International Journal of Computer Applications (0975 - 8887)
Volume 132 - No.14, December 2015

A Performance Study for the Multi-objective Ant Colony Optimization Algorithms on the Job Shop Scheduling Problem

I.D.I.D. Ariyasingha
Department of Mathematics and Computer Science,
Faculty of Natural Sciences,
Open University of Sri Lanka, Sri Lanka

T.G.I. Fernando
Department of Computer Science,
Faculty of Applied Science,
University of Sri Jayewardenepura, Sri Lanka

ABSTRACT

Most of the research on job shop scheduling problem are concerned with minimization of a single objective. However, the real world applications of job shop scheduling problems are involved in optimizing multiple objectives. Therefore, in recent years ant colony optimization algorithms have been proposed to solve job shop scheduling problems with multiple objectives. In this paper, some recent multi-objective ant colony optimization algorithms are reviewed and are applied to the job shop scheduling problem by considering two, three and four objectives. Also in this study, four criteria: makespan, mean flow time, mean tardiness and mean machine idle time are considered for simultaneous optimization. Two types of models are used by changing the number of ants in a colony and each multi-objective ant colony optimization algorithm is applied to sixteen benchmark problem instances of up to 20 jobs $\times$ 5 machines, for evaluating the performances of these algorithms. A detailed analysis is performed using the performance indicators, and the experimental results have shown that the performance of some multi-objective ant colony optimization algorithms depend on the number of objectives and the number of ants.

Keywords

ant colony optimization, job shop scheduling problem, multi-objective problem, non-dominated solution, pareto optimal front, performance indicator

1. INTRODUCTION

The job shop scheduling problem (JSSP) is NP-hard [10] and also one of the most complex combinatorial optimization problems. It is defined as the process of assigning a set of tasks to resources over a period of time [13]. Many methods have been developed to solve the single objective optimization problem, in order to optimize the time required to complete all jobs, i.e. considering a single objective to minimize the makespan criterion. However, most real world applications of scheduling require the simultaneous optimization of multiple objectives. During the past few years most researches have proposed to solve job shop scheduling problems which consist of multiple objectives. Ant colony optimization (ACO) is a meta-heuristic which can be used to solve the combinatorial optimization problems such as the traveling salesman problem, the job shop scheduling problem and the quadratic assignment problem, etc [8]. In the recent years, several papers reviewed the MOACO algorithms and conducted different kinds of experimentation. Garcia-Martinez et al. [9] reviewed and experimentally compared the MOACO algorithms when applied to the bi-objective traveling salesman problem. Angus and Woodward [3] reviewed the MOACO algorithms but did not carry out an experimental study. Also, Lopez-Ibanez and Stutzle [12] reviewed and provided an experimental study to understand the effects of various algorithmic components for the MOACO algorithms when applied on the bi-objective traveling salesman problem. Further, our previous study [4] reviewed and performed experimentation in order to analyze the performance of MOACO algorithms when applied to the traveling salesman problem. In fact, all these papers have focused on solving the traveling salesman problem. Hence, in this study a different problem domain is considered for solving to understand the behaviour of MOACO algorithms. Therefore, the job shop scheduling problem is selected as the problem domain in this study.

The aim of this paper is to review some recent multi-objective ant colony optimization (MOACO) algorithms and evaluate their performance by changing the number of objectives and ants. Further, MOACO algorithms are applied on several benchmark problem instances of the job shop scheduling problem by considering two, three and four objectives and a detail analysis has been performed using performance indicators. This paper is organized as follows: Section 2 reviews some preliminaries about the multi-objective optimization problem, the job shop scheduling problem, ant colony optimization algorithm and MOACO algorithms. The experimentation with parameter values, problem instances and performance indicators are presented in Section 3. Section 4 presents the analysis of the experimental results. Finally, Section 5 provides the conclusion.

2. PRELIMINARIES

2.1 Multi-objective optimization problem

A general multi-objective optimization problem (MOOP) [7] can be formulated as follows, which consists of more than one objective to be minimized or maximized simultaneously. MOOP includes a

set of m objectives, n decision variables with k restrictions.

$$\left.\begin{array}{l} y = f(x) = [f_1(x), f_2(x), ..., f_m(x)], \\ e(x) = [e_1(x), e_2(x), ..., e_k(x)] \geq 0, \\ x = (x_1, x_2, ..., x_n) \in X \textit{ is the decision vector}, \\ y = (y_1, y_2, ..., y_m) \in Y \textit{ is the objective vector}. \end{array}\right\} \quad (1)$$

where X denotes the decision space of n decision variables, while the objective space with m set of objective functions is denoted by Y. Only one optimal solution is found by the single objective optimization problem. In general, multi-objective optimization problems find more than one optimal solutions.

2.2 Job shop scheduling problem

The job shop scheduling problem (JSSP) [15] is one of the combinatorial optimization problems and it is considered as a most difficult problem in the literature. The general JSSP can be formulated as n x m, which consists of a set of n jobs to be processed on a set of m machines. The set of n jobs can be presented as $J = J_1, J_2, ..., J_n$ and the set of m machines can be presented as $M = M_1, M_2, ..., M_m$. An operation O_{ij} is defined as the processing of a job J_i on a machine M_j. Hence, the set of operations O can be represented as $O = \{O_{ij} | i \in [1, n], j \in [1.m]\}$, where n is the number of jobs and m is the number of machines. Each operation $O_{ij} \in O$ processes on machine j in an uninterrupted time period, is called processing time $p_{ij} (p_{ij} > 0)$ and it has defined in advance for each job. Two constraints – the operation precedence constraint and machine processing constraint – should be satisfied when an job J_i is processed on m machines.

In this study, the objective of the JSSP is to find a schedule which minimizes makespan, mean tardiness, mean flow time and mean machine idle time, simultaneously as given in Eq. (2).

$$min_{x \in X} f(x) = \{f_1(x), f_2(x), f_3(x), f_4(x)\} \quad (2)$$

where X is a set of feasible schedules, x is a schedule and $f_1(x), f_2(x), f_3(x), f_4(x)$ are the objective functions of the four criteria: makespan, mean tardiness, mean flow time and mean machine idle time. The following notations are used in the multi-objective job shop scheduling problem.

Completion time (C_i): Completion time of job i
makespan (C_{max}): Total time taken to complete all the jobs
Flow time (F_i): Total amount of time which the job i spends in the system
Due date (d_i) : Due date of job i
Tardiness (T_i): The positive difference between the completion time C_i and due date d_i of job i
Machine idle time (I_j): Idle time of the machine j

Hence, the four objectives which are used in scheduling in this study are formulated as follows.

makespan: $C_{max} = max(C_i)$

Mean tardiness: $\overline{T} = \frac{1}{n} \sum_{i=1}^{n} (T_i)$

Mean flow time: $\overline{F} = \frac{1}{n} \sum_{i=1}^{n} (F_i)$

Mean machine idle time: $\overline{I} = \frac{1}{n} \sum_{j=1}^{m} (I_j)$

2.3 Ant colony optimization

The more general social insect societies called ant colonies, present a highly structured distributed system. Ants in the colony use indirect communication methodology named *stigmergy* for coordinating their activities, because of ants species are totally blind. Foraging ant in the colony deposits a chemical called *pheromone trail* on the ground when travels back to the nest, which depends on the quality and the quantity of food source it carries. Hence, the other foraging ants in the colony will follow the same path in high probability. Pheromones deposited on the ground are evaporated over time. This behavior of real ants allows them to find the shortest path between their nest and food source, eventually. The self-organization behavior of real ant colonies is based on the agents called artificial ants.
In early 1990s Dorigo et al. introduced ant colony optimization (ACO) algorithms [8], which can be applied to solve combinatorial optimization problems. Ants in the artificial colony traverse from one node to another on a graph to find a solution for a problem and they use data structures in memory, which consists of the information of their previous actions. The probability of choosing a node as the next node is depended on the *artificial pheromone trail* laid on the path and the *heuristic information*, which measure the quality of the path. After completing a tour by an ant, the pheromone trail is evaporated and applies a new pheromone trail on the path. If the completed path by ant is a good solution, then the pheromone trail on that path will be high and vice-versa.

2.4 Multi-objective ant colony optimization algorithms (MOACO)

Since most of the real world applications are concerned about the multi-objective optimization problems, several algorithms have been proposed to solve multi-objective combinatorial optimization problems. The taxonomy of the previous review papers including the details of this study, is presented in Table 1. Garcia-Martinez et al. [9] reviewed some of the multi-objective ant colony optimization algorithms until 2007 and proposed a taxonomy. Performances of these algorithms have been analyzed on some instances of bi-objective traveling salesman problem. It has been concluded that multiple ant colony system (MACS) [5] algorithm performs better and achieves distributed pareto-optimal front.
Further, Angus and Woodward [3] reviewed some MOACO algorithms and presented a detail classification based on the different features of MOACO algorithms. However, it has not been performed any experimental analysis for the MOACO algorithms. Lopez-Ibanez and Stutzle [12] presented a new formulation for classifying the MOACO algorithms, which is based on the other algorithmic components which are not presented, previously. Also, it has been performed an experimental analysis to understand, how these algorithmic components effect the shape and the quality of pareto front of MOACO algorithms. Therefore, four MOACO algorithms including MACS [5] and $mACO_2$ [1], are applied to bi-objective traveling salesman problem.
Seven recent MOACO algorithms have been considered in our previous study [4] and it has been analyzed the performances of MOACO algorithms when applied to the multi-objective traveling salesman problem. It has shown that AMPACOA algorithm [6] obtains extremely poor results and poor computational times, hence it is not considered in this study. Therefore, other six recent MOACO algorithms are considered in this study, which produces a set of non-dominated solutions. They are – efficient ant colony optimization algorithm for multi-objective flow shop scheduling prob-

Table 1. A taxonomy for review papers on multi-objective ACO algorithms

Title of the Paper	Problem Domain	Combinatorial Optimization Problem Considered	Number of Objectives Considered
A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP [9]	Reviewed and presented a classification for MOACO algorithms according to the usage of number of heuristic matrices and number of pheromone trails. Performed a emperimental analyze for these MOACO algorithms when applied to bi-objective traveling salesman problem.	Traveling salesman problem	Two
Multiple objective ant colony optimization [3]	Reviewed and presented a detail classification of existing MOACO algorithms based on algorithmic components. However, it is not presented an experimental analysis.	None	None
An experimental analysis of design choices of multi-objective ant colony optimization algorithms [12]	Reviewed and presented a taxonomy for MOACO algorithms in terms of the other algorithmic components. Experimental work has been performed to understand how algorithmic components affect on the quality and the shape of MOACO algorithms	Traveling salesman problem	Two
Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem [4]	Review and the detail analysis has been performed to compare the performances of MOACO algorithms when applied on multi-objective traveling salesman problem by changing the number of ants, objectives and iterations.	Traveling salesman problem	Four
A performance study for the multi-objective ant colony optimization algorithms on the job shop scheduling problem	Study the performances of MOACO algorithms by changing the number of objectives and ants when applied to the job shop scheduling problem	Job shop scheduling problem	Four

Table 3. Objective functions considered in each JSSP instance

Objective function	Criterion
Two objectives	makespan, mean tardiness
Three objectives	makespan, mean tardiness, mean flow time
Four objectives	makespan, mean tardiness, mean flow time, mean machine idle time

Table 4. Parameter values considered

MOACO algorithm	τ_0	α	β	ρ	ρ'	q_0
ACOMOFS, $MACO_4$, MCAA	0.000125	0.1	0.5	0.2	0.05	0.98
CPACO, MACS, PSACO	5.5498E-18	1	2	0.2	0.05	0.98

lem (ACOMOFS) [14], crowding population-based ant colony optimization (CPACO) [2], ant colony optimization for multi-objective optimization problems (m-ACO_4) [1], a multi-objective ant colony system (MACS) for vehicle routing problem with time windows [5], multi objective optimization of time cost quality quantity using multi-colony ant algorithm (MCAA) [17], pareto strength ant colony optimization (PSACO) [18].
Taxonomy of all the multi-objective algorithms considered in this study can be represented in Table 2.

3. EXPERIMENTATION

The goal of this experimentation is to analysis the performances of MOACO algorithms, which are given in the previous Section 2, by changing the number of ants and number of objectives. In most of the previous studies [19, 20], the number of ants was set equal to the number of operations, when applying ant colony optimization algorithms for the job shop scheduling problem. i.e. number of ants $N = n \times m$, where n is the number of jobs and m is the number of machines. Therefore, number of ants are changed in this study to see, how they effect on the performances of MOACO algorithms.
Our previous study [4] has been concluded that ACOMOFS and MACS algorithms performed best and CPACO, $mACO_4$, MCAA and PSACO algorithms achieved good results for some instances in the traveling salesman problem. Therefore, these six recent MOACO algorithms: ACOMOFS, CPACO, $mACO_4$, MACS, MCAA and PSACO have been considered for this study. All these MOACO algorithms have been introduced for different applications. In order to apply them on job shop scheduling problem, some changes should be performed as given in the following section.

3.1 Parameter values and problem instances

The job shop scheduling problem is selected to analyze the performances of the six recent MOACO algorithms, which given in Section 2. Each MOACO algorithm considered in this study is applied to solve sixteen JSSP instances (la01-la16) from the literature [11] by considering two objectives, three objectives and four objectives. Table 3 shows each objective function with their criterion which optimized simultaneously, while satisfying the constraints of JSSP. MOACO algorithms in this study use two types of parameter settings for fair comparison, which is based on their performances. First, the most effective parameters are identified for each MOACO algorithm and they can be presented as in Table 4. Also, all algorithms are considered 100 iterations and run in 10 times for fair comparison.
Further, the study considers two types of models by changing the number of ants, in which it uses 20 ants in model 1 and 50 ants in model 2. These two models consider same parameter settings as given in Table 4. All algorithms are implemented in the same computer using CodeBlocks 13.12 under Ubuntu 14.04 environment running on Intel Core i3 CPU at 2.40GHz, with 4GB memory.

Table 2. 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	One	No	No	Iteration best solutions	Yes	Globally best solution
CPACO	One	No	Yes	Non-dominated solutions	No	-
$mACO_4$	One	Yes	No	Iteration best solutions	No	-
MACS	One	No	Yes	Non-dominated solutions	Yes	Initial pheromone
MCAA	Multiple	Yes	No	Non-dominated solutions	No	-
PSACO	One	No	No	Non-dominated solutions	No	-

Table 5. Results of each JSSP problem instance for two objectives, obtained with model 1 (20 ants are used)

Problem	Size	ACOMOFS	CPACO	$MACO_4$	MACS	MCAA	PSACO
LA01	10x5	0.00	0.30	0.37	0.26	1.27	1.10
LA02	10x5	1.67	5.12	3.32	3.78	4.49	4.96
LA03	10x5	8.52	9.77	0.00	9.31	7.62	1.85
LA04	10x5	0.97	1.18	0.00	1.27	4.15	0.03
LA05	10x5	0.00	0.40	1.48	0.72	1.77	1.56
ARPD		2.23	3.35	1.03	3.07	3.86	1.90
Average CPU time (seconds)		128	96	98	103	146	121
LA06	15x5	0.00	0.83	4.14	2.00	3.20	2.69
LA07	15x5	1.85	6.83	3.54	5.76	7.06	3.41
LA08	15x5	2.08	4.84	5.17	1.02	4.38	0.29
LA09	15x5	15.83	18.06	21.19	15.83	3.02	13.86
LA10	15x5	0.00	4.51	7.17	3.04	8.56	4.76
ARPD		3.95	7.01	8.24	5.53	5.24	5.00
Average CPU time (seconds)		417	334	329	335	452	365
LA11	20x5	0.16	2.48	6.91	1.12	0.00	3.06
LA12	20x5	0.00	4.09	4.96	2.00	0.51	2.78
LA13	20x5	0.00	3.81	7.30	1.36	2.42	4.23
LA14	20x5	2.35	7.23	9.43	1.11	0.72	6.68
LA15	20x5	1.81	4.86	9.72	0.00	5.04	8.08
LA16	20x5	3.15	1.50	0.00	1.72	2.25	2.41
ARPD		1.25	3.99	6.39	1.22	1.82	4.54
Average CPU time (seconds)		1004	876	808	796	1160	827
Mean of ARPDs		2.40	4.74	5.29	3.14	3.53	3.86
# of optimum solutions		8	0	3	1	3	1

3.2 Performance measures

Relative percentage deviation (RPD) and average relative percentage deviation (ARPD) [16] are used as the performance measures to evaluate the performances of MOACO algorithms, considered in this study. Relative percentage deviation (RPD) is calculated as follows for multi-objective optimization problems.

$$RPD = \left[\sum_{h=1}^{H} w_h \left(\frac{Obj_{sol} - Obj^*}{Obj^*}\right)\right] \times 100\% \qquad (3)$$

where Obj_{sol} is the minimum objective function value of the algorithm and Obj^* is the best objective function value of all the algorithms considered in the study. H is the number of objectives and

Table 6. Results of each JSSP problem instance for three objectives, obtained with model 1 (20 ants are used)

Problem	Size	ACOMOFS	CPACO	$MACO_4$	MACS	MCAA	PSACO
LA01	10x5	3.88	3.76	0.72	0.17	4.79	5.09
LA02	10x5	1.32	5.26	13.03	6.88	10.24	7.21
LA03	10x5	1.55	4.98	3.11	2.54	6.72	1.28
LA04	10x5	3.16	15.14	0.00	9.27	21.36	0.69
LA05	10x5	0.00	0.00	1.63	0.00	0.00	0.50
ARPD		1.98	5.83	3.70	3.77	8.62	2.95
Average CPU time (seconds)		122	94	99	99	212	124
LA06	15x5	1.07	0.24	10.86	6.61	6.85	12.62
LA07	15x5	3.07	6.04	12.80	4.90	9.34	0.15
LA08	15x5	2.55	2.84	13.34	6.70	3.56	5.59
LA09	15x5	5.44	0.77	13.28	1.44	7.46	6.88
LA10	15x5	2.59	0.64	4.94	3.20	9.65	7.99
ARPD		2.94	2.11	11.04	4.57	7.37	6.65
Average CPU time (seconds)		395	326	331	318	704	366
LA11	20x5	0.28	6.67	12.53	2.77	2.34	9.98
LA12	20x5	0.40	4.00	12.37	0.37	5.59	7.15
LA13	20x5	0.00	4.50	11.35	3.67	0.48	5.61
LA14	20x5	1.43	4.64	18.59	2.81	2.21	13.97
LA15	20x5	0.90	10.52	18.65	4.93	7.99	13.11
LA16	20x5	5.29	2.08	0.00	1.85	5.35	2.77
ARPD		1.38	5.40	12.25	2.73	3.99	8.77
Average CPU time (seconds)		941	801	811	758	1688	833
Mean of ARPDs		2.06	4.51	9.20	3.63	6.50	6.29
# of optimum solutions		7	4	2	3	1	2

w_h is the weighting coefficient which equally considers for each objective. Minimum value of RPD obtains better performance.
Average relative percentage deviation (ARPD) is calculated for each problem set as follows:

$$ARPD = \frac{1}{I}\sum_{i=1}^{I} RPD_i \quad (4)$$

where I is the number of problem instances and RPD_i is the relative percentage deviation (RPD) of the problem instance i. If the values of RPD and ARPD are very close to zero, then it gives better performance.

4. ANALYSIS OF RESULTS

Tables 5 – 7 represent the results obtained for two objectives, three objectives and four objectives JSSP problem instances with model 1, respectively. Results obtained with model 2 represent in Table 8 for two objectives JSSP problem instances. Each model considers different number of ants which is 20 ants in model 1 and 50 ants in model 2. Each table presents the relative percentage deviation (RPD) values of each JSSP instance for each MOACO algorithm, average relative percentage deviation (ARPD) values of small (la01-la05), medium (la06-la10) and large (la11-la16) JSSP instances with their average CPU times in seconds. Also, mean of ARPDs and number of optimum solutions of each MOACO algorithm are presented in each table. If the average relative percentage deviation (ARPD) values are close to zero, then it returns better performances. Each table uses different colors to represent the different stages of RPDs and ARPDs in each row such that, it presents the minimum values in green color, second minimum values in brown color and the third minimum values in blue color.
Table 5 presents the results obtained for two objectives of model 1. According to this table, $mACO_4$ algorithm returns minimum ARPD value for small instances (la01-la05). Therefore, it is the best algorithm as it obtains ARPD value very close to zero. Secondly, PSACO is the best algorithm, as it obtains second minimum ARPD value. Thirdly, ACOMOFS is better than the other MOACO algorithms, as it returns third minimum ARPD value. MACS, CPACO and MCAA algorithms are not performed well, as they return maximum ARPD values. Moreover, ACOMOFS, PSACO, MCAA and MACS algorithms are better than CPACO and $mACO_4$ algorithms for medium instances (la06-la10), as they obtain better ARPD values. Further, MACS, ACOMOFS and MCAA algorithms return better performances than CPACO, PSACO and $mACO_4$ algorithms for large instances (la11-la16), as they obtains ARPD values close to zero. It can be seen that $mACO_4$ algorithm obtains best performance for small instances and worst performances for medium and large instances. Also, PSACO algorithm performs better in small and medium instances, while returning poor performances for large instances. Further, MCAA algorithm obtains good performances only in medium and large instances.
Results obtained for three objectives of model 1, is presented in Table 6. As given in this table, ACOMOFS performs best for small and large instances, as its ARPD values are close to zero. PSACO

Table 7. Results of each JSSP problem instance for four objectives, obtained with model 1 (20 ants are used)

Problem	Size	ACOMOFS	CPACO	$MACO_4$	MACS	MCAA	PSACO
LA01	10x5	1.03	2.74	4.09	5.77	6.45	0.24
LA02	10x5	2.74	3.89	3.59	1.29	6.55	2.49
LA03	10x5	8.63	10.59	0.90	9.70	1.04	7.05
LA04	10x5	1.35	17.89	5.85	15.76	18.61	6.36
LA05	10x5	2.29	4.93	2.09	2.29	2.29	0.30
ARPD		3.21	8.01	3.30	6.94	6.99	3.29
Average CPU time (seconds)		127	96	100	8	294	126
LA06	15x5	2.24	7.33	5.91	0.00	1.15	7.95
LA07	15x5	7.30	13.61	14.00	8.03	9.59	2.88
LA08	15x5	2.77	5.63	8.12	1.42	3.29	0.27
LA09	15x5	0.00	11.71	13.83	6.33	5.16	9.76
LA10	15x5	0.00	8.14	3.93	3.99	4.22	6.01
ARPD		2.46	9.28	9.16	3.95	4.68	5.37
Average CPU time (seconds)		410	332	333	20	972	374
LA11	20x5	0.92	6.18	12.90	1.76	0.00	6.25
LA12	20x5	0.00	4.33	11.82	1.12	3.32	4.93
LA13	20x5	0.82	8.21	10.29	0.00	0.82	2.47
LA14	20x5	9.53	17.34	15.98	11.97	2.73	10.76
LA15	20x5	9.49	19.75	18.44	1.86	14.78	7.84
LA16	20x5	4.81	1.74	12.89	2.42	10.08	6.47
ARPD		4.26	9.59	13.72	3.19	5.29	6.45
Average CPU time (seconds)		984	813	811	40	2348	854
Mean of ARPDs		3.37	9.00	9.04	4.60	5.63	5.13
# of optimum solutions		4	1	1	3	2	4

algorithm is the second best algorithm for small instances, as it returns second best minimum ARPD value. $mACO_4$ and MACS algorithms are significantly better than CPACO and MCAA algorithms in small instances. Nevertheless, CPACO algorithm is the best for medium instances, as it obtains minimum ARPD value. ACOMOFS algorithm is better than MACS, PSACO, MCAA and $mACO_4$ algorithms. ACOMOFS, MACS and MCAA algorithms are better than CPACO, PSACO and $mACO_4$ algorithms in large instances. However, it can be seen that, $mACO_4$ and PSACO algorithms obtain good performance only in small instances. Also, MCAA algorithm obtains good performance only in large instances, while CPACO algorithm obtains best performance only in medium instances.

Table 7 presents the results obtained with four objectives of model 1. According to this table, ACOMOFS algorithm obtains best performances for small and medium instances, as it obtains minimum ARPD values and it returns the second best performance for large instances. PSACO algorithm obtains better performance for small instances, while returning significantly good performances in other instances. However, $mACO_4$ algorithm obtains very poor performance in medium and large instances. Further, MACS algorithm obtains best results in large instances, as it returns minimum value of ARPD and second best performance in medium instances. MCAA algorithm obtains significantly good performance in all the instances. Also, CPACO algorithm returns very poor performances in all the instances, as it obtains very poor values in ARPD.

Results obtained for two objectives with model 2 are presented in Table 8. As given in this table, PSACO algorithm performs best only in small and medium instances, as it returns minimum ARPD values. Also, it returns better values than CPACO and $mACO_4$ algorithms for larger instances. $mACO_4$ algorithm obtains second best performance for small instances, as it returns second best value in ARPD, while it obtains very poor performances for medium and larger instances. Also, ACOMOFS algorithm performs best in larger instances as it obtains minimum ARPD value and significantly performs better in other instances. CPACO algorithm returns significantly good performances in all the instances. Further, MCAA algorithm obtains very poor performance in small and medium instances, as it obtains maximum value in ARPD. However, it obtains better performance in larger instances.

When two, three and four objectives of model 1 are considered (see Tables 5 – 7). it can be shown that ACOMOFS algorithm performs best for most of the instances of all the objectives except in two objective small instances, as it returns significantly good value in ARPD. Also, CPACO algorithm returns significantly good results in larger instances of two and three objectives and it obtains best performance in medium instances of three objectives. But, it obtains poor performance in all the instances of four objectives. Further, $mACO_4$ algorithm returns better performances only for small instances, while obtaining very poor performance for other instances of all the objectives. Moreover, MACS algorithm returns best performances for larger instances, while returning significantly good performances for other instances of all the objectives. MCAA

Table 8. Results of each JSSP problem instance for two objectives, obtained with model 2 (50 ants are used)

Problem	Size	ACOMOFS	CPACO	$MACO_4$	MACS	MCAA	PSACO
LA01	10x5	0.00	0.78	0.00	0.54	2.28	0.48
LA02	10x5	6.41	8.88	6.09	6.50	7.16	0.00
LA03	10x5	5.79	3.38	0.00	4.31	8.77	3.21
LA04	10x5	0.68	0.41	0.53	1.24	5.55	0.00
LA05	10x5	0.00	0.70	1.66	0.00	2.40	1.19
ARPD		2.58	2.83	1.66	2.52	5.23	0.98
Average CPU time (seconds)		319	248	264	265	359	351
LA06	15x5	2.83	6.41	6.44	0.00	6.88	6.06
LA07	15x5	2.72	1.45	4.61	0.17	7.09	1.15
LA08	15x5	5.18	7.68	6.73	6.13	8.08	0.30
LA09	15x5	0.00	4.22	10.36	5.16	6.58	0.55
LA10	15x5	2.29	3.19	8.03	2.68	9.25	1.37
ARPD		2.61	4.59	7.23	2.83	7.58	1.89
Average CPU time (seconds)		1046	850	841	857	1198	1001
LA11	20x5	0.00	1.89	6.50	0.61	1.23	1.67
LA12	20x5	0.00	5.90	9.21	1.30	5.50	3.37
LA13	20x5	0.44	1.03	5.65	1.07	2.03	0.82
LA14	20x5	3.67	5.68	7.78	2.49	2.41	7.36
LA15	20x5	0.69	7.96	10.11	5.47	2.12	5.58
LA16	20x5	1.73	1.50	1.41	0.40	2.39	1.48
ARPD		1.09	3.99	6.78	1.89	2.61	3.38
Average CPU time (seconds)		2511	2070	2035	2017	2895	2156
Mean of ARPDs		2.03	3.81	5.32	2.38	4.98	2.16
# of optimum solutions		7	0	2	4	1	4

algorithm returns very poor performance in small instances while obtains significantly good performance for most of the other instances of all the objectives. Moreover, PSACO algorithm obtains better performances only in small instances of all the objectives and medium instances of two objectives. Therefore, it can be shown that the performances of some algorithms in model 1 depend on the number of objectives, while all the algorithms depend on the size of the problem instance.

When compare the results obtained with model 1 and model 2, it can be shown that, PSACO algorithm performs best among other algorithms in model 2, while it is the second best algorithm in model 1, for small instances of two objectives and medium instances of two objectives. ACOMOFS algorithm performs best in model 1 for small instances of three objectives and medium instances of two objectives, although it is outperformed by PSACO algorithm in model 2. However, it performs best among other algorithms, while it obtains second best performance for large instances of two objectives. MACS algorithm achieves significantly good performance for medium instances of three objectives in model 1, where as it performs better in model 2. MCAA algorithm obtains significantly good performance for medium instances of two objectives in model 1, however it achieves very poor performance in model 2. Therefore, it can be shown that PSACO and MACS algorithms achieve better performances for some instances in model 2 than model 1. Also, MCAA algorithm returns good performance for some instances in model 1 than model 2. Therefore, it is shown that the performances of algorithms depend on the number of ants used.

Nevertheless, ACOMOFS algorithm performs best in most of the instances in both models. Also, PSACO and $mACO_4$ algorithms achieve better performances for small instances of all the objectives considered, in both models. MACS algorithm obtains best performances for larger instances of some objectives in both models. Further, CPACO and MCAA algorithms achieve significantly good performances for most of the instances in both models. Moreover, $mACO_4$ algorithm returns very poor performances for medium and large instances in both models. However, MCAA algorithm achieves very poor performances for small instances of all the objectives in both models. Therefore, it can be shown that, the performances of algorithms are not changed for some instances of both models.

5. CONCLUDING REMARKS

In this contribution, the performances of six recent multi-objective ant colony optimization algorithms are evaluated by applying them to the job shop scheduling problem and taxonomy is presented on these MOACO algorithms. In order to compare the performance of algorithms, sixteen benchmark instances of JSSP are tested on two, three and four objectives by changing the number of ants. A detailed analysis is performed by using the performance indicators: RPD and ARPD.

According to the experimental results in this study, ACOMOFS algorithm is the best performing algorithm in most of the instances,

as it obtains minimum mean of ARPDs, while PSACO performs best for small instances, as it returns minimum of ARPDs. Further, MACS algorithm achieves better performances for larger instances than others and $mACO_4$ algorithm performs better in small instances. Also, MCAA achieves significantly better performances for medium and larger instances. However, CPACO algorithm is outperformed by MACS and ACOMOFS algorithms and it is better than the $mACO_4$ algorithm in medium and larger instances.

Moreover, according to the analysis of the results, it can be concluded that the performances of the MOACO algorithms depend on the number of objectives considered in both models. Also, performances of algorithms for some instances are changed when using the different number of ants. In other words, some algorithms like PSACO and MACS perform better when using the large number of ants in a colony. However, some algorithms like ACOMOFS and $mACO_4$ are not depend on the number of ants used in a colony.

According to our previous study (Ariyasingha and Fernando 2015) and this study, it is shown that the ACOMOFS algorithm performs best in both combinatorial optimization problems: TSP and JSSP. Although, MACS algorithm performs better in TSP, it is not performed well in JSSP. Further, PSACO algorithm achieves best performance for small instances of JSSP, while obtaining slightly better performance in TSP. Also, CPACO algorithm returns good performance in TSP, but it obtains slightly better performance in JSSP. Further, MCAA algorithm achieves slightly good performance only for larger instances of both combinatorial optimization problems. Although, $mACO_4$ algorithm obtains very poor performance in TSP, it achieves better performance only for small instances of JSSP. Therefore, it can be concluded that MOACO algorithms behave differently and obtain different performances for each combinatorial optimization problem. In other words, the performances of MOACO algorithms depend on the combinatorial optimization problem.

The future work that arises out of this study is to apply the other nature-inspired algorithms – such as the particle swarm optimization algorithm and the artificial bee colony algorithm – to the job shop scheduling problem and compare their performances.

Acknowledgement

The authors are grateful to Prof. Arjuna Parakrama, Department of English, University of Peradeniya, for language editing of this paper.

6. REFERENCES

[1] I. Alaya, C. Solnon, and K. Ghedira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, pages 450–457, Los Alamitos, CA, vol. 1, 2007. IEEE Computer Society Press.

[2] 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)*, pages 333–340. Honolulu, Hawaii, USA, IEEE Press, 2007.

[3] D. Angus and C. Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, 3(1):69–85, 2009.

[4] I. D. I. D. Ariyasingha and T. G. I Fernando. Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem. *Swarm and Evolutionary Computation*, 23:11–26, 2015.

[5] B. Baran and 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*, pages 97–102. Insbruck, Austria, February 10-13, 2003.

[6] L. T. Bui, J. M. Whitacre, and H. A. Abbass. Performance analysis of elitism in multi-objective ant colony optimization algorithms. In *2008 Congress on Evolutionary Computation (CEC2008)*, pages 1633–1640. Hong Kong, IEEE Service Center, 2008.

[7] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[8] M. Dorigo and T. Stutzle. *Ant colony optimization*. MIT Press, Cambridge, MA, 2004.

[9] C. Garcia-Martinez, O. Cordon, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180(1):116–148, 2007.

[10] M. R. Garey and D. S. Johnson. *Computers abd Intertracability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, California, 1979.

[11] S. Lawrence. *Resource Constrained Project Scheduling: An experimental Investigation of Heuristic Scheduling Techniques*. Graduate School of Industrial Administration,Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[12] M. Lopez-Ibanez and T. Stutzle. An experimental analysis of design choices for multi-objectives ant colony optimisation algorithms. *Swarm Intelligence*, 6(3):207–232, 2012.

[13] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, New York, 2001.

[14] A. Rabanimotlagh. An efficient ant colony optimization algorithm for multi-objective flow shop scheduling problem. *World Academy of Science, Engineering and Technology*, 75:127–133, 2011.

[15] I. Sabuncuoglu and M. Bayiz. Job shop scheduling with beam search. *European Journal of Operational Research*, 118:390–412, 1999.

[16] P. Semanco and V. Modrak. A comparison of constructive heuristics with the objective of minimizing makespan in the flow-shop scheduling problem. 9(5), 2012.

[17] R. Shrivastava, S. Singh, and G. C. Dubey. Multi-objective optimization of time cost quality quantity using multi colony ant algorithm. *International Journal of Contemporary Mathematical Sciences*, 7:773–784, 2012.

[18] G. I. F. Thantulage. *Ant colony optimization based simulation of 3D automatic hose / pipe routing*. PhD Thesis, School of Engineering and Design, London, UK, March 2009.

[19] A. Udomsakdigool and V. Khachitvichyanukul. Ant colony algorithm for multi-criteria job shop scheduling to minimize makespan, mean flow time and mean tardiness. *Int J Manag Sci Eng Manag*, 6(2):117–123, 2011.

[20] M. Ventresca and B. Ombuki. Ant colony optimization for job shop scheduling problem. In *Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ASC 2004)*. Marbella, Spain, September 2004, CDROM 451–152, 2004.