of AS when applied to the traveling salesman problem. Artificial ants of ACS use parallel searching procedure to find better solutions for small TSP instances. ACS identified three main modifications which differ from the Ant System.

(i) Each ant in ACS uses the state transition rule to select the next node to be visited as given in the following equation:

$$j = \begin{cases} \arg\max_{j \in N_i^k} [\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases} \tag{5}$$

where q is a random number selects in [0, 1] and $q \in [0, 1]$ is a parameter. J is a random value calculated using the probability distribution given by Eq. (2). β is a parameter which represents the relative importance of pheromone information versus distance. When consider the Eqs. (2) and (5) together it is called pseudo-random-proportional rule.

(ii) After all ants completed their tour, the global updating rule is applied for the best ant tour from the beginning of the trail and deposit the pheromones using the following equation:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best} \tag{6}$$

The best ant deposits the pheromone $\Delta\tau_{ij}^{best} = 1/L_{gb}$, which generates the best solution. Where the length of the best ant tour from the beginning of the run denotes as L_{gb}.

(iii) After an ant moves from one node to another, the local updating rule is performed for the pheromone trail of the edge as in the following equation:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{7}$$

where $\Delta\tau_{ij} = \tau_0$. τ_0 is the initial pheromone trail of the edge.

3. Multi objective ant colony optimization algorithms

3.1. Pareto strength ant colony optimization (PSACO)

Thantulage [5] introduced pareto strength ant colony optimization (PSACO) algorithm which is based on the first ant colony algorithm, ant system (AS). For all of the objectives it uses a same pheromone matrix while pheromone trail is updated using the domination concept as in SPEA II [6]. PSACO algorithm has been extended to solve multi-objective problems. When an ant moves from one node to another it uses the random propositional rule as defined in ant system (AS) algorithm (Eq. (2)).

Pheromone updating procedure is the major change in this algorithm. For each iteration t, PSACO algorithm maintains two sets of solutions; population P_t and archive A_t. Solutions produced by the current iteration is kept in the set, population P_t. Also, archive A_t contains a fixed number of globally best non-dominated solutions. If the size of the archive is greater than the number of globally best non-dominated solutions, then the archive A_t fills up by the current best dominated solutions. A strength value is assigned for the number of solutions dominated by each individual in the population P_t and the archive A_t as in the following equation:

$$S(i) = |\{j | j \in P_t \cup A_t \wedge i \succ j\}| \tag{8}$$

where the cardinality of a set is denoted by $|\cdot|$ and $i \succ j$ represent that solution i dominates solution j. On the basis of this $S(i)$ value, the raw fitness $R(i)$ of an individual solution i is calculated. $k-th$ nearest neighbour method is used to calculate the density information $D(i)$. After that, using Eq. (9) the quality $Q(i)$ of a solution i is calculated:

$$Q(i) = \frac{1}{R(i) + D(i)} \tag{9}$$

This $Q(i)$ value is used for the pheromone updating of PSACO and it uses the pheromone updating procedure as in AS (ant system) as defined in Eqs. (3) and (4). Finally, new archive is created by copying all the non-dominated solutions from the population P_t and the archive A_t. Some solutions are removed from the archive if non-dominated solutions are greater than the fixed size of the archive.

3.2. Crowding population-based ant colony optimization (CPACO)

Angus [7] introduced the crowding population based ant colony optimization (CPACO) algorithm for solving the multi-objective traveling salesman problem and it extends the population-based ant colony optimization algorithm (PACO). CPACO uses a crowding replacement scheme while the PACO uses the super/sub population scheme. Moreover, this scheme maintains a preset size of single population (S) and randomly generated solutions are used to initialize it. Every generation it creates a new population of solutions (Y) and to find its closest match a randomly selected subset S' of S is compared with each new solution. If the new solution is better than the existing solution it replaces the existing solution with the new solution. CPACO uses different heuristic matrices with a same pheromone trail for each objective. It initializes the pheromone matrix with some initial value τ_{init}. Thereafter, pheromone values of all solutions are updated in each iteration, according to their inverse of the rank as in the following equation:

$$\Delta\tau_{ij}^{s} = \frac{1}{S_{rank}} \tag{10}$$

An integer rank is assigned for all solutions in the population using the dominance ranking method. CPACO generates a correction factor (λ) of the heuristic component for each objective function of each ant. Therefore, this correction factor allows each ant to use different amount of heuristic matrices. Transition probability has been calculated using the following equation:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot \prod_{d=1}^{h} [\eta_{ij}^{d}]^{\lambda_d \beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha} \cdot \prod_{d=1}^{h} [\eta_{ij}^{d}]^{\lambda_d \beta}} \tag{11}$$

Summary attainment surface comparison and the C matrix (dominance ranking) performance measures are used to test the performance of the CPACO algorithm. This performance measures use to test whether the solutions are close to pareto front and also it obtains a diverse set of solutions. Moreover, the comparison has been done between the CPACO and the PACO algorithms. The CPACO algorithm obtains better solutions by covering the all areas of the pareto front as it locates and maintains a diverse set of solutions. Both algorithms use the distance sorting routine. CPACO algorithm maintains a smaller population than PACO and to assign ranks to solutions it applies a sorting method for non-dominated solutions. To assign ranks to the objectives PACO uses the average-rank-weight method. Therefore, CPACO obtained better results and its computational complexity is lower than PACO.

3.3. An efficient ant colony optimization algorithm for multi-objective flow shop scheduling problem (ACOMOFS)

Rabanimotlagh [8] introduced an efficient ant colony optimization algorithm for solving flow shop scheduling problem and it is named "ACOMOFS." It uses only one colony having one pheromone structure and several heuristic matrices to optimize two objectives, makespan and total flow time. In order to apply the state transition rule, a random number q is generated in the range $[0, 1]$. If the generated number $q \le q_0$, selects the next job according to the exploitation step given in the following equation:

$$j = \arg\max\{(\tau_j^t)^{\alpha}(\eta_j^t)^{\beta}\} \tag{12}$$

Exploration step is performed as follows when, $q > q_0$,

$$P_j = \frac{(\tau_j^t)^{\alpha}(\eta_j^t)^{\beta}}{\sum_{j' \in N}(\tau_{j'}^t)^{\alpha}(\eta_{j'}^t)^{\beta}} \quad \forall j \in N \tag{13}$$

where the relative importance of exploration versus exploitation is denoted by q_0. τ_j^t denotes the pheromone trail of the schedule when placing of job j at position t. η_j^t denotes the heuristic information of positioning job j at position t of the schedule. The relative importance of the heuristic preference and the pheromone trail are represented by β and α respectively. When selecting the next job j of the schedule, the heuristic preference is calculated in each construction step as in the following equation:

$$\eta_j^t = \frac{1}{Z^{\hat{s} \cup j} - Z^{\hat{s}}} \tag{14}$$

If the job j has not been added to the schedule yet, it is called the partial solution. Therefore, the total weighted objective function of the partial solution is represented by $Z^{\hat{s}}$. After adding the job j to this partial solution, the total weighted objective function is denoted by $Z^{\hat{s} \cup j}$. Before, selecting the next job at each construction step, the pheromone trail of the step is updated by applying the local updating rule as follows:

$$\tau_j^t = \mathrm{Min}\{\tau_j^t, (1-\rho')\tau_j^t + \rho'\tau_u\} \tag{15}$$

where τ_u calculated as follows:

$$\tau_u = \theta \frac{1}{\rho Z^{GBS}} \tag{16}$$

where Z^{GBS} denotes the objective function value of the global best solution and θ is a parameter. The global updating rule is performed after completing the tour by all ants in the colony. The iteration best solution or the globally best solution is used to apply the global updating of pheromone trails. Therefore as the first step, it evaporates the pheromone trails as in the following equation:

$$\tau_j^t = (1-\rho)\tau_j^t \tag{17}$$

Then, the global updating rule is applied as follows:

$$\tau_j^t = \tau_j^t + \rho \frac{1}{Z^{GBS}} \tag{18}$$

Moreover, the local search algorithm has been coupled with the main ACO algorithm for obtaining high quality solutions. After applying global updating, the iteration best solution is used to apply the local search algorithm and it uses two types of neighborhood structures. First one is "adjacent pair wise interchange method" and the second one is "the insertion of each job in each possible position of the sequence." The results of the proposed approach of ACOMOFS are compared with HAMC algorithms [9] and MOACSA algorithm [10]. This shows that ACOMOFS performs best for makespan and total flow time and multiple objectives than other algorithms. Also it archives lower computational time to obtain very good solutions when compared with other algorithms.

3.4. Ant colony optimization for multi-objective optimization problems (m-ACO)

Alaya et al. [11] proposed a generic ant colony optimization algorithm to solve multi-objective optimization problems which is called "m-ACO." It uses several ant colonies with several pheromone trails for solving the multi-objective knapsack problem. The m-ACO algorithm follows the MAX-MIN Ant System [12] scheme. Pheromone trails are connected with edges or vertices of the graph and a number of pheromone structures are included. Moreover, it assumes that every objective function uses a one heuristic trail value. Two bounds of pheromone trails τ_{min} and τ_{max} are defined to prevent premature convergence and these bounds should be in the interval $0<\tau_{min}<\tau_{max}$. At the beginning of the run, pheromone trails are initialized with upper bound of the pheromone trail, τ_{max}. At each iteration, an ant evaporates the pheromones by considering these bounds. The algorithm iterates until reach to the maximum number of iterations. At every iteration, each vertex of the graph selects the next vertex to move within the set of candidate list N according to the following probability as in the following equation:

$$P_S^c(v_i)=\frac{[\tau_S^c(v_i)]^{\alpha}[\eta_S^c(v_i)]^{\beta}}{\sum\limits_{v_i \in N}[\tau_S^c(v_i)]^{\alpha}[\eta_S^c(v_i)]^{\beta}} \qquad (19)$$

where the pheromone trail and the heuristic factor of the vertex v_i is represented by $\tau_S^c(v_i)$ and $\eta_S^c(v_i)$, respectively. The relative importance of the pheromone trail and the heuristic information is denoted by two parameters α and β. Pheromone trails are globally updated as follows:

$$\tau^i(c)\leftarrow(1-\rho)\times\tau^i(c)+\Delta\tau^i(c) \qquad (20)$$

There are four variants of this generic algorithm which differ from the number of colonies and pheromone trails.

3.4.1. Variant 1: mACO$_1$ (m+1, m)

This variant uses m number of pheromone trails and $(m+1)$ number of colonies, where number of objectives are represented by $m=|F|$. A single different objective function is used by each colony and it considers a single pheromone trail with single heuristic information. To optimize all objectives, it uses an extra ant colony. Pheromone may be lying on vertex v_i or on the edge between vertex v_i and the vertex of the partial solution S. Randomly chosen objectives have been assigned for the extra multi objective colony for optimization. Therefore, the pheromone trail considered for this colony is the pheromone trail of the randomly chosen colony. The sum of heuristic factors of all objectives is used as the heuristic factor of the extra multiobjective colony. The best solution of the current iteration is used to update the pheromone trails in each colony. In order to update the pheromone trail which corresponding to each objective, it uses best solution of each objective function.

3.4.2. Variant 2: mACO$_2$ (m+1, m)

This uses $(m+1)$ colonies and m number of pheromone structures. It has introduced an extra multiobjective colony which optimizes all objective functions. This variant is almost the same as the variant one. But the difference is, to build solutions of the extra multi objective colony it uses the pheromone factors of other colonies. Therefore, the sum of each pheromone factor of each colony is used to obtain the pheromone factor of the extra multi-objective colony.

3.4.3. Variant 3: mACO$_3$ (1, 1)

This variant of m-ACO uses a single colony with single pheromone structure. Pheromone structure is dependent on the considered application. The sum of heuristic factors of all objective functions is considered as the heuristic factor of the colony. Pheromone has been updated for all non-dominated solutions.

3.4.4. Variant 4: mACO$_4$ (1, m)

This variant of m-ACO uses a single colony and m number of pheromone factors. At each step an objective is chosen randomly for optimization. Therefore, the pheromone has been defined for this randomly chosen objective. The sum of heuristic factors of all objective functions is used as the heuristic factor of the colony. Pheromone is updated for m best solutions with regard to m objectives. Experimental results show that the mACO$_4$ (1, m) obtains better solutions for larger instances than the other variants.

3.5. Multiobjective optimization of time cost quality quantity using multicolony ant algorithm (MCAA)

Shrivastava et al. [13] proposed a new multi-colony ant algorithm (MCAA) for optimizing four objectives – time, cost, quality with quantity. The number of colonies are set to be equal to the number of objectives. Each colony uses different heuristic information and different pheromone structure. At the time period t, the state transition rule for moving an ant from node i to node j can be given as in the following equation:

$$j=\begin{cases}\arg\max\limits_{j \in allowed}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta} & \text{if } q\le q_0\\ J & \text{otherwise}\end{cases} \qquad (21)$$

At time t, the total pheromone trail deposits on path i,j is represented by τ_{ij}. Using the measurements of the objective functions the heuristic value of path i,j is denoted by η_{ij}. α and β are parameters which represents the relative importance of the pheromone trail and the heuristic information. J is a node and the probability distribution of the node is selected according to the following equation:

$$P_{ij}(t)=\begin{cases}\frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{l \in allowed}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}} & \text{if } j\in \text{allowed}\\ 0 & \text{otherwise}\end{cases} \qquad (22)$$

After all the ants in a colony complete their tour, the global updating is performed for non-dominated solutions as follows:

$$\tau_{ij}(t+1)\leftarrow\rho\tau_{ij}(t)+\Delta\tau_{ij} \qquad (23)$$

where ρ is the evaporation rate

$$\Delta\tau_{ij}=\begin{cases}\frac{Q}{nf(k)} & \text{if edge}(i,j)\text{ is transuessed by the }k\text{th ant}\\ 0 & \text{otherwise}\end{cases} \qquad (24)$$

where Q is a constant, n is the number of objectives and the value of the objective function in each iteration is represented by $f(k)$.

3.6. *Performance analysis of elitism in multi-objective ant colony optimization algorithms (AMPACOA)*

Bui et al. [14] studied the effect of elitism on multi-objective ant colony optimization algorithms and also, to control the effect of elitism a new adaptation strategy is proposed. The proposed approach is called as "AMPACOA" and it is based on the first ant colony optimization algorithm, ant system (AS). The way of pheromone updating is the only difference of this algorithm. This adaptation strategy gives an extra weighting to some solutions in the archive. The most recently entered solutions to the archive are assigned an extra weighting and the solutions are in the archive for a long period of time is assigned less weightings. The transition probability for moving from node i to node j is given as follows:

$$P_{ij}^{h} = \sum_{k=1}^{M} w_k P_{ij}^{k} \tag{25}$$

where p_{ij}^{k} is a probability which can be calculated using Eq. (2), M is the number of objectives and w_k is the weighting coefficient. Before laying an amount of pheromone at each location in each iteration, evaporation performs to decrease the pheromone trail values as in Eq. (17). When considering the adaptive elitism procedure, an age is assigned for each solution in the archive. At the time t, the coefficient δ is calculated for the solution h in the archive as follows:

$$\delta = \frac{m}{t - a_h + 1} \tag{26}$$

where a_h is the age of solution h in the archive and m is the number of ants. Using this aging strategy τ_{ij}^{h} can be calculated as follows:

$$\tau_{ij}^{h} = \tau_{ij}^{h} + \frac{\delta}{C^{p}} \tag{27}$$

3.7. *A multi-objective ant colony system (MACS) for vehicle routing problem with time windows*

Baran and Schaerer [15] proposed a multi-objective ant colony system (MACS) for solving the vehicle routing problem with time windows (VRPTW). The proposed approach for the VRPTW uses only one ant colony to optimize two objectives simultaneously. This incorporates the pareto optimal concept to obtain a set of pareto optimal solutions. All objectives use two heuristic information, η_{ij}^{0} and η_{ij}^{1} and a single pheromone trail τ in the colony. It follows the same transition rule of ACS to move from one node i to the next node j by each ant m. But it has applied for the multi-objective context as follows:

$$j = \begin{cases} \arg\max_{j \in N_i^k}\{\tau_{ij}[\eta_{ij}^{0}]^{\lambda\beta}[\eta_{ij}^{1}]^{(1-\lambda)\beta}\} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases} \tag{28}$$

where q is a random number in [0, 1] and the relative importance of the objectives is represented by β. For each ant k, λ is computed as $\lambda = k/m$, where m is the total number of ants and the next node J is selected as in the following equation:

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}[\eta_{ij}^{0}]^{\lambda\beta}[\eta_{ij}^{1}]^{(1-\lambda)\beta}}{\sum_{u \in N_i^k}\tau_{iu}[\eta_{iu}^{0}]^{\lambda\beta}[\eta_{iu}^{1}]^{(1-\lambda)\beta}} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

The local pheromone is updated as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0 \tag{30}$$

where τ_0 is the initial pheromone trail value of each objective function:

$$\tau_0 = \frac{1}{n f^{1}(s_h) f^{2}(s_h)} \tag{31}$$

where the initial number of nodes is denoted by n. After completing each iteration by each ant k, the pareto optimal set P is compared with the complete solution of the ant. Then each non-dominated solution is included and dominated ones are removed from the archive. Using the average values of the pareto optimal set, τ_0' is calculated at the end of each iteration. If $\tau_0' > \tau_0$, the pheromone trail is reinitialized with the new value τ_0'. Otherwise the pheromone trail of each solution in the current pareto optimal set is globally updated as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho/(f^{1}(s_h)f^{2}(s_h)) \tag{32}$$

Table 1 shows the taxonomy of the above MOACO algorithms.

4. Experimentation

To accomplish the experimental comparison, we considered only the seven MOACO algorithms that have been presented in the above section because they have been implemented in most recent years and their final aim is to produce a set of non-dominated solutions. However, the pareto strength ant colony optimization (PSACO) algorithm presented in Section 3.1, has been proposed and applied to automatic multi-objective hose routing in 3D space. The experimental results have shown that solutions of P-ACO are dominated by the corresponding solutions of PSACO [5]. Thus, in this experimentation PSACO algorithm was applied to the traveling salesman problem and the performance was compared with other MOACO algorithms.

Moreover, four variants have been considered in ant colony optimization for multi-objective optimization problems (m-ACO) [11]. It has been concluded that especially for large instances, the m-ACO$_4$ (1, m) variant is better than the other three variants. Thus, the m-ACO$_4$ (1, m) variant was selected for this experimentation. Finally, seven MOACO algorithms – ACOMOFS, AMPACOA, CPACO, mACO$_4$, MACS, MCAA and PSACO – were considered for this study and their performance levels were compared by applying them to the travelling salesman problem.

Table 1
A taxonomy for multi-objective ACO algorithms.

Algorithm	Number of colonies	Use of multiple pheromone matrices	Use of multiple heuristic matrices	Which solutions were used for global pheromone updating	Local pheromone updating	Which component was used for local pheromone updating
ACOMOFS	1	No	No	Iteration best solutions	Yes	Globally best solution
AMPACOA	1	Yes	Yes	Non-dominated solutions	No	–
CPACO	1	No	Yes	Non-dominated solutions	No	–
mACO$_4$	1	Yes	No	Iteration best solutions	No	–
MACS	1	No	Yes	Non-dominated solutions	Yes	Initial pheromone
MCAA	Multiple	Yes	No	Non-dominated solutions	No	–
PSACO	1	No	No	Non-dominated solutions	No	–

4.1. Adaptation of MOACO into multi-objective TSP

Only CPACO and AMPACOA algorithms have been proposed for the multi-objective travelling salesman problem while the other five MOACO algorithms have been proposed for different applications such as the flow shop scheduling problem, vehicle routing problem, etc. In order to adapt all the MOACO algorithms into TSP, several changes have to be done. Mostly, the heuristic information value should be changed, which represents *a priori* information of the problem instance. Also, the value of the objective function should be changed because different applications use different objective function values. Hence, in order to apply the five MOACO algorithms to the TSP problem, the following changes were performed.

4.1.1. Pareto strength ant colony optimization (PSACO)

The PSACO algorithm has been applied to the multi-objective problems for solving hose routing problem which is quite similar to TSP. But some changes should be performed in order to adapt the PSACO algorithm to TSP. Hence, the heuristic information will be set as in the following equation:

$$\eta_{ij} = \frac{k}{\sum_{k=1}^{K} d_{ij}^{k}} \tag{33}$$

where k is the number of objectives and the cost associated to the edge (i,j) of each objective is denoted by d_{ij}^{k}.

4.1.2. An efficient ant colony optimization algorithm for multi-objective flow shop scheduling problem (ACOMOFS)

ACOMOFS uses only one colony to optimize two objectives with a single heuristic matrix and only one pheromone value. Therefore, when applying the algorithm for multi-objective TSP the heuristic matrix will be changed as in Eq. (34), which is based on the heuristic matrix defined in Eq. (14).

$$\eta_{ij} = \frac{1}{\sum_{k=1}^{K} w_k d_{ij}^{k}} \tag{34}$$

where w_k is the weighting coefficient of each objective. Moreover, the pheromone updating will be done as Eq. (15) where τ_u calculated as follows:

$$\tau_u = \theta \frac{1}{\rho F} \tag{35}$$

where F is the total objective function value as defined in Eq. (36), θ and ρ are fixed parameters.

$$F = \sum_{k=1}^{K} w_k f_k \tag{36}$$

where f_k is the objective function value of objective k and w_k is the weighting coefficient of each objective. Equal weights are used for each of the objective functions. The weight can be computed as 0.5 for each objective when applied to the algorithm for the bi-criteria optimization problem.

4.1.3. Multi-objective optimization of time cost quality quantity using multi-colony ant algorithm (MCAA)

When applying this algorithm to the multi-objective TSP the heuristic information will be computed as in Eq. (34). Furthermore, a single pheromone coefficient and the same heuristic information will be set for each colony and the number of colonies will be set to the number of objectives.

4.1.4. Ant Colony Optimization for Multi-objective Optimization Problems (mACO$_4$)

When applying the mACO$_4$ algorithm in to multi objective TSP, the heuristic information which will be used by a colony will be set equal to the sum of all the heuristic information values related with all the objectives as follows:

$$\eta_{ij} = \frac{1}{\sum_{k=1}^{K} d_{ij}^{k}} \tag{37}$$

where k is the number of objectives and the cost associated to the edge (i,j) of each objective is denoted by d_{ij}^{k}.

4.1.5. A multiobjective ant colony system for vehicle routing problem with time windows (MACS)

The MACS algorithm has been proposed only for solving optimization problems with two objectives. A single colony uses two heuristic information values and only one pheromone matrix. MACS algorithm cannot be extended to solve multi-objective optimization problems. Therefore, some changes are made in order to adapt it to dealing with multi-objective TSP. Therefore, in the multi-objective context it will use different heuristic information values and only one pheromone matrix for all the objectives. Hence, an ant moves from node i to node j as in the following equation:

$$j = \left\{ \arg\max_{j \in N_i^k} \{\tau_{ij} \prod_{d=1}^{h} [\eta_{ij}^{d}]^{y_d \beta} \text{if } q \le q_0 J \text{otherwise} \right. . \tag{38}$$

where h denotes the number of objectives and the node J will be selected as in the following equation:

$$P_{ij}^{k} = \begin{cases} \frac{\tau_{ij} \prod_{d=1}^{h} [\eta_{ij}^{d}]^{y_d \beta}}{\sum_{u \in N_i^k} \tau_{iu} \prod_{d=1}^{h} [\eta_{iu}^{d}]^{y_d \beta}} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{39}$$

The sum of all heuristic component factors (y) set equal to one and it will be calculated as follows:

$$\sum_{d=1}^{h} y_d = 1 \tag{40}$$

4.2. Performance indicators

Different facts of the algorithms performance are reflected by performance indicators, such as their closeness to the true pareto optimal front and the distribution of solutions in the pareto optimal front. Many performance indicators can be categorized into two types: unary measures and binary measures. A quality value to a pareto set is represented by unary measures and binary measures compare two different pareto fronts obtained by two different algorithms.

4.2.1. Overall non-dominated vector generation (ONVG)

As given in the following equation, ONVG [16] measures the number of non-dominated solutions in each pareto front, denoted as $|y_{known}|$.

$$ONVG = |y_{known}| \tag{41}$$

where $|.|$ represents the cardinality. If the value of ONVG is larger, then the pareto front is better.

4.2.2. Overall true non-dominated vector generation (OTNVG)

OTNVG [16] calculates the number of solutions of y_{known} that are in true pareto optimal front, y_{true}. The higher the value of OTNVG, the better the solutions are obtained. This indicator is defined as given in the following equation:

$$OTNVG = |\{y | y \in y_{known} \wedge y \in y_{true}\}| \tag{42}$$

4.2.3. Overall true non-dominated vector generation ratio (OTNVGR)

OTNVGR [16] measures the ratio between the number of solutions in OTNVG to the number of solutions of true pareto optimal front. This can be expressed as a percentage as given in the following equation:

$$\text{OTNVGR} = \frac{\text{OTNVG}}{|y_{true}|} \times 100\% \tag{43}$$

A good solution should have a value of OTNVGR which is close to 100%.

4.2.4. Error ratio (ER)

This performance indicator [2] calculates the number of solutions of approximation set Q which are not members of the pareto optimal set P^* as given in the following equation:

$$\text{ER} = \frac{\sum_{i=1}^{|Q|} e_i}{|Q|} \tag{44}$$

when i is in P^* then $e_i = 0$ and otherwise $e_i = 1$. The lower values of the error ratio indicate that non-dominated solutions are better.

4.2.5. Hypervolume ratio (HVR) performance indicator

It can calculate the HVR value [2] as in the following equation:

$$\text{HVR} = \frac{\text{HV}(Q)}{\text{HV}(P)} \tag{45}$$

where HV(Q) and HV(P) represent the volume of the approximate pareto set and the volume of the true pareto set, respectively. If the approximate pareto set and the true one are equal, then the HVR value is equal to one. In other words, when the HVR value is close to one, the approximate pareto set is near the true pareto set. The better solutions have HVR values close to one.

4.2.6. Coverage performance indicator (C performance indicator)

This performance indicator [2] can be used to compare the two sets of non-dominated solutions A and B to find their relative spread of solutions. The C performance indicator $C(A, B)$ calculates the ratio between the number of solutions of B dominated by solutions in A to the number of solutions in B using the following equation:

$$C(A,B) = \frac{|b \in B | \exists a \in A : a \le b|}{|B|}. \tag{46}$$

Hence the value $C(A, B) = 1$ represents that all the solutions in B are dominated by or equal to solutions in A. On the other hand, $C(A, B) = 0$ means that none of the solutions in B are dominated by solutions in A. It is important to note that both $C(A, B)$ and $C(B, A)$ should be considered, because $C(A, B)$ is not necessarily equal to $1 - C(B, A)$.

It has been observed that the true pareto optimal front is used by most performance indicators. Therefore, it should be calculated an approximation to the true pareto optimal front which is called "pseudo-optimal pareto front", using the following 5 steps [17]:

- Each algorithm was run for 10 times to obtain non-dominated solutions; $y_1, y_2, \ldots, y_{10}$
- For each algorithm, the union of all runs were calculated as

$$y' = \bigcup_{i=1}^{10} y_i \tag{47}$$

- The pareto front of each algorithm was obtained from the union of all runs by removing the dominated solutions, as follows:

$$y_{ACOMOFS}, y_{AMPACOA}, y_{CPACO}, y_{mACO_4} y_{MACS}, y_{MCAA}, y_{PSACO} \tag{48}$$

- A set of solutions y' was obtained as

$$y' = y_{ACOMOFS} \cup y_{AMPACOA} \cup y_{CPACO} \cup y_{mACO_4} \cup y_{MACS} \cup y_{MCAA} \cup y_{PSACO} \tag{49}$$

- Dominated solutions were removed and finally the approximation to the true pareto front was obtained, called y_{apr} practically $y_{apr} \approx y_{true}$. In other words, y_{apr} is an excellent approximation to the true pareto front, y_{true}.

4.3. Problem instances and parameter setting

The traveling salesman problem (TSP) was selected to compare the MOACO algorithms presented in Section 3. Each algorithm in this study, two, three and four objectives were considered and ten independent runs were made to solve the TSP problem. Therefore, six multi-objective TSP benchmark instances were considered: kroab50, kroac50, kroabc50, kroabcd50, kroab100 and kroabc100 which involve 50 and 100 cities respectively [18]. For fair comparison, the same parameter setting was applied for all the MOACO algorithms used in this study. First, the most effective parameters which cause to the performance of the algorithms were identified and the best values were: $\alpha = 1, \beta = 2, \rho = 0.2, \rho' = 0.05$ and $q_0 = 0.98$. The initial pheromone value τ_0 of each algorithm was set as $5.5498 - E18$. Nevertheless, the $mACO_4$ algorithm was applied in different values for some parameters. In these $\beta = 5$ and $q_0 = 0.5$ gave the better non-dominated solutions. Furthermore, MOACO algorithms were compared by changing both the number of ants and the number of iterations. Hence, all the algorithms were applied to the three different models as given in Table 2.

The number of ants (n) and number of iterations (m) are ($n \times m$) ranging from 10×100, 20×100 and 20×50 were named as *model 1*, *model 2* and *model 3* respectively. Each MOACO algorithm of each model was run 10 times for a fixed number of iterations. All the algorithms were initiated in the same computer: Intel Core i3 CPU at 2.13 GHz, 1GB memory, Ubuntu 10.04 environment using CodeBlocks 10.05.

5. Analysis of results

Some of the performance indicators described in Section 4.2 were considered to analyze the performance of the MOACO algorithms. The ER performance indicator calculates the number of solutions of the approximation set which are not members of the pareto optimal set. In this experimentation, each algorithm returns different pareto fronts of different sizes. Hence, ER may calculate misleading results when comparing the small and large pareto fronts with each other. Also, the hypervolume ratio (HVR) performance indicator produces misleading results when a convenient reference point is not found. In addition, it cannot be easily applied to more than two objective problem instances, since calculations for finding the reference point and the volume is more complex. Therefore, ER and HVR performance indicators were not considered for analyzing the performance of MOACO algorithms in this study.

In contrast, OTNVGR measures better results than the ER performance indicator since it measures the number of solutions

Table 2
Three different models which MOACO algorithms were applied by changing both the number of ants and number of iterations.

Model	Number of ants (n)	Number of iterations (m)	Notation ($n \times m$)
Model 1	10	100	10×100
Model 2	20	100	20×100
Model 3	20	50	20×50

of the approximation set which are in the pareto optimal front. Also, it is easy to obtain a relative coverage comparison of two solution sets using the coverage performance indicator. Moreover, it is more useful when the true pareto optimal front is not known of the real world problems. Furthermore, OTNVGR and C performance indicators can be easily applied to analyze the performances of multiple objective problems. Therefore, ONVG, OTNVG, OTNVGR and C performance indicators were selected to analyze the performance of MOACO algorithms. ONVG and OTNVG performance indicators were used to represent only the non-dominated solutions in this study.

5.1. Analysis of the completion time

Figs. 1–9 represent the statistics of the completion time returns by each algorithm for six multi-objective TSP instances for three models separately. As discussed in Section 4.3, three different models were selected by changing the number of ants and number of iterations. All the completion times were represented using box-plots in milliseconds. If the completion time is lower, the algorithm is faster and vice versa.

When comparing the completion times of bi-objective TSP instances (see Figs. 1, 2, 4, 5, 7 and 8) of three models, it is noticeable that ACOMOFS and MACS algorithms are faster than other algorithms as they obtained minimum completion times. When we consider the completion times of four objective TSP instance (see Figs. 3, 6 and 9), the MACS algorithm becomes slower than ACOMOFS, CPACO, mACO$_4$ and PSACO algorithms as its completion time becomes longer. The reason is that, in each iteration, the MACS algorithm returns more non-dominated solutions than other MOACO algorithms in the four objective TSP instance (Kroabcd50).

When comparing the completion times of all TSP instances (see Figs. 1–9), AMPACOA and MCAA algorithms are slower than other algorithms as they return maximum completion times. The reason is that the AMPACOA algorithm takes more time to find a solution as it considers different transition probabilities for each objective as given in Eq. (25). On the other hand, the MCAA algorithm takes more time to find a solution as it considers one colony for one objective (see Table 1). Therefore, completion times of both algorithms are higher as they take more time to find a solution iteratively.

However, when comparing each TSP instance and the three models with each other, completion times obtained for model 1 and model 3 are less than model 2. This is because the problem size of model 2 (20 numbers of ants used in 100 iterations, hence problem size equal to 2000) is two times larger than the problem size of model 1 (10 numbers of ants used in 100 iterations, hence problem size equal to 1000) and model 3 (20 numbers of ants used in 50 iterations, hence problem size equal to 1000). When considering all TSP instances, the ACOMOFS algorithm is the fastest among all MOACO algorithms as its completion time is minimum. Moreover, it can be observed that the order of completion times of each MOACO algorithm does not change with each model of each TSP instance except for the MACS algorithm. Therefore, the order of completion time does not depend on the number of ants and number of iterations except for the MACS algorithm.

5.2. Visual representation of non-dominated solutions

All the non-dominated solutions generated by each algorithm by their 10 runs are joining together. Then, a single pareto front is obtained by removing dominated solutions [19]. Afterwards, the generated pareto fronts of TSP instances with bi-objectives: Kroab50, Kroac50 and Kroab100 for three different models are visually present using scatter-plot matrix method (see Figs. 10–18).

According to these figures, it is shown that the MACS algorithm returns good distribution over the pareto front while other MOACO algorithms return non-dominated solutions only in the central part of the pareto front. According to Fig. 10, the AMPACOA algorithm returns good non-dominated solutions for small TSP instances of size 50 (Kroab50) of model 1. Also, the mACO$_4$ algorithm returns poor non-dominated solutions for small size TSP instances of 50 cities (Kroab50 and Kroac50). However, AMPACOA and mACO$_4$ algorithms return very poor non-dominated solutions for large TSP instance of 100 cities (Kroab100). Therefore, it can be noted that AMPACOA and mACO$_4$ algorithms are outperformed by other MOACO algorithms. Moreover, it is observed that ACOMOFS, CPACO, MCAA and PSACO algorithms return better solutions than the MACS algorithm in the central part of the pareto front but they are not able to obtain solutions in the entire extent of the pareto front. However, it cannot be easily determined which algorithm is best among the ACOMOFS, CPACO, MCAA and PSACO algorithms as they have obtained similar pareto fronts.

5.3. Analysis of the overall true non-dominated vector generation ratio (OTNVGR)

Pseudo optimal pareto front y_{apr} can be obtained using the method explained in Section 4.2 which is a better approximation to the true pareto front. Tables 3 and 4 represent the number of solutions of $|y_{apr}|$ which were experimentally found for each TSP instance with respect to each model.

Tables 5–10 represent a comparison between the solutions with respect to the ONVG, OTNVG and OTNVGR for each model found with MOACO algorithms. The high value of OTNVGR indicates that the solution is better. According to these tables it can be observed that the MACS algorithm found many more

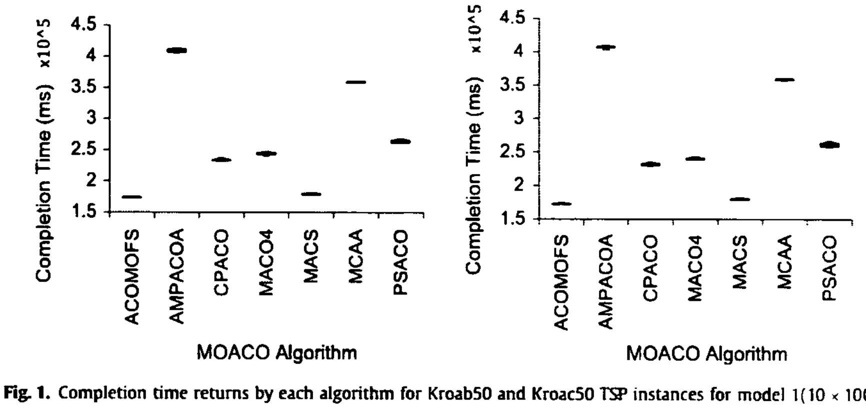

Fig. 1. Completion time returns by each algorithm for Kroab50 and Kroac50 TSP instances for model 1(10 × 100).

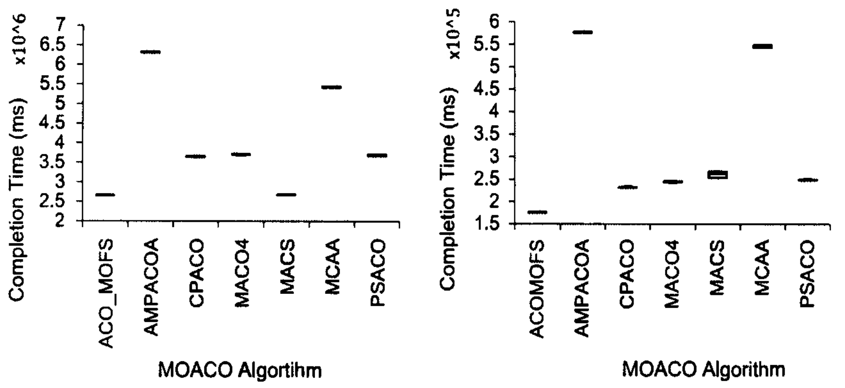

Fig. 2. Completion time returns by each algorithm for Kroab100 and Kroabc50 TSP instances for model 1(10 × 100).

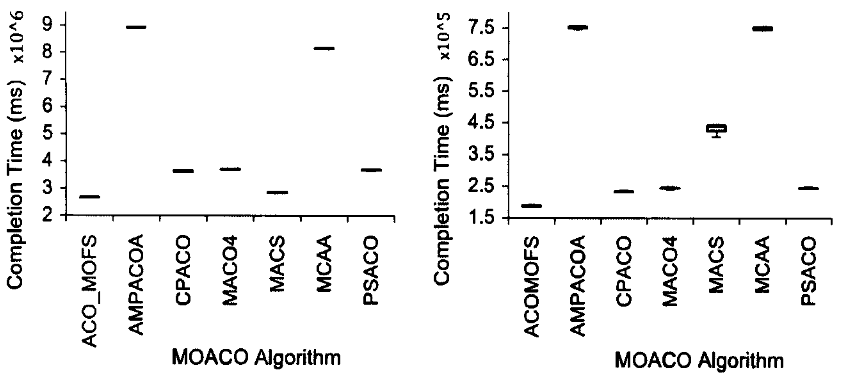

Fig. 3. Completion time returns by each algorithm for Kroabc100 and Kroabcd50 TSP instances for model 1(10 × 100).

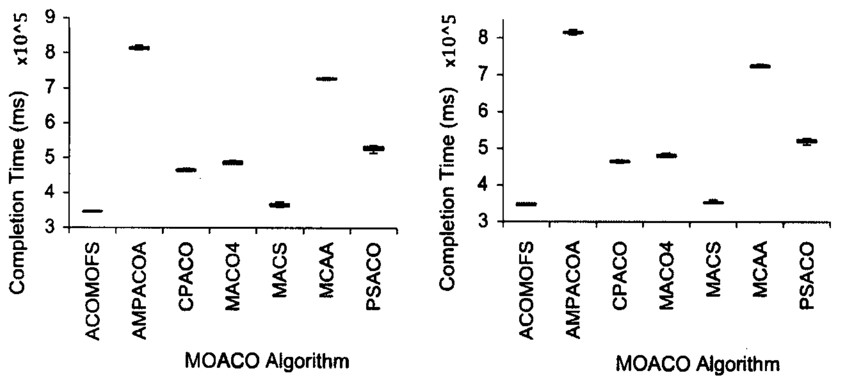

Fig. 4. Completion time returns by each algorithm for Kroab50 and Kroac50 TSP instances for model 2(20 × 100).

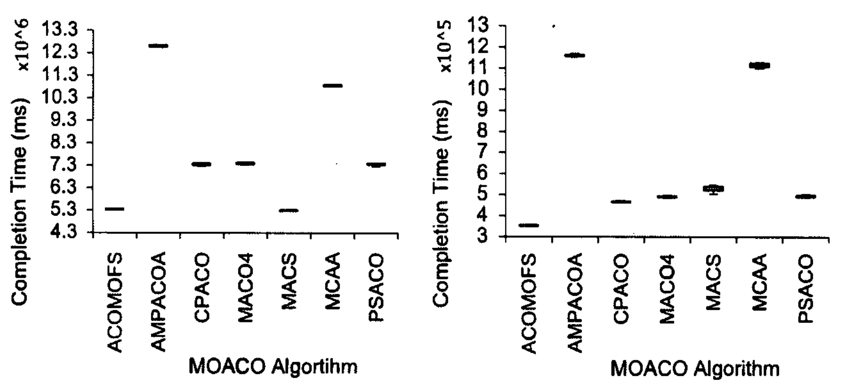

Fig. 5. Completion time returns by each algorithm for Kroab100 and Kroabc50 TSP instances for model 2(20 × 100).

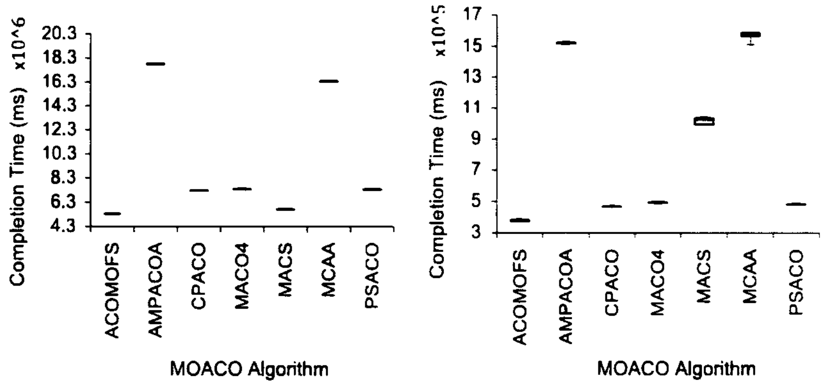

Fig. 6. Completion time returns by each algorithm for Kroabc100 and Kroabcd50 TSP instances for model 2(20 × 100).

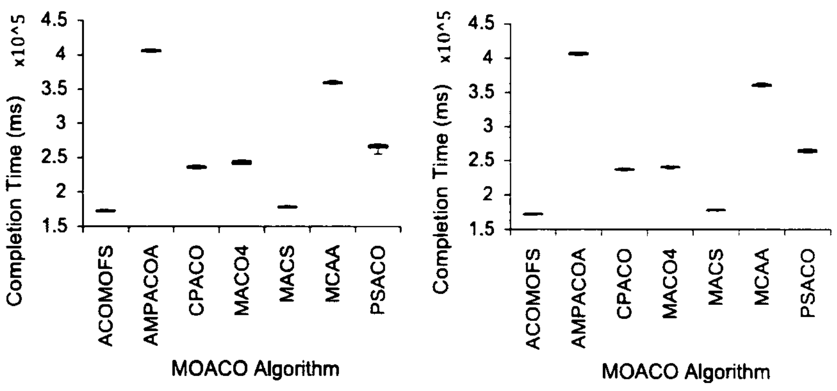

Fig. 7. Completion time returns by each algorithm for Kroab50 and Kroac50 TSP instances for model 3(20 × 50).

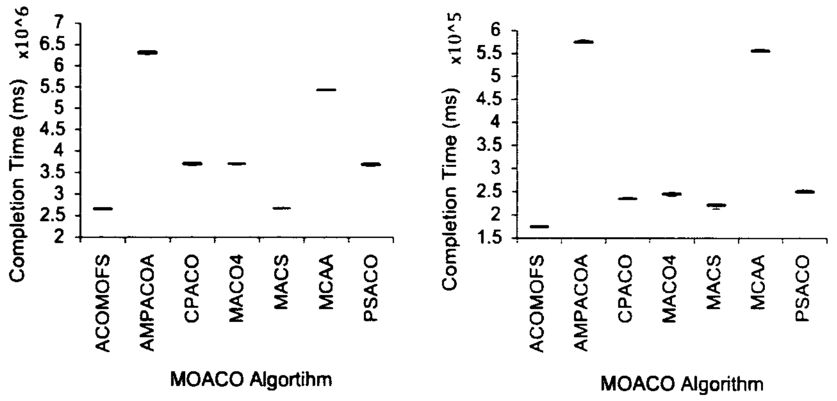

Fig. 8. Completion time returns by each algorithm for Kroab100 and Kroabc50 TSP instances for model 3(20 × 50).

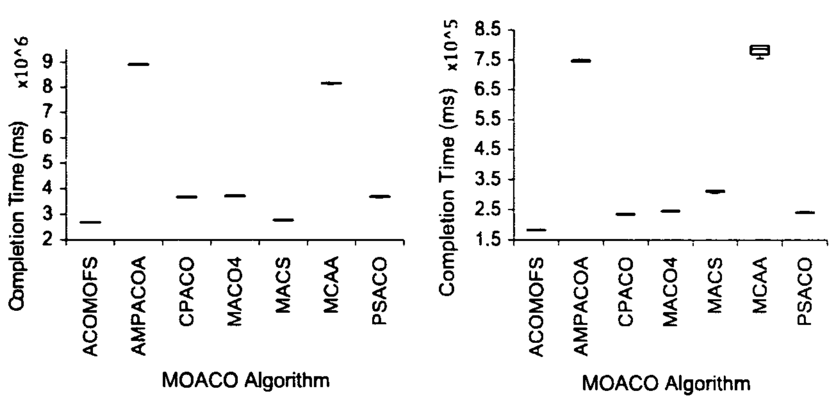

Fig. 9. Completion time returns by each algorithm for Kroabc100 and Kroabcd50 TSP instances for model 3(20 × 50).

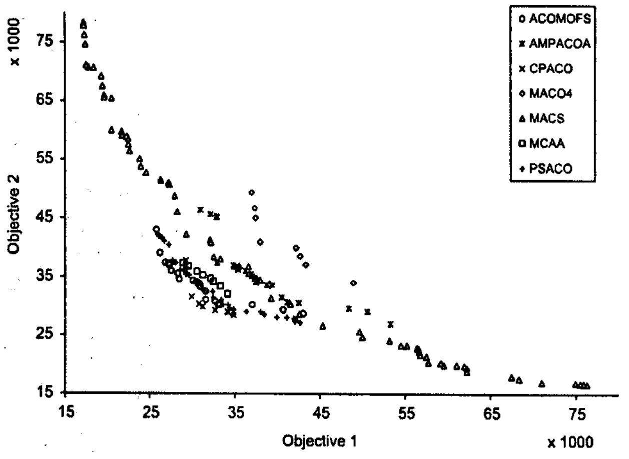

Fig. 10. Pareto fronts returns by each algorithm for Kroab50 TSP instance in model 1 (10 × 100).

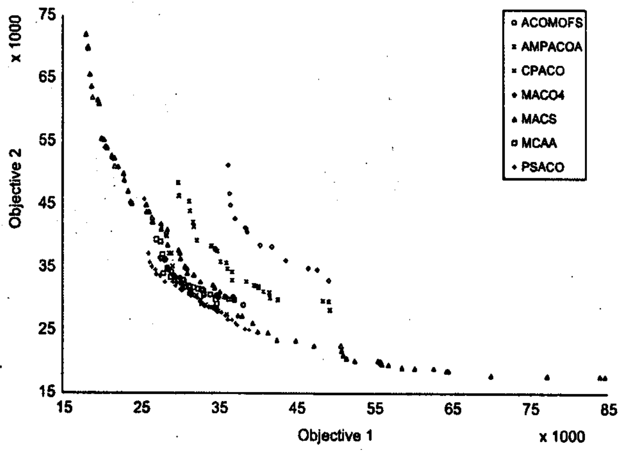

Fig. 11. Pareto fronts returns by each algorithm for Kroab50 TSP instance in model 2 (20 × 100).

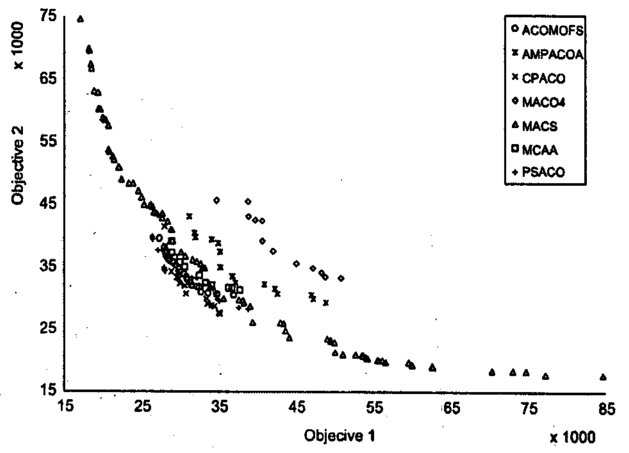

Fig. 12. Pareto fronts returns by each algorithm for Kroab50 TSP instance in model 3 (20 × 50).

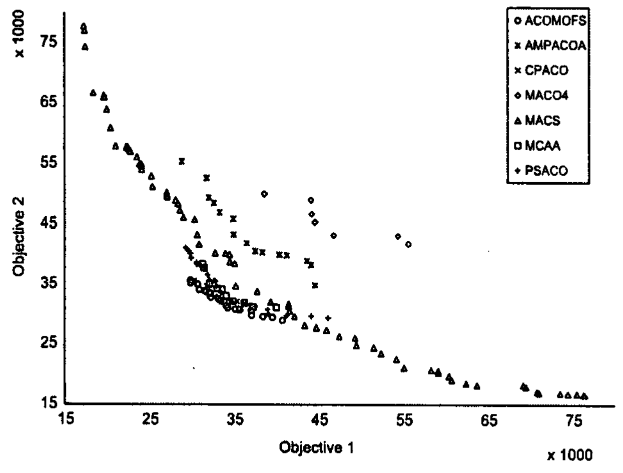

Fig. 13. Pareto fronts returns by each algorithm for Kroac50 TSP instance in model 1 (10 × 100).

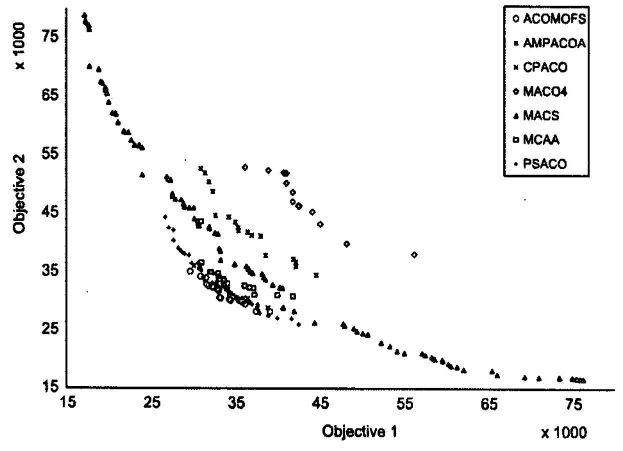

Fig. 14. Pareto fronts returns by each algorithm for Kroac50 TSP instance in model 2 (20 × 100).

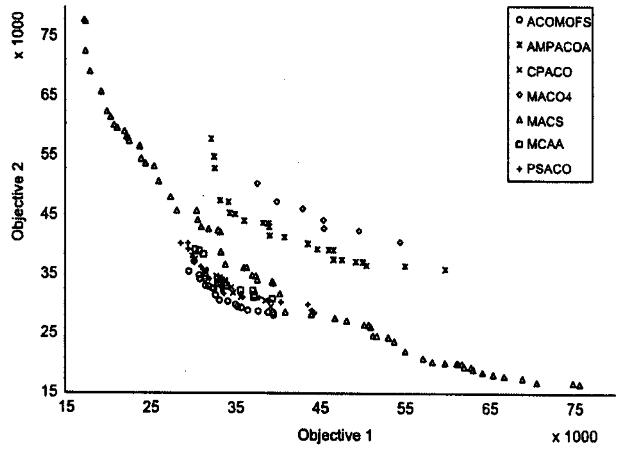

Fig. 15. Pareto fronts returns by each algorithm for Kroac50 TSP instance in model 3 (20 × 50).

solutions in OTNVGR than other MOACO algorithms in each model of each TSP instance. Therefore, the MACS algorithm is the best algorithm and also it is the best approximation to the true pareto front, y_{apr}. The reason is that the MACS algorithm obtains a set of non-dominated solutions which has been distributed all over the pareto front. Therefore, the most part of the pareto optimal front is covered by the MACS algorithm.

Secondly, ACOMOFS algorithm is better than all the other MOACO algorithms, but it could not obtain more solutions in OTNVGR than the MACS algorithm. This is because the ACOMOFS algorithm obtains better non-dominated solutions by covering only the central part of the pareto optimal front and was not able to obtain results to the extent of the pareto optimal front. Also, it is the best approximation to the central part of the pareto optimal

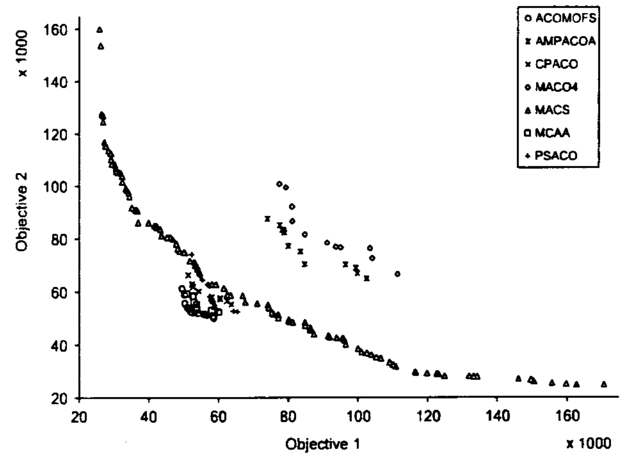

Fig. 16. Pareto fronts returns by each algorithm for Kroab100 TSP instance in model 1 (10 × 100).

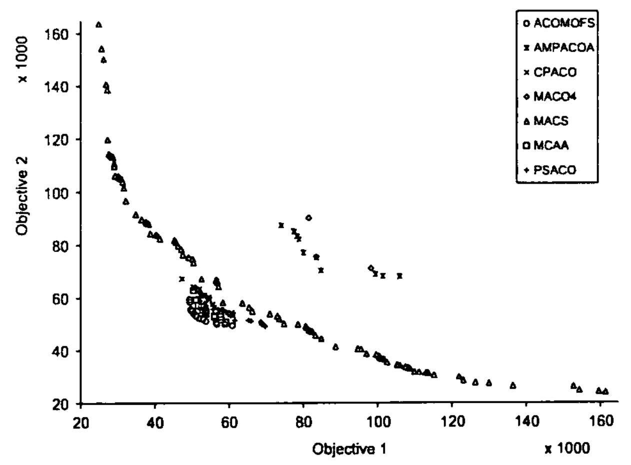

Fig. 17. Pareto fronts returns by each algorithm for Kroab100 TSP instance in model 2 (20 × 100).

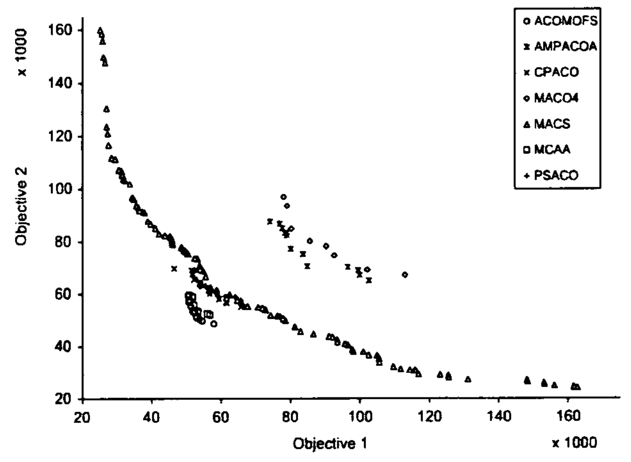

Fig. 18. Pareto fronts returns by each algorithm for Kroab100 TSP instance in model 3 (20 × 50).

front. Moreover, AMPACOA and $mACO_4$ algorithms are outperformed by all the other MOACO algorithms because they obtain no solutions in OTNVGR. Also, they obtained very poor pareto fronts which are not able to give better approximation to the pareto optimal front. Furthermore, Table 7 represents the large TSP instance of 100 cities: Kroab100. According to the Table, only MACS and ACOMOFS algorithms found better solutions in OTNVGR in each model. Therefore, other algorithms are outperformed by MACS and ACOMOFS algorithms as they did not find any solutions in OTNVGR.

When considering all the TSP instances (see Tables 5–10), ACOMOFS algorithm found better solutions in OTNVGR in model 1 and model 3. Also, PSACO algorithm found better solutions in OTNVGR in model 2 than the other two models. Moreover, CPACO and MACS algorithms found different values of OTNVGR in each model in each TSP instance. The reason is that it considered different numbers of ants and different numbers of iterations in each model. Therefore, it can be noted that when changing the number of ants and number of iterations, OTNVGR values of MOACO algorithms have been changed. Hence, the performances of MOACO algorithms changed with the number of ants and number of iterations used.

In addition, the PSACO algorithm outperformed AMPACOA, $mACO_4$, CPACO and MCAA algorithms in small TSP instances (Kroab50 and Kroac50) as it obtained more solutions in OTNVGR than those algorithms. However, CPACO and MCAA algorithms perform better in three and four objective TSP instances as their OTNVGR values are higher than in bi-objective TSP instances.

5.4. Analysis of the coverage performance indicator

The summary of the results obtained by all the algorithms is presented using a set of box-plots as shown in Figs. 19–21. In each box, six box-plots are represented from left to right: Kroab50, Kroac50, Kroab100, Kroabc50, Kroabc100 and Kroabcd50. Algorithms A and B refer to two algorithms in a row and column, respectively. This indicator measures the proportion of solutions of set B which are dominated by solutions of set A. Thus, $C(A,B)=1$ means that all solutions of B are weakly dominated by solutions of A. Also, $C(A,B)=0$ indicates that no solution of B is weakly dominated by A as described in Eq. (46) in Section 4.2. The middle line of each box represents the medians of $C(A,B)$. The bottom scale represents 0 and 1 at the top per each box.

Fig. 19 represents the box-plots of results obtained for model 1 and it should be noted that the ACOMOFS algorithm performs best among other MOACO algorithms since its C performance indicator values are almost zero most of the time. This is because pareto fronts obtained by the ACOMOFS algorithm dominate the pareto fronts obtained by other MOACO algorithms only in the central part of the pareto optimal front (see Figs. 13–18). Secondly, the MACS algorithm performs better than other MOACO algorithms as its C indicator values are close to zero. However, the ACOMOFS algorithm performs better than the MACS algorithm because the ACOMOFS algorithm dominated solutions of the MACS algorithm.

Also, AMPACOA and $mACO_4$ algorithms are outperformed by all the other algorithms as their solutions are dominated by other algorithms and most of the time their C indicator values are close to one. In addition, CPACO and PSACO algorithms perform similarly as they return similar C indicator values. Furthermore, the ACOMOFS algorithm performs better for bi-objective TSP instances in model 1 and 3 as their C indicator values are less than model 2 (see Figs. 19–21). In each model ACOMOFS, CPACO, MACS and MCAA algorithms return better solutions for three and four objective TSP instances as their C indicator values are minimum in comparison with bi objective TSP instances. Therefore, it can be noted that the performance of MOACO algorithms depends on the variation of objectives.

5.5. General analysis

By summarizing all the detailed analyses, some key global conclusions can be drawn as follows: In terms of completion

Table 3
Total number of non-dominated solutions of $|y_{apr}|$ for Kroab50, Kroac50 and Kroab100 TSP instances.

TSP instance	Kroab50			Kroac50			Kroab100		
Model	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
$\|y_{apr}\|$	91	136	110	101	143	81	101	91	93

Table 4
Total number of non-dominated solutions of $|y_{apr}|$ for Kroabc50, Kroabc100 and Kroabcd50 TSP instances.

TSP instance	Kroabc50			Kroabc100			Kroabcd50		
Model	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
$\|y_{apr}\|$	877	976	830	1187	1533	1213	2537	3774	2779

Table 5
Comparison of solutions with y_{apr} of Kroab50 TSP instance for each model.

MOACO algorithm	Kroab50								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	24	9	9.89	27	1	0.74	42	0	0.00
AMPACOA	20	0	0.00	36	0	0.00	15	0	0.00
CPACO	14	11	12.08	32	2	1.47	20	17	15.46
mACO$_4$	11	0	0.00	22	0	0.00	16	0	0.00
MACS	85	59	64.84	116	86	63.23	108	86	78.18
MCAA	10	0	0.00	31	0	0.00	15	0	0.00
PSACO	60	12	13.19	48	47	34.56	23	7	6.36

Table 6
Comparison of solutions with y_{apr} of Kroac50 TSP instance for each model.

MOACO algorithm	Kroac50								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	35	35	34.65	29	27	18.88	24	24	29.63
AMPACOA	15	0	0.00	17	0	0.00	25	0	0.00
CPACO	6	0	0.00	17	0	0.00	16	0	0.00
mACO$_4$	32	0	0.00	29	0	0.00	19	0	0.00
MACS	79	64	63.37	128	88	61.54	72	54	66.67
MCAA	20	0	0.00	42	0	0.00	32	0	0.00
PSACO	34	2	1.98	62	28	19.58	30	3	3.70

Table 7
Comparison of solutions with y_{apr} of Kroab100 TSP instance for each model.

MOACO algorithm	Kroab100								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	20	20	19.80	15	15	16.48	19	19	20.43
AMPACOA	16	0	0.00	21	0	0.00	16	0	0.00
CPACO	9	0	0.00	12	1	1.10	16	1	1.07
mACO$_4$	24	0	0.00	10	0	0.00	21	0	0.00
MACS	102	81	80.20	92	73	80.22	99	73	78.50
MCAA	12	0	0.00	15	0	0.00	9	0	0.00
PSACO	24	0	0.00	26	2	2.20	11	0	0.00

times, the results obtained show that ACOMOFS is the fastest algorithm as it obtains the minimum completion time while AMPACOA and MCAA are the slowest algorithms as they obtain poor completion times. In addition, the MACS algorithm achieves a good completion time except for the four objective TSP instance. Furthermore, it is observed that the order of completion time of

Table 8
Comparison of solutions with y_{apr} of Kroabc50 TSP instance for each model.

MOACO algorithm	Kroabc50								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	160	131	14.94	194	71	7.27	143	134	16.15
AMPACOA	92	0	0.00	155	0	0.00	98	0	0.00
CPACO	35	23	2.62	62	32	3.29	54	46	5.54
$mACO_4$	77	0	0.00	83	0	0.00	86	0	0.00
MACS	914	657	74.91	1119	794	81.35	846	627	75.54
MCAA	49	11	1.26	72	1	0.10	88	5	0.60
PSACO	110	55	6.27	105	78	7.99	57	18	2.17

Table 9
Comparison of solutions with y_{apr} of Kroabc100 TSP instance for each model.

MOACO algorithm	Kroabc100								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	123	123	10.36	208	208	13.57	156	156	12.86
AMPACOA	54	0	0.00	138	0	0.00	120	0	0.00
CPACO	33	30	2.53	77	61	3.98	73	52	4.29
$mACO_4$	73	0	0.00	136	0	0.00	120	0	0.00
MACS	1183	1034	87.11	1463	1262	82.32	1133	1002	82.60
MCAA	50	0	0.00	65	2	0.13	77	0	0.00
PSACO	63	0	0.00	66	0	0.00	64	3	0.25

Table 10
Comparison of solutions with y_{apr} of Kroabcd50 TSP instance for each model.

MOACO algorithm	Kroabcd50								
	Model 1(10 × 100)			Model 2(20 × 100)			Model 3(20 × 50)		
	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)	ONVG	OTNVG	OTNVGR (%)
ACOMOFS	535	535	21.01	672	655	17.35	492	488	17.56
AMPACOA	236	0	0.00	435	0	0.00	340	0	0.00
CPACO	38	35	1.38	104	100	2.65	83	81	2.91
$mACO_4$	228	0	0.00	255	0	0.00	254	0	0.00
MACS	2642	1933	76.19	3821	2956	78.33	2797	2169	78.05
MCAA	157	23	0.91	250	11	0.29	236	25	0.90
PSACO	127	13	0.51	130	52	1.38	148	16	0.58

MOACO algorithms are not dependent on the number of ants and the number of iterations used except for the MACS algorithm.

When representing bi-objective TSP instances graphically, it shows that only the MACS algorithm returns good distribution all over the pareto front while all other algorithms obtain non-dominated solutions only in the central part of the pareto front. Moreover, the AMPACOA and $mACO_4$ algorithms are outperformed by other algorithms. Further, ACOMOFS, CPACO, MCAA and PSACO algorithms obtain better solutions in the central part of the pareto front and they are competitive with each other.

When considering the OTNVGR performance indicator, it can be concluded that the MACS algorithm is the best performing algorithm over the other MOACO algorithms, while ACOMOFS is the second best algorithm. According to Table 1, only the MACS and ACOMOFS algorithms use the local pheromone updating. In addition, AMPACOA and $mACO_4$ algorithms are outperformed by other algorithms in all situations. As given in Table 1, AMPACOA and $mACO_4$ algorithms use one colony with multiple pheromone matrices. Moreover, the PSACO algorithm returns better solutions for the problem of large size of 2000 (20 × 100 model). Also, the ACOMOFS algorithm produced the best results for the problem of small size of 1000 (10 × 100 and 20 × 50 models). Therefore, it can be concluded that the performance of some MOACO algorithms depend slightly on the numbers of ants and numbers of iterations used.

According to the C performance indicator, the ACOMOFS and MACS algorithms are the best MOACO algorithms. But ACOMOFS is slightly better than the MACS algorithm as the solutions of the ACOMOFS algorithm dominates solutions of the MACS algorithm. Also, AMPACOA and $mACO_4$ algorithms are dominated by all the other MOACO algorithms. Nevertheless, ACOMOFS, CPACO, MACS and MCAA algorithms produce better results for three and four objective TSP instances than for bi-objective TSP instances. Therefore, it can be concluded that variation of objectives change the performance of MOACO algorithms.

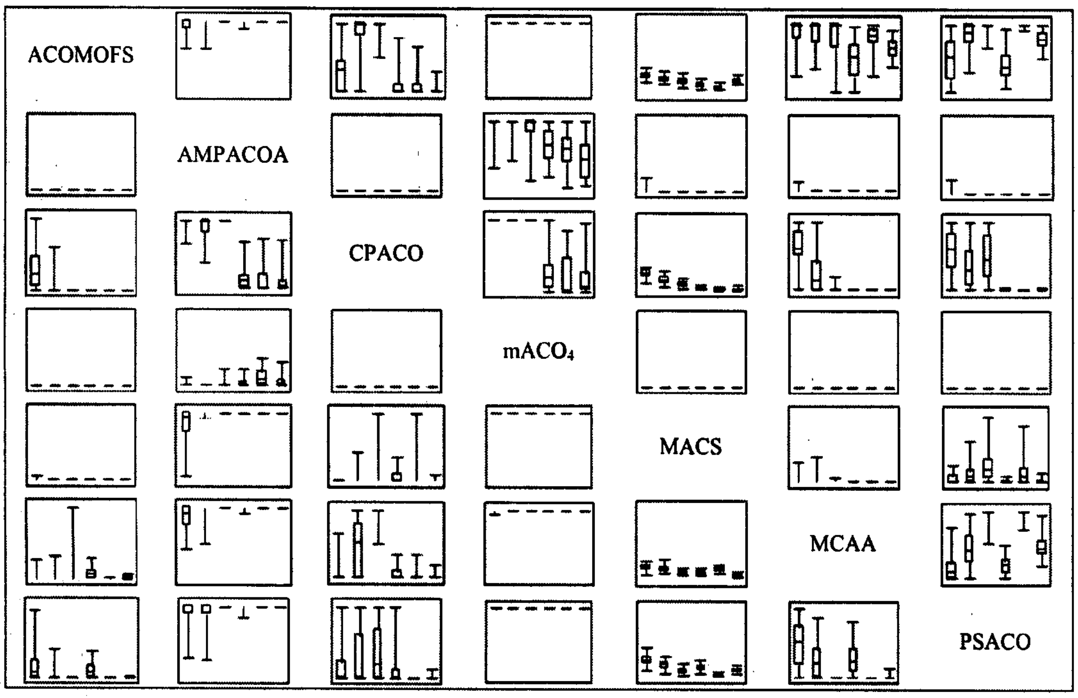

Fig. 19. Box-plots of the results obtained by the *C* performance indicator for MOACO algorithms of model 1.

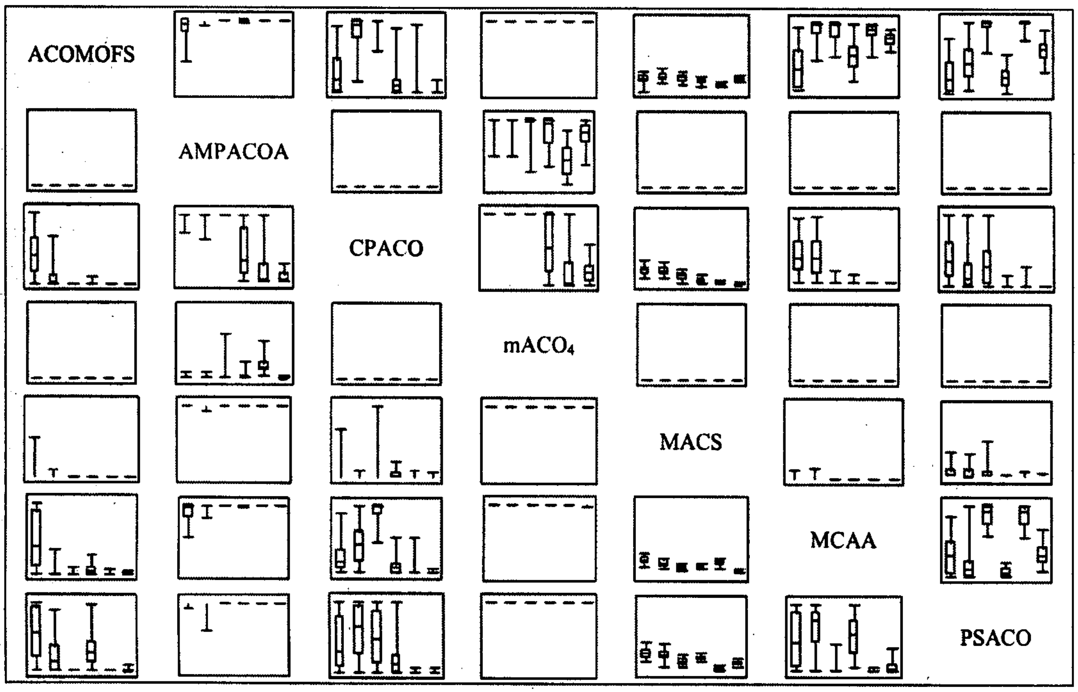

Fig. 20. Box-plots of the results obtained by the *C* performance indicator for MOACO algorithms of model 2.

6. Concluding remarks

This contribution has demonstrated that the comparison of the recent MOACO algorithms when applied to the six instances of the multi-objective TSP problem to optimize two, three and four objectives. The comparison was performed by changing the numbers of ants and numbers of iterations. MOACO algorithms have been compared graphically using the scatter-plot matrix method, as well as in terms of performance indicators (OTNVGR and C indicator) and completion times. According to the results obtained, some key conclusions can be drawn. In terms of completion time, the results showed that the ACOMOFS algorithm is the fastest algorithm in all situations. However, the order of completion times of MOACO algorithms are not dependent on the number of objectives, number of ants and number of iterations used except for the MACS algorithm. Graphically, it can be seen that AMPACOA and *m*ACO$_4$ algorithms are outperformed by all the other algorithms. The MACS algorithm returns non-dominated solutions throughout the pareto front and also it covers a greater proportion of the pseudo pareto optimal front than other algorithms.

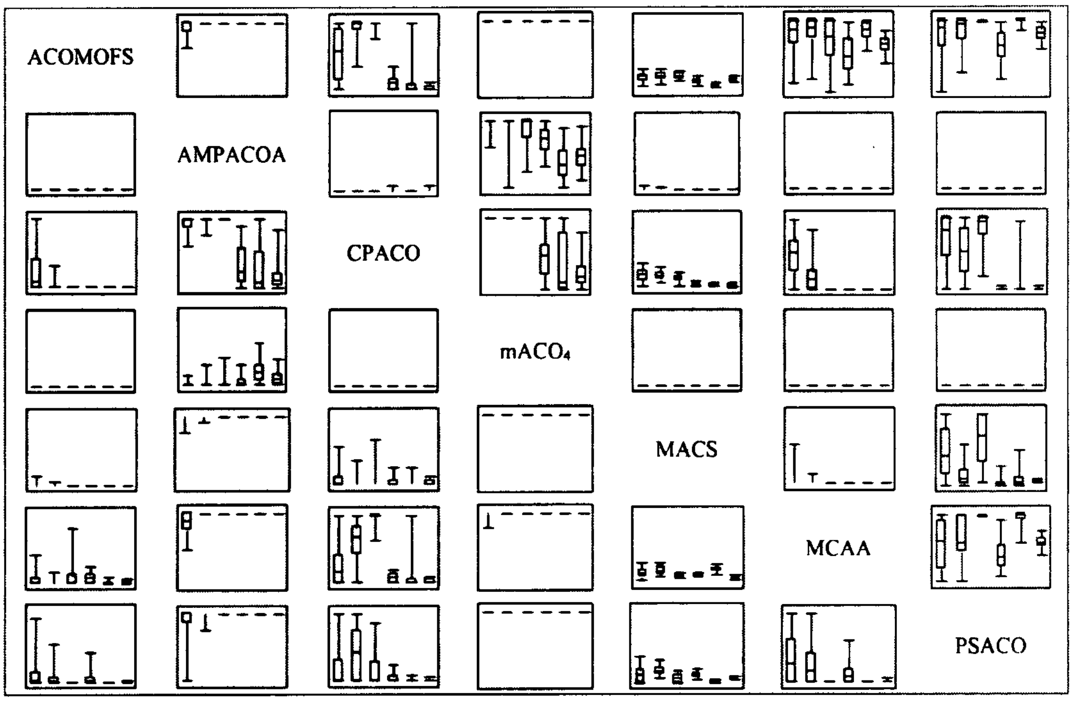

Fig. 21. Box-plots of the results obtained by the *C* performance indicator for MOACO algorithms of model 3.

According to the OTNVGR performance indicator, the MACS algorithm produces the best performance while the ACOMOFS algorithm is found to have the second best performance over other MOACO algorithms. But when considering the *C* performance indicator, the ACOMOFS algorithm dominates the solutions of the MACS algorithm as the ACOMOFS algorithm obtains better non-dominated solutions in the central part of the pareto front. Therefore, it can be concluded that MACS and ACOMOFS algorithms perform better in all the situations considered. Moreover, CPACO, MCAA and PSACO algorithms are competitive with each other. Furthermore, the performances of some MOACO algorithms such as ACOMOFS, CPACO, MCAA and PSACO algorithms depend slightly on the number of objectives, number of ants and number of iterations used.

For future development, an idea which arises from this study is to study the performance of these MOACO algorithms in other combinatorial optimization problems such as the quadratic assignment problem and the job shop scheduling problem.

Acknowledgment

The authors would like to thank Prof. Arjuna Parakrama, Department of English, University of Peradeniya, for his devoted support on language editing of this paper.

References

[1] M. Dorigo, T. Stutzle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[2] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
[3] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern.-Part B 26 (1) (1996) 29–41.
[4] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[5] G.I.F. Thantulage, Ant Colony Optimization Based Simulation of 3D Automatic Hose/Pipe Routing (Ph.D. Thesis), School of Engineering and Design, London, UK, 2009.
[6] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm, in: K. Giannakoglou, et al. (Eds.), EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, 2001, pp. 12–21.
[7] D. Angus, Crowding population-based ant colony optimization for the multi-objective travelling salesman problem, in: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-criteria Decision Making (MCDM 2007), Honolulu, Hawaii, USA, IEEE Press, 2007, pp. 333–340.
[8] A. Rabanimotlagh, An efficient ant colony optimization algorithm for multi-objective flow shop scheduling problem, World Acad. Sci. Eng. Technol. 75 (2011) 127–133.
[9] B. Yagmahan, M.M. Yenisey, A multi-objective ant colony system algorithm for flow shop scheduling problem, Expert Syst. Appl. 37 (2010) 1361–1368.
[10] J.C. Ho, Y.L. Chang, A new heuristic for the n-job, m-machine flow shop problem, Eur. J. Oper. Res. 52 (1991) 194–202.
[11] I. Alaya, C. Solnon, K. Ghedira, Ant colony optimization for multi-objective optimization problems, in: 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol. 1, IEEE Computer Society Press, Los Alamitos, CA, 2007, pp. 450–457.
[12] T. Stutzle, H. Hoos, MAX–MIN ant system, J. Future Gener. Comput. Syst. 16 (2000) 889–914.
[13] R. Shrivastava, S. Singh, G.C. Dubey, Multi-objective optimization of time cost quality quantity using multi colony ant algorithm, Int. J. Contemp. Math. Sci. 7 (2012) 773–784.
[14] L.T. Bui, J.M. Whitacre, H.A. Abbass, Performance analysis of elitism in multi-objective ant colony optimization algorithms, in: 2008 Congress on Evolutionary Computation (CEC2008), Hong Kong, IEEE Service Center, 2008, pp. 1633–1640.
[15] B. Baran, M. Schaerer, A multi objective ant colony system for vehicle routing problem with time windows, in: Proceedings of the 21st IASTED International Conference on Applied Informatics, Insbruck, Austria, February 10–13, 2003, pp. 97–102.
[16] J. Gottlieb, G.R. Raidl, Evolutionary computation in combinatorial optimization, in: 6th European Conference, EvoCOP 2006, Budapest, Hungary, April 10–12, 2006.
[17] D. Pinto, B. Baran, Solving multi-objective multicast routing problem with a new ant colony optimization approach, in: Second IFIP/ACM Latin American Conference on Networking (LANC05), Cali, Colombia, 2005, pp. 11–19.
[18] G. Reinelt, TSPLIB—a traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384.
[19] C. Garcia-Martinez, O. Cordon, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, Eur. J. Oper. Res. 180 (1) (2007) 116–148.