IMPROVED MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION FOR JOB-SHOP SCHEDULING PROBLEMS

N.I. Anuar[1,2], M.H.F. Md Fauadi[2], A. Saptari[3] and X. Hao[4]

[1]Faculty of Engineering and Technology,
Multimedia University, Jalan Ayer Keroh Lama,
75450 Ayer Keroh, Melaka, Malaysia.

[2]Faculty of Manufacturing Engineering,
Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya,
76100 Durian Tunggal, Melaka, Malaysia.

[3]Department of Industrial Engineering,
President University, Cikarang Baru,
17550 Bekasi, Indonesia.

[4]Changzhou Institute of Technology,
Changzhou, 213032
Jiangsu, China.

Corresponding Author's Email: [2]hafidz@utem.edu.my

Article History: Received 25 August 2020; Revised 29 November 2020;
Accepted 11 December 2020

ABSTRACT: The Job-shop Scheduling Problems (JSP) is a typical production scheduling problem widely applied as a single-objective optimization in existing research. However, this is not suitable for cases in the real world, which normally consist of multi-objective criteria. In this paper, a multi-objective Particle Swarm Optimization (MOPSO) for solving JSP is developed, where it involves three key MOPSO attributes to be improved as identified from the literature which are diversity of swarm solutions, exploitation/exploration mechanisms throughout the search process and premature convergence. In order to address the issues related to these attributes, improvement strategies are implemented that include reinitialization of particles, systematic switch of best solutions and Tabu search-based mutation. The computational results in solving benchmark instances demonstrated that the improved MOPSO performs well in terms of finding non-dominated solutions in different regions of the Pareto fronts with a wider spread and producing a higher percentage of solutions in comparison with other established techniques.

KEYWORDS: *Job-shop Scheduling Problems; Particle Swarm Optimization; Multi-objective Optimization; Pareto Optimality; Metaheuristic Optimization*

1.0 INTRODUCTION

The Job-shop Scheduling Problem (JSP) belongs to one of the best known and most studied production scheduling problems, where the aim is to obtain a sequence of jobs in optimizing one or multiple objectives. Although a single objective is often used, multiple objectives such as cost improvement, machine utilization and on-time deliveries are among the greater concerns encountered in real-world production systems [1–4]. Nevertheless, research works on solving JSP with multiple objectives are still limited compared to the single objective [5–8].

The existing methods used on the standard single-objective model are also impractical to directly be applied to real-world scheduling scenarios in solving multiple objectives simultaneously. A multi-objective case is more challenging to solve as the objectives are normally in conflict with each other where one objective cannot be improved without degrading at least another objective [9]. Instead of a unique, single solution produced as the output in a single-objective case, there exists a number of solutions in a multi-objective case which correspond to the most feasible compromises among the objectives. A multi-objective case is also more challenging to solve as the objectives are normally in conflict with each other, i.e. one objective cannot be improved without degrading at least another objective [10]. Thus, the challenge is to improve the existing methods by first identifying the methods' key attributes in optimizing the JSP with multiple objectives and to address the issues related to these attributes.

There have been several methods proposed to solve multi-objective problems. More recently, swarm intelligence has been developed for this purpose [11], where the success of the Particle Swarm Optimization (PSO) in the optimization of a single objective has inspired researchers to extend the use of this technique to multi-objective optimization. The relative simplicity of PSO, its straightforward implementation and adaptability to a wide range of domains have made it an emerging candidate to be extended for multi-objective optimization [12].

On the other hand, a modified representation of position, movement, and velocity of particles has been proposed [1]. Additionally, the method also performed a diversification strategy to update non-

dominated solutions. Furthermore, Lei [5] introduced the first implementation of multi-objective PSO (MOPSO) in JSP known as Pareto archive PSO (PAPSO), whereby a crowding measure-based archive maintenance was merged with the selection of global best position and a mutation was performed on archive members. The algorithm was also applied to solve fuzzy JSP [12]. Feng et al. [6] presented the multi-objective orthogonal PSO (MOOPSO), where the orthogonal design method was used during the generation of the initial swarm and for selection of more than one global best position. Tavakkoli-Moghaddam et al. [2] employed genetic operators merged within a Pareto archive PSO, along with variable neighbourhood search (VNS), to respectively update and improve particles. They also constructed initial solutions using new elite Tabu search (ETS) method. Another version of the technique [7] applied a character of Scatter Search (SS) to choose a new swarm in every iteration. Wisittipanich and Kachitvichyanukul [8] adopted a combination of four groups of particles within a single swarm with unique movement schemes. An elite group was also utilized to keep the updated non-dominated solutions found by the entire swarm which is used as guidance for flying of particles. In a later study, an integrated metaheuristics method has been proposed to optimize the performance of a job shop scheduling problem in a robotic cell. The proposed method successfully optimized the makespan and the total operational punctuality including the earliness and tardiness [3]. Meanwhile, Meng et al. [4] proposed a dual-population hybrid PSO (NMOPSO) based on a greedy strategy, where one population is to lead another population to convergence. The simulated annealing was utilized as a local search. It also performed crossover and mutation operations on individuals in the two populations.

One main shortcoming of the existing PSO design is that the swarm is inclined to cluster rapidly towards the current best location, resulting in a stagnation of the searching process when the swarm becomes stuck at a local optimum. The problem is magnified especially when dealing with multi-objective problems. Therefore, as established in these research works [1–8, 13], there are many strategies adopted to design MOPSO algorithm in solving JSP to cope with this issue. Different strategies can be combined simultaneously to reap the collective advantages of each individual strategy. Based on these works available in the literature, this research has identified three key MOPSO attributes to be improved to solve the JSP. Consequently, an improved MOPSO is proposed, where key improvement strategies are implemented in the existing PSO design to address the issues related to these attributes. Additionally, this study utilizes dominant-based

optimization approach in order to provide flexible way to determine the feasible options for optimization. The proposed method may be utilized as the reference architecture to optimize multi-objective scheduling problem using dominance-based approach.

The remainder of the paper is structured as follows: Section 2 describes the methodology and the improved MOPSO is proposed. In Section 3, the results of solving benchmark instances are presented and discussed. The paper concludes in Section 4.

2.0 METHODOLOGY

2.1 Standard PSO

The particles navigate throughout the search space in a PSO algorithm by following the current best particles, whereby they have velocities that guide the movement of the particles [14]. The particles update their velocities and positions using Equations (1) and (2), respectively:

$$v_{id} = w \times v_{id} + c_1 \times r_1 \times \left(p_{id} - x_{id}\right) + c_2 \times r_2 \times \left(p_{gd} - x_{id}\right) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

for i = 1, 2, …, N_s, d = 1, 2, …, D, g = index of the best particle in the swarm, where N_s = swarm size, D = problem dimension, w = inertia weight, c_1 and c_2 = learning factors, r_1 and r_2 = random numbers in the range [0, 1], x_{id} = position of i^{th} particle, v_{id} = velocity of i^{th} particle, p_{id} = best position of i^{th} particle, and p_{gd} = best position in the swarm.

2.2 Problem Description

In JSP, a collection of jobs is to be scheduled on a collection of machines in a given order. A job contains a number of operations, where the operation indicates the processing of a job on a particular machine. Every operation has a job duration, which is known in advance, specific for every machine. The sequence of machines occupied by the job is the precedence constraints for that job [15]. In this study, the objective of the problem is to minimize the makespan and total tardiness simultaneously, in which the formulae are given by Equations (3) and (4), respectively;

$$C_{\max} = \max\{C_i\} \quad (3)$$

$$T_{tot} = \sum T_i \quad (4)$$

where C_i is the completion time of job *i*, C_{max} is the makespan, $T_i = \max(C_i - d_i, 0)$ is the tardiness of job *i* that is when completion of job *i* is after its due date, d_i and T_{tot} is total tardiness. As dominance-based approach is used to solve the problem, the optimization for each function will be carried out independently.

2.3 Solution Representation

A solution representation refers to the transformation procedure from a particle in PSO to a schedule in JSP. The representation implemented in this study is based on the random key representation and the smallest position value (SPV) rule presented in [16]. According to the SPV rule, each job in a schedule is first sequenced according to the particle's continuous position values sorted in ascending order. In this new sequence, every job will afterwards be scheduled consecutively on every machine by adhering to the precedence constraints of JSP. Thus, any permutation of this representation always leads to a feasible schedule.

2.4 Archive Maintenance

An external archive is used in this study similar to [1] throughout the searching process to store the set of non-dominated solutions discovered by the swarm so far. In every iteration, all positions of particles are updated and the objective functions associated with these positions are re-evaluated to obtain the new fitness values. Both of these fitness values and the members of the archive are screened afterwards to identify the next non-dominated solutions, which are then stored back in the archive. When the archive is full, the members who are duplicates in terms of fitness values are removed. Using this procedure, the size of the archive will not exceed the pre-determined maximum capacity. Furthermore, if the solutions are similar, they are susceptible to be trapped in a local optimum. Thus, this procedure will maintain the required capacity in the archive, while ensuring the non-dominated solutions are different from time to time.

2.5 Improved MOPSO in Solving JSP

As established in these research works [1–8, 13], there are many strategies adopted to design MOPSO algorithm in solving JSP. Based on these works available in the literature, this paper has identified three key MOPSO attributes to be improved to solve the JSP: i) diversity of swarm solutions, ii) exploitation/exploration mechanisms throughout the search process, and iii) premature convergence.

These attributes were consolidated in the form of a general MOPSO structure in solving JSP, as shown in Figure 1. The structure illustrates the context of integrating the strategies for improvements within two phases of MOPSO - initialization and swarm-evolving phases. It also illustrates the multiple aspects of improvements that can be carried out during the swarm-evolving phase including solution diversity, exploitation/exploration mechanisms and premature convergence. These aspects may interrelate with each other as well. With regard to the improvement in the diversity of solutions, it is not only targeted to the non-dominated solutions on the Pareto front but also includes the diversity of the initial solutions and current solutions of the swarm. In essence, this structure represents the basis of our perspective on improvement strategies available in the literature.

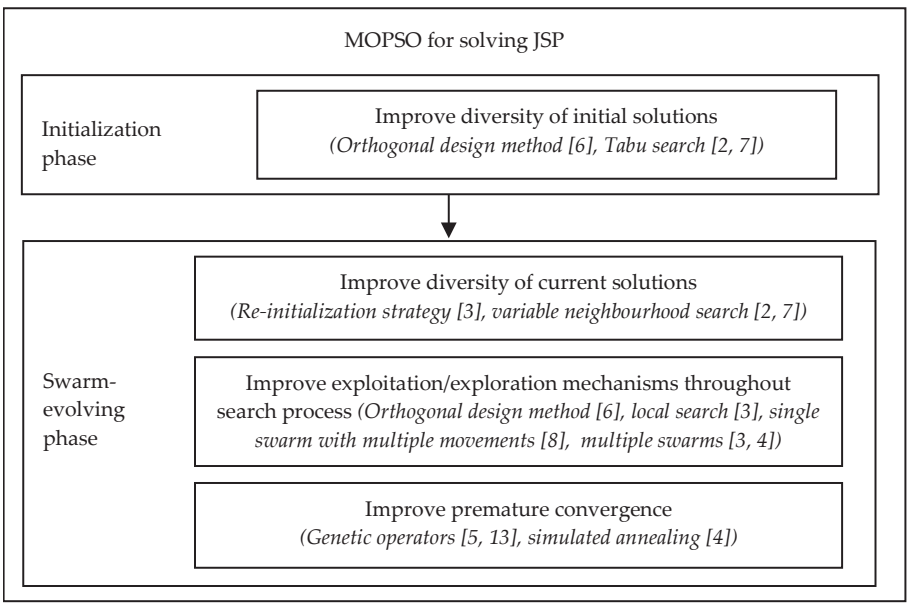

Figure 1: General MOPSO structure with improvement strategies in solving JSP

An improved MOPSO is proposed based on the structure in Figure 1, where the improvement strategies are implemented to address the issues related to these attributes. Given that x_i = position of the i^{th} particle, Fx_i = fitness value of x_i, v_i = velocity of the i^{th} particle, A = external archive, p_i = best position of the i^{th} particle, Fp_i = fitness value of p_i, p_g = best position in the swarm, Fp_g = fitness value of p_g, and F_w = worse fitness value among all Fp_i, the improved algorithm can be summarized in the following steps:

Step 1: Initialize the swarm with random positions and velocities. Evaluate the fitness values of the swarm.
Step 2: Check for duplicates in the fitness values and reinitialize the positions of the affected particles. Evaluate their fitness values. Repeat this step until no duplicate exists.
Step 3: Initialize A with the non-dominated solutions.
Step 4: For each particle i, update v_i and x_i according to Equations (1) and (2), respectively. Evaluate Fx_i.
Step 5: If Fx_i is similar to Fp_g since the last m iterations, reinitialize x_i and v_i. Evaluate Fx_i.
Step 6: Apply Pareto dominance to select Fp_i. Update A.
Step 7: Select a solution randomly in A as Fp_g.
Step 8: If there is no improvement in Fp_g since the last d iterations, preserve Fp_i and Fp_g alternately for a duration of d. Update A. Otherwise, repeat Steps 6 and 7.
Step 9: If A contains only one unique member or a single member, randomly select F_w as Fp_g.
Step 10: Update p_i and p_g.
Step 11: Repeat Steps 4 until 10 while maximum iteration is not attained

2.5.1 Reinitialization of Particles to Diversity Swarm Solutions

The swarm solutions in MOPSO can be viewed according to the three different stages in the algorithm which are initial solutions, current solutions and best solutions or non-dominated solutions. Most MOPSO techniques generally target to improve the diversity of the non-dominated solutions only which is during the best fitness evaluation stage. But if the solution diversity is also improved beforehand during the stages of initialization and current fitness evaluation, it could ensure that the search conducted is more widespread from the beginning, leading to reasonably better diversity in the non-dominated solutions.

In terms of improving the diversity of the initial solutions, several works implemented orthogonal design method [6] or Tabu search [2, 7] during the initialization stage. On the other hand, a majority of MOPSO start with initializing the positions randomly [1, 3–5, 8, 13]. The improved MOPSO in this paper also uses the standard random initialization procedure but applies an additional step which is checking for the duplicates in fitness values and replacing them by

reinitializing the positions of the affected particles. This is to avoid having similar starting solutions among the initial swarm, thus it is able to obtain a distinct permutation of schedules that would produce unique fitness values.

Next, to improve the diversity of the current solutions, several works implemented an integrated Mixed Interger Programming (MIP) – metaheuristics strategy [3] and variable neighbourhood search [2, 7] during the current fitness evaluation stage. The re-initialization strategy in [3] involves re-initializing the positions and velocities of all particles randomly after a certain duration of iterations and it will be repeated after each predetermined duration of iterations. The improved MOPSO in this paper also applies a random re-initialization of positions and velocities but only to selected particles whose current fitness values are identical to the global best fitness value since the last m iterations. This is to keep these particles from converging prematurely on the best fitness value obtained so far, thus it is able to diversify the swarm from time to time to prevent the stagnation of the searching process.

2.5.2 Systematic Switch of Search Mechanism

The performance of MOPSO is mostly influenced by its two abilities: i) exploitation around a promising region in the search space in order to improve the quality of a current solution, and ii) exploration throughout various areas in the search space in order to discover prospective solutions in other regions. These two abilities should be balanced during the search process to obtain good performance.

In terms of improving the exploitation/exploration mechanisms throughout the search process, several works implemented orthogonal design method for selection of more than one global best position [6], single swarm with multiple movements [8] and multiple swarms [3–4]. The single swarm with multiple movements in [8] adopted a combination of four groups of particles within a single swarm with unique movement schemes. The improved MOPSO in this paper also utilizes a single swarm where the particles move as a whole towards a global best location, but they will be redirected to another location when there is no improvement in the global best solution within a pre-defined duration.

The density-estimation procedure is proposed by a majority of MOPSOs in order to choose the global best solution in the non-dominated set [2, 5, 7, 13]. Generally, a single solution in that set which

belongs to a sparse region is likely to be selected as the global best. These algorithms will tend to direct the swarm movement towards this single particle for a relatively prolonged period of time, thus potentially converging prematurely on the best fitness value obtained so far.

In this research, the discrete MOPSO ensures that there is an alternation or rotation among the non-dominated solutions accepted as the personal and global best solutions. The selected best solution is also preserved for a pre-determined duration to give sufficient chance for the algorithm to perform the necessary exploitation around promising areas detected so far in the search space. On the other hand, to facilitate the exploration of the search space, the best solution is selected at random in a periodic manner. It will alternate systematically between the random selection and the preserved selection in order to balance between the exploration and exploitation phases.

The algorithm will first run with the random selection of the best solutions and if there is no improvement in the global best solution since the last d iterations, it will invoke a mechanism to preserve the current best solution for a pre-determined duration of d. Afterwards, the algorithm will invoke the random selection again, also for a duration of d. The procedure will be repeated to alternate the selection of the best solution not only randomly, but also in the manner of selecting the best solutions that have not been chosen so far. The chosen solution will be maintained for a duration of d. If a new global best solution is found, the algorithm will be back to applying the random selection of the best solutions again in order to diversify the search.

A systematic switch of one best solution to another is carried out to ensure every solution has a chance to participate in the searching process by guiding the flight of the particles at least once throughout the iterations. Moreover, it can keep the swarm from being led only by a certain solution for a prolonged interval, thus potentially being biased towards a certain region of the search space. In addition, the best solution is maintained for a fixed duration so that the swarm is able to intensify the search around potential region discovered so far in the search space, before moving on to explore another potential region when it is time for the algorithm to switch to another best solution.

2.5.3 Tabu Search-Based Mutation for Premature Convergence

Premature convergence stemmed from a rapid loss of diversity within the swarm where particles are stagnating about a sub-optimal location and become incapable to produce new solutions. This behaviour can

cause the entire swarm to be trapped in a local optimum from which it is challenging to escape. Since the global best particle attracts all particles of the swarm, it is possible to guide the swarm out of this state through a mutation or change in the global best solution.

In terms of improving the premature convergence, several works implemented genetic operators [5, 13] and simulated annealing [4]. The genetic operator in [5, 13] involves a mutation on chosen archive members involving one real variable corresponding to sub-string of a chromosome of each archive member. The improved MOPSO in this paper also employs a kind of 'mutation' or change, which involves the selection criterion of the global best solution, based on the experience of the particles throughout the flight.

Referring to the previous subsection, if there is only one unique non-dominated solutions or a single non-dominated solution to alternate as the global best solution during the duration d, the swarm becomes susceptible to converge prematurely on that best solution obtained so far. In this study, the algorithm will invoke a mechanism of Tabu search and make use of the experience of the particles, where it is permitted to select a worse solution randomly among the personal best solutions of the particles as the global best solution. This is carried out since there is no other solution to alternate as the global best solution and to give the algorithm a chance to move away from a local minimum. By taking advantage of the prior experience of the swarm, it may discover another path to a better solution which is not explored previously.

3.0 RESULTS AND DISCUSSION

In this section, the performance of MOPSO with the improvement strategies discussed in the previous section is evaluated in the case of solving JSP. First, an experimental study is conducted to assess the effect of implementing these improvement strategies in MOPSO compared to when no strategy is implemented. Three test instances of varying complexities known as FT06, FT10 and FT20 were selected from the OR-Library [17]. The objective is to find the non-dominated solutions in minimizing the makespan and total tardiness. Both methods are programmed using MATLAB on an Intel Core i5-6200U CPU at 2.3 GHz with 4GB of RAM running on Windows 10. The swarm size is 100 for all instances. For FT06 problem, the inertia weight is decreased linearly from 0.9 at the beginning to 0.4 at the end of the run. While for FT10 and FT20 problems, it is set from 2.8 to 0.4. Both learning factors are set to 2.0. The maximum iterations are 6000 for FT06 and

10000 for both FT10 and FT20, where both methods are executed randomly for 20 runs in every instance. The parameters *m* and *d* are set to 1 and 300, respectively. Due date data for FT06, FT10 and FT20 is set according to Lei and Wu [18].

The results of the improved MOPSO algorithm are compared with the basic MOPSO in terms of Pareto fronts as illustrated in Figure 2. It shows that basic MOPSO is competitive with improved MOPSO for FT06 instance, where both methods managed to find a similar non-dominated set as depicted in Figure 2(a). For FT10 and FT20, improved MOPSO was able to obtain 20% and 75% more solutions than basic MOPSO, as shown in Figures 2(b) and 2(c) respectively. Furthermore, 70% and 75% of the solutions obtained by basic MOPSO for the respective FT10 and FT20 instances are dominated by those obtained using improved MOPSO. Besides, there is no solution produced by improved MOPSO that is dominated by basic MOPSO.

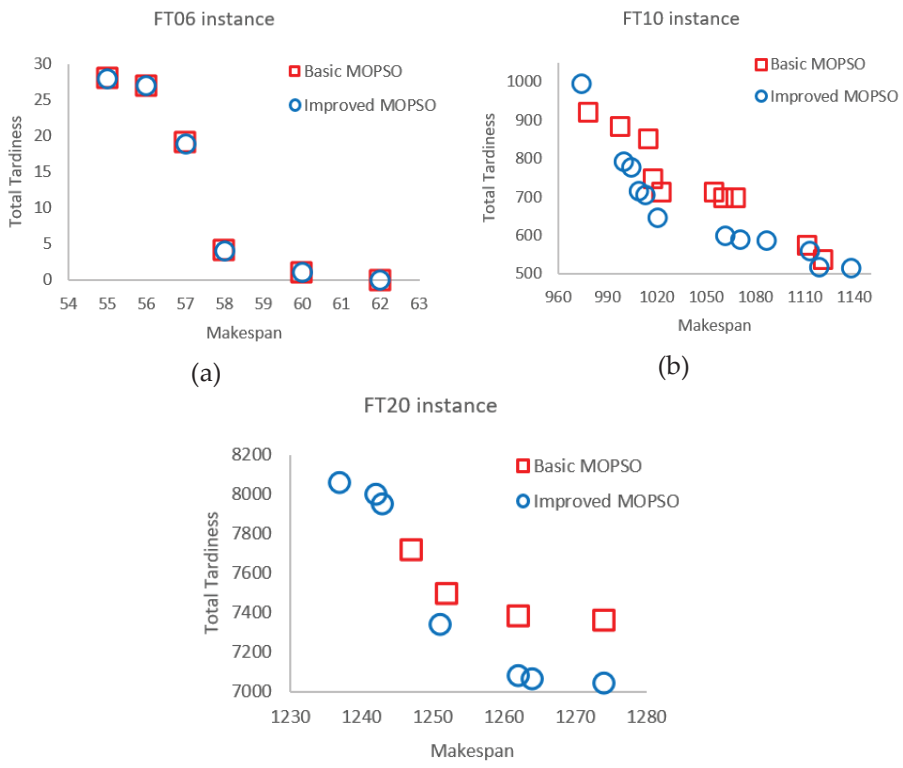

Figure 2: Pareto fronts of basic and improved MOPSO for (a) FT06, (b) FT10 and (c) FT20 instances

Table 1 gives the average computational times consumed by both methods in completing one run. The basic algorithm consumed a slightly lesser computational time in FT06 instance to obtain a similar non-dominated set as the improved one. On the other hand, the improved algorithm expended slightly more computational times in FT10 and FT20 instances, but its results significantly outperformed the results produced by the basic MOPSO.

Table 1: The average computational times per run (in seconds)

Instance	Basic MOPSO	Improved MOPSO
FT06	236	368
FT10	976	1013
FT20	1043	1061

Based on the results, for solving a test problem of a lower complexity like FT06, the use of improvement strategies may not be necessary since the basic algorithm is capable to solve it in a time-efficient manner. On the contrary, for solving test problems of a higher complexity like FT10 and FT20 instances, the basic algorithm is demonstrated to be not effective and it is outperformed by the improved algorithm with a corresponding increase in computational costs. For this type of problems, it is clear that MOPSO with improvement strategies is superior and capable of producing a higher number of non-dominated solutions.

After the improved algorithm has been demonstrated to be effective in solving JSP, its results are compared with the ones reported in the literature employing other metaheuristic techniques. In Figure 3, the results of the improved MOPSO are compared to those obtained using other existing algorithms including Strength Pareto Evolutionary Algorithm (SPEA) and Crowding Measure-Based Multi-Objective Evolutionary Algorithm (CMOEA) given by Lei and Wu [18]. On the other hand, SPEA method as proposed by Zitzler and Thiele [9] utilizes a fitness assignment based on principles of coevolution and the niching technique founded on the concept of Pareto dominance. Meanwhile, CMOEA applies a crowding-measure-based method to amend the external population and allocate distinct fitness for members in the population.

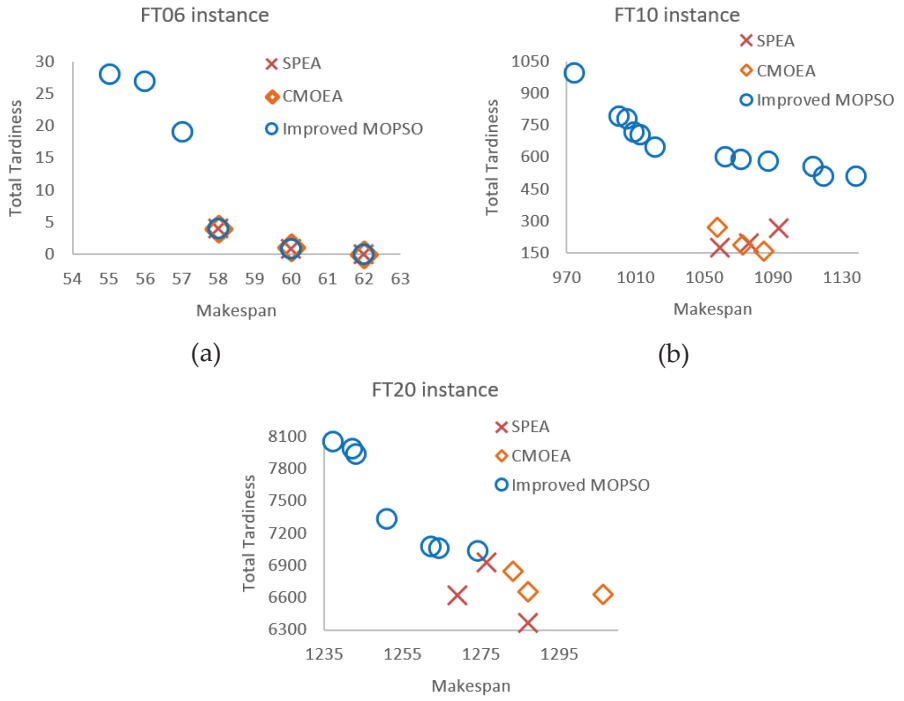

Figure 3: Pareto fronts of SPEA, CMOEA and improved MOPSO for (a) FT06, (b) FT10 and (c) FT20 instances

Figure 3 shows that improved MOPSO, SPEA and CMOEA managed to find a set of non-dominated solutions on different regions of the Pareto fronts. The solutions of SPEA and CMOEA tend to cluster at the lower-half region, while the solutions of improved MOPSO incline to occupy the upper-half region of the front, although its spread is much wider than those belong to SPEA and CMOEA. For FT06 instance as in Figure 3(a), improved MOPSO was able to obtain 50% more solutions than SPEA [9] and CMOEA [18]. Although some solutions of improved MOPSO in FT10 and FT20 as in Figures 3(b) and 3(c) are dominated by SPEA and CMOEA, both methods do not manage to find solutions on another region of the front, unlike the improved MOPSO. It is found that two solutions of FT10 and one solution of FT20 generated by SPEA are actually dominated solutions. Overall, even though improved MOPSO has some dominated solutions, it managed to find 67% to 100% of the total non-dominated solutions compared to SPEA and CMOEA, as shown in Table 2. This will provide the decision-maker a wider range of possible solutions to choose from based on the requirement of the production system; either to prioritize the makespan or the total tardiness, or to strike a balance between the two objectives.

With regards to FT10 instance as illustrated in Figure 3(b), the best makespan value obtained by improved MOPSO is 974, which is close by 4.73% to the optimal makespan value of 930. This makespan value by improved MOPSO is also better compared to SPEA by 8.03% and CMOEA by 7.85%. On the other hand, the best total tardiness value obtained by improved MOPSO is 514 which is outperformed by SPEA by 64.98% and CMOEA by 69.65%.

In a production system, the decision-maker may be concerned with achieving a better makespan value since makespan signifies a good measure in performance for job-shop; a schedule with minimum makespan is a sign of high throughput rate and high machine utilization which is very important in production systems since machines are costly to procure and operate. Minimizing the makespan is also associated with minimizing the machine idle time. On the other hand, the tardiness of jobs depends on job due dates, where they are typically set by the customers. The decision-maker may examine what due date setting that result in jobs finishing late or on time so that the customers can be informed. The decision-maker may also propose to the customers different due date options that fulfill the makespan objective in order for jobs to complete close to their assigned due dates. In this way, the tardiness value is able to be reduced while still achieving a good makespan value [7–8, 15].

Table 2: Percentage of non-dominated solutions found based on Figure 3

Instance	Percentage of non-dominated solutions found by each algorithm		
	SPEA (%)	CMOEA (%)	Improved MOPSO (%)
FT06	50	50	100
FT10	11	33	67
FT20	22	33	67

The core underlying difference among these algorithms is that SPEA and CMOEA concentrate on the application of fitness assignment procedures whereas improved MOPSO focuses on escaping the local minimum by diversifying the swarm periodically throughout the different phases of the algorithm. The implementation of various strategies in improved MOPSO are able to prevent the particles from converging towards a specific region on the Pareto front, thus a more widespread search can be conducted to discover new solutions in other regions.

4.0 CONCLUSION

In this paper, an improved MOPSO for solving JSP was developed based on proposed key improvement strategies identified in the literature that include reinitialization of particles, systematic switch of best solutions and Tabu search-based mutation. The computational results in solving benchmark instances demonstrated that the improved MOPSO performs well in terms of finding non-dominated solutions in different regions of the Pareto fronts with a wider spread and producing a higher percentage of solutions in comparison with other established techniques. By using the improved MOPSO, the makespan values for all instances are consistently reduced compared to the basic MOPSO, SPEA and CMOEA with improvements varying from 0.41% to 13.11%. In a nutshell, the rapid clustering problem of a standard PSO has been successfully resolved by improving three searching mechanisms. Nevertheless, the improved MOPSO exhibits a potential for further improvement. The next research plan will attempt to include the density-estimation procedure for the global best selection and apply other performance metrics to measure and evaluate the convergence and diversity of non-dominated solutions.

ACKNOWLEDGMENTS

The authors would like to thank Universiti Teknikal Malaysia Melaka and Multimedia University in providing facilities for the research project to be conducted.

REFERENCES

[1] D.Y. Sha and H.H. Lin, "A multi-objective PSO for job-shop scheduling problems", *Expert Systems with Applications*, vol. 37, no. 2, pp. 1065–1070, 2010.

[2] R. Tavakkoli-Moghaddam, M. Azarkish and A. Sadeghnejad-Barkousaraie, "Solving a multi-objective job shop scheduling problem with sequence-dependent setup times by a Pareto archive PSO combined with genetic operators and VNS", *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 5–8, pp. 733–750, 2011.

[3] X. Li, X. Yang, Y. Zhao, Y. Teng and Y. Dong, "Metaheuristic for Solving Multi-Objective Job Shop Scheduling Problem in a Robotic Cell", *IEEE Access*, vol. 8, pp. 147015–147028, 2020.

[4] Q. Meng, L. Zhang and Y. Fan, "Research on Multi-objective Job Shop Scheduling with Dual Particle Swarm Algorithm Based on Greedy Strategy", *Wireless Personal Communications*, vol. 103, no. 1, pp. 255–274, 2018.

[5] D. Lei, "A Pareto archive particle swarm optimization for multi-objective job shop scheduling", *Computers and Industrial Engineering*, vol. 54, no. 4, pp. 960–971, 2008.

[6] M. Feng, S. Tang, H. Li, W. Li, C. Guo, Y. Xu, Y. Zhang, "Orthogonal particle swarm optimization for multi-objective job shop scheduling problems", in International Conference on Computational Intelligence and Natural Computing, China, 2010, pp. 256–260.

[7] R. Tavakkoli-Moghaddam, M. Azarkish and A. Sadeghnejad-Barkousaraie, "A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem", *Expert Systems with Applications*, vol. 38, no. 9, pp. 10812–10821, 2011.

[8] W. Wisittipanich and V. Kachitvichyanukul, "An Efficient PSO Algorithm for Finding Pareto-Frontier in Multi-Objective Job Shop Scheduling Problems", *Industrial Engineering and Management Systems*, vol. 12, no. 2, pp. 151–160, 2013.

[9] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms - A Comparative Case Study and the Strength Pareto Approach", *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[10] H.M. Asih, K.E. Chong and M. Faishal, "Capacity Planning and Product Allocations under Testing Time Uncertainty in Electronic Industry", *Journal of Advanced Manufacturing Technology*, vol. 12 no. 1, pp. 103-116, 2018.

[11] H. Afaq and S. Saini, "Swarm Intelligence based Soft Computing Techniques for the Solutions to Multiobjective Optimization Problems", *International Journal of Computer Science Issues*, vol. 8, no. 3, pp. 498–510, 2011.

[12] A. Banks, J. Vincent and C. Anyakoha, "A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications", *Natural Computing*, vol. 7, no. 1, pp. 109–124, 2008.

[13] D. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems", *International Journal of Advanced Manufacturing Technology*, vol. 37, no. 1–2, pp. 157–165, 2008.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization", in IEEE International Conference on Neural Networks, Australia, 1995, pp. 1942–1948.

[15] J.F. Muth and G.L. Thompson, *Industrial Scheduling*. New Jersey: Prentice Hall, 1963.

[16] M.F. Tasgetiren, Y.C. Liang, M. Sevkli and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem", *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.

[17] J.E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail", *The Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[18] D. Lei and Z. Wu, "Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling", *International Journal of Advanced Manufacturing Technology*, vol. 30, no. 1–2, pp. 112–117, 2006.