International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 051-061

Received: 27 December, 2020; Accepted: 4 May, 2021; Published: 10 June, 2021

A Study on Multi-Objective Particle Swarm Optimization in Solving Job-Shop Scheduling Problems

Nurul Izah Anuar[1,2] and Muhammad Hafidz Fazli Md Fauadi[2]

[1] Faculty of Engineering and Technology, Multimedia University,
Jalan Ayer Keroh Lama, 75450 Ayer Keroh, Melaka, Malaysia
nurulizah.anuar@mmu.edu.my

[2] Faculty of Manufacturing Engineering, Universiti Teknikal Malaysia Melaka,
Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
hafidz@utem.edu.my

***Abstract*: Particle Swarm Optimization (PSO) is a population-based metaheuristic that was modelled based on the social interaction and communication of organisms, such as a flock of birds or a school of fishes. It is widely applied to solve a single-objective function in existing research, but this is not suitable for cases in the real world, which normally consist of multiple-objective criteria. Such cases encompass the Job-shop Scheduling Problem (JSP), where it is a typical production scheduling problem and belongs to one of the most difficult problems of combinatorial optimization. Subsequently, the multi-objective Particle Swarm Optimization (MOPSO) was established to accommodate the requirement of multiple-objective cases encountered in real-world production systems. Nevertheless, research works on solving JSP with multiple objectives using MOPSO are still limited compared to the single objective. In this study, comparison and discussion of existing works, in terms of objective functions, test problems, multi-objective optimization methods, scheduling constraints, strategies and performances are conducted. This study also highlights current MOPSO improvement strategies and the aims of their implementation in solving JSP. Finally, this study proposes a MOPSO model in solving JSP that consolidates these aspects of improvement strategies, which would set the path for future directions of research provided in the final part of the paper.**

***Keywords*:** Particle Swarm Optimization, Combinatorial Optimization, Pareto Optimality, Multi-objective Optimization, Job-shop Scheduling Problems, Production Scheduling**.**

I. Introduction

The Job-shop Scheduling Problem (JSP) is amongst one of the best known and most studied scheduling problems, with widespread applications in a variety of settings. The problem appears in many important applications involving transportation and logistics, production planning, information processing and communication [1]. It is well-known that even very simple versions of the JSP belong to the class of non-deterministic polynomial-time hard (NP-hard) [2], therefore an exact solution cannot be obtained in an acceptable computational time [3]. This is notably demonstrated by the fact that an instance of JSP comprising 10 jobs and 10 machines introduced in 1963, which is currently available in the OR-Library [4], remained unsolved for more than 20 years. It was eventually solved in 1989 [5] and has been used as a benchmark for most algorithms in this field since then.

The goal behind production scheduling is to obtain a sequence of jobs with the purpose that one or more objectives are optimized. The objective in question for JSP is commonly to find the optimum value of the makespan, i.e. the minimum completion time of the final job to leave the system [6]. Although a single objective like makespan is often used, the achievement of multiple objectives such as the improvement of cost, machine utilization and on-time deliveries are among the greater concerns encountered in the real-world production systems [7]-[8]. Thus, a single-objective problem needs to be extended to a multi-objective problem, whereby the optimization of the multiple objectives is carried out simultaneously and the objectives are generally in conflict with each other. They are more challenging to solve compared to single-objective cases, as there is no unique, single solution. Instead, there is a set of acceptable trade-off solutions which correspond to the most feasible compromises among the objectives [9].

There have been several methods and algorithms proposed to solve multi-objective problems. More recently, swarm intelligence approaches have been developed for this purpose [10]-[11], where the success of Particle Swarm Optimization (PSO) in solving single-objective optimization problems has inspired research works in the extension of this method to problems of multi-objective optimization. PSO has been observed to be capable of producing superior solutions at a very low computational cost, where it has performed considerably well in a broad range of applications [12]. In

comparison with evolutionary algorithms, PSO has inherent advantages on scheduling problems. For instance, it does not have to devise special mutation or crossover operators to inhibit the presence of illegal individuals. It also contains less complex mathematical calculation and requires fewer parameter adjustments, which furnishes it with high search efficiency. The relative simplicity of PSO, its straightforward implementation and adaptability to a wide range of domains have rendered it an emerging prospect to be extended for multi-objective optimization [13].

A preliminary attempt to extend the PSO scheme was presented by Moore and Chapman [14] in solving multi-objective problems, designed with personal best repositories and a global best repository. Another one of the first Pareto-based PSO schemes, Coello and Lechuga [15] proposed the Multi-objective PSO (MOPSO). The improved version of the algorithm with a mutation operator and constraint-handling mechanism was proposed later [16]. A Dynamic Neighbourhood PSO (DNPSO) was introduced by Hu and Eberhart [17]. A modified dynamic neighbourhood PSO algorithm was presented [18] as a subsequent improvement that decreased the computational time. Parsopoulos and Vrahatis [19]-[20] investigated the capability of PSO to produce non-dominated solutions by employing a multi-swarm variant of PSO, Vector Evaluated Particle Swarm Optimization (VEPSO). It was modelled based on the idea of Vector Evaluated Genetic Algorithm (VEGA) developed by Schaffer [21]. Subsequent work showed the method could be enhanced using a parallel version of the VEPSO [22]. Fieldsend and Singh [23] continued the effort by proposing a novel technique for selecting the best global and local individuals for multi-objective PSO swarm members.

Recently, numerous research articles have conducted various discussions regarding multi-objective Particle Swarm Optimization (MOPSO) that touch upon multiple facets of applications. Lalwani et al. [24] conducted studies regarding applications of MOPSO in diverse areas including the types of MOPSO variants. A taxonomy was proposed by Reyes-Sierra and Coello [25] to classify the current MOPSO techniques, in addition to its corresponding survey of approaches. A descriptive overview of the state-of-the-art MOPSO variants was provided by Parsopoulos and Vrahatis [26], along with the future trends and most active research directions. Fieldsend [27] discussed the effect of the different global best (gbest) and personal best (pbest) selection methods on MOPSO search, along with turbulence variable within MOPSO algorithms. Some articles briefly include MOPSO as part of the more comprehensive review, such as by Zhang et al. [12], Banks et al. [13], and Song and Gu [28]. There is also an article that specifically discussed the applications of MOPSO in a particular area i.e., Li and Yang [29] presented a survey on MOPSO applications in power system economic dispatch problems. However, despite the successful implementations in diverse areas, the MOPSO applications in JSP are still very limited. Further works of improving MOPSO algorithm and the challenges in employing JSP with multiple objectives need to be further investigated. In this paper, the discussion of MOPSO and its application of solving JSP is carried out in detail. The study done in this paper is then leveraged to establish a proposed model of MOPSO in solving JSP. This paper, thus, may aid as a helpful reference for practitioners involved in solving JSP with multiple objectives by using PSO.

The remainder of the article is structured as follows: Section II describes the basic concept of multi-objective optimization and its general categorization, followed by Section III on the standard and multi-objective PSO descriptions and algorithms. Section IV provides the background of JSP, along with the related objective functions. In Section V, a comparison is made on existing works concerning the application of MOPSO in solving JSP. Section VI discusses the summary on variations and improvements of MOPSO implementation in JSP, leading to a proposed MOPSO model to solve JSP and the potential directions of future research. The paper concludes in Section VII.

II. Multi-Objective Optimization

A multi-objective optimization involves a problem with a number of objectives to be achieved and these objectives are generally conflicting i.e., one objective cannot be made better without making at least another one objective to be worse in value [30]. Mathematically, a multi-objective optimization problem can be formulated using equation (1):

$$\begin{aligned} &\text{Optimize} && f_i(x) && i = 1,2,\ldots,n \\ &\text{Subject to} && g_j(x) \le 0 && j = 1,2,\ldots,p \\ & && h_k(x) = 0 && k = 1,2,\ldots,q \end{aligned} \tag{1}$$

The search variable x is a vector of m decision variables $x = (x_1, x_2, ..., x_m)^T$ and $x \in (x_{\min}, x_{\max})$. The objective function $f_i(x)$ is a vector having n objectives to be optimized simultaneously. $g_j(x)$ and $h_k(x)$ represent inequality and equality constraints, respectively.

With the aim of evaluating the solutions of multi-objective problems, we normally make use of the notions of Pareto optimality and Pareto dominance [31]. For a minimization problem, an objective vector $u = (u_1,\ldots,u_k)$ is said to dominate another objective vector $v = (v_1,\ldots,v_k)$ in the search space, if and only if, no component of u is greater than the corresponding component of v and at least one component is smaller. This property is described as Pareto dominance. In this case, we also say that u is non-dominated i.e., not dominated by any other solution. A set of solutions that are non-dominated with respect to each other is described as a Pareto optimal set [32].

There are two general schemes to solve the problems in multi-objective optimization [26]. The categorization made is not stringent, however, since there are approaches that combine characteristics from the two schemes, as well as approaches that could not fit any of the two schemes. The first scheme comprises transforming the multi-objective problems into single-objective ones, taking advantage of the PSO efficiency as a single-objective optimizer. These include aggregation strategy [20], [33], lexicographic ordering strategy [17]-[18] and multi-swarm strategy [20], [22], [34].

These strategies either merge the objective functions into a single combined function or handle every objective function separately and consecutively, to be optimized. The second scheme comprises employing all objective functions simultaneously to be optimized [14]–[16], [23]. According to the notion of Pareto optimality, it produces a set of Pareto optimal solution or a representative subset. A solution is part of the Pareto set if there does not exist another solution that is better in at least one of the objectives without being worse in another objective. The non-dominated solutions represent diverse compromises or trade-offs among the objectives.

III. Particle Swarm Optimization (PSO)

Each potential solution in the PSO algorithm is called a 'particle'. Each particle navigates throughout the search space by pursuing the current best particles, where they have velocities that guide the particles' movements. This velocity is maintained continuously by the experiences of the particle itself and its neighbours or the entire swarm [35]. The particle updates its velocity and position using equations (2) and (3) respectively, as follows:

$$v_{id} = w \times v_{id} + c_1 \times r_1 \times \left(p_{id} - x_{id}\right) + c_2 \times r_2 \times \left(p_{gd} - x_{id}\right) \quad (2)$$

$$x_{id} = x_{id} + v_{id} \quad (3)$$

for $i = 1,2,3,\ldots,N_s$, $d = 1,2,3,\ldots,D$, g = index of the best particle in the swarm, where

N_s = swarm size

D = dimension of the problem

w = inertia weight

c_1 and c_2 = learning factors

r_1 and r_2 = random numbers in the range [0, 1]

$x_{id} = \left[x_{i1}, x_{i2}, \ldots, x_{iD}\right]$, the position of the i^{th} particle

$v_{id} = \left[v_{i1}, v_{i2}, \ldots, v_{iD}\right]$, the velocity of the i^{th} particle

$p_{id} = \left[p_{i1}, p_{i2}, \ldots, p_{iD}\right]$, the best position of the i^{th} particle

$p_{gd} = \left[p_{g1}, p_{g2}, \ldots, p_{gD}\right]$, the best position in the swarm

Equations (2) and (3) given above depict the flying path of the particle population. The new velocity of the particle is computed using equation (2), taking into account its former velocity as well as the distances of its existing position from the best experience of its own and the best experience of the neighbourhood. Next, following equation (3), each particle will move towards its new position. The performance of every particle is evaluated in accordance with a pre-determined fitness function that is linked to the problem in question. The general pseudo-code of the PSO is presented as follows:

1: Initialize swarm positions and velocities
2: while stopping criterion not attained
3: for each particle
4: Calculate velocity according to equation (2)
5: Update position according to equation (3)
6: Evaluate fitness value
7: Update best position if the current fitness value is better
8: end for
9: Update global best position having the best fitness value of the swarm
10: end while

The general MOPSO algorithm can be represented by the pseudocode below. The differences between single-objective and multi-objective optimization are denoted in italics.

1: Initialize swarm positions and velocities
2: *Initialize external archive*
3: while stopping criterion not attained
4: for each particle
5: *Select a member of the external archive*
6: Calculate velocity according to equation (2)
7: Update position according to equation (3)
8: Evaluate fitness value
9: Update best position if the current fitness value is better
10: end for
11: *Update members in the external archive*
12: end while

The use of an external archive, the member selection from the archive, along with the archive update, establish the major concepts in the enhancement of MOPSO techniques, though not the only ones [36]. In MOPSO, the velocity and position update equations are still similar to equations (2) and (3) in PSO; whereas the objective function now consists of multiple objectives as formulated in equation (1).

There are several MOPSO variants proposed in the literature. Sha and Lin [37] altered the representation of velocity, movement and position of particles. Unlike the standard PSO that retains the best positions discovered as yet, it preserved the best schedules produced up till now. Consequently, the particle movement was also modified based on a swap operator. For the particle velocity, instead of moving them toward the best solutions, it focused on avoiding the particles from getting trapped in local optima. Wisittipanich and Kachitvichyanukul [38] adopted a combination of four groups of particles within a single swarm with unique movement schemes. The first group of particles conducted their explorations according to their personal experiences, while the second group of particles employed the global knowledge and were directed to the sparse regions of the Pareto front. The third group of particles were assigned to fulfill the gaps among Pareto front, while the fourth group of particles sought to explore around the boundary of the non-dominated front. Meng et al. [39] proposed a structure of dual-population i.e., a searching-population and a leading-population. The searching population is assigned the role to search while the leading-population is assigned the role to guide the searching-population to the Pareto front.

IV. Job-Shop Scheduling Problems (JSP)

In general, the JSP is designated as having a collection of jobs to be scheduled on a collection of machines in a given order [40]. Each job contains a number of operations, where the operation denotes the processing of a job on a particular

machine. Every operation has a duration or processing time, which is known in advance, specific for every machine. Every operation is arranged to be scheduled on a specified machine adhering to a predetermined sequence, known as the precedence constraints, in which the sequence of machines is distinct for every job [41]. These precedence constraints, which dictate the specific order of operations, impose some complexity to the JSP. The assignment of operations of a job for predefined processing times on a machine is known as a schedule. The JSP is also subject to the following constraints [42]: Each job must be executed exactly once, without recirculation, on every machine; each machine can only process one operation at a time; each job can only pass through the machine in a certain order; no pre-emption is allowed, or once a job has started processing, it cannot be interrupted. When the schedule manages to determine the best sequence of operations processed on all machines in optimizing particular objectives, it is regarded as a good or optimal schedule.

In terms of objective functions, the majority of the works in solving job-shop problems has focused on minimizing the makespan, where the formula is given by equation (4):

$$\text{Makespan, } C_{\max} = \max\{C_i\} \tag{4}$$

where C_i is the completion time of job *i*.

In order to reflect real-world scheduling problems, more objective functions are considered, such as to minimize the total tardiness, in which the formula is given by equation (5):

$$\text{Total tardiness, } T_{tot} = \sum T_i \tag{5}$$

$$T_i = \max(C_i - d_i, 0) \tag{6}$$

where T_i in equation (6) is the tardiness of job *i*, i.e. when job *i* is completed after its due date, d_i.

Other objective functions that can be employed include total machine idle time [37], mean weighted completion time [43], sum of weighted tardiness and earliness costs [43], total penalties of tardiness and earliness [44], weighted mean flow time [44] and many more. Oyetunji [45] presented diverse scheduling objectives along with the formulation of their mathematical expressions.

V. Applications of MOPSO in Job-Shop Scheduling Problems

Tables 1 and 2 outline the applications of MOPSO in JSP in terms of types of objective functions, types of test problems and methods of multi-objective (MO) optimization. The tables also summarize the works in the aspect of their strategies and performances.

No.	Refs	Objective Functions	Test Problems	MO Optimization Methods	Strategies	Performances	Remarks
1	[46]	1. Makespan 2. Total tardiness	OR-Library benchmark problems	Pareto	1. Combined global best position selection with crowding measure-based archive maintenance. 2. Performed mutation on archive members.	1. Outperformed strength Pareto evolutionary algorithm2 (SPEA2) in 13 out of 18 problems. 2. Outperformed MOPSO in 17 out of 18 problems.	1. Combination procedure resulted in fast approximation of high-quality Pareto optimal front. 2. Mutation of chosen archive members reduced stagnation of search process to produce more non-dominated solutions.
2	[37]	1. Makespan 2. Total machine idle time 3. Total tardiness	OR-Library benchmark problems	Pareto	1. Modified representation of position, movement, and velocity of particles. 2. Performed diversification strategy to update non-dominated solutions.	1. Outperformed multi-objective Genetic Algorithm (MOGA) in relative error of solution for makespan and total idle time in all 23 problems. 2. Outperformed MOGA in relative error of solution for total tardiness in 22 out of 23 problems.	1. Best schedules were recorded rather than best positions found so far. 2. Particle movement was based on swap operator. 3. Particle velocity focused on avoiding particles from getting trapped in local optima rather than moving them toward best solutions.
3	[47]	1. Makespan 2. Workload equitableness among machines 3. Total	OR-Library benchmark problems	Pareto	Performed orthogonal design method during generation of initial swarm and selection of global best position.	Outperformed MOGA, Non-dominated Sorting GA II (NSGA-II) and Variable Neighborhood Particle Swarm Optimization (VNPSO) by dominating solutions found by those	1. Orthogonal operator enhanced PSO searching capability by distributing the swarm evenly during initial stage and leading it toward Pareto Front dispersedly. 2. Became more

		maintaining cost of all jobs 4. Total compensation of delayed jobs			algorithms.	prevalence over other algorithms as the scale of chosen instances increased.
4	[38]	1. Makespan 2. Total tardiness	OR-Library benchmark problems	Pareto	1. Employed a combination of four groups of particles within a single swarm with unique movement schemes. 2. Retained updated non-dominated solutions discovered by entire swarm as a common elite group, which used as guidance for flying of particles.	1. Outperformed crowding measure-based multi-objective evolutionary algorithm (CMOEA) and strength Pareto evolutionary algorithm (SPEA) in producing non-dominated solutions in 7 out of 15 instances and most cases under ***C*** metric. 2. SPEA and CMOEA were marginally better in producing non-dominated solutions in 4 out of 15 instances and 2 and 3 instances respectively under ***C*** metric.	1. Each group of particles performed unique movement schemes with its own benefit to explore diverse potential areas. 2. Search procedure was faster and quality of solutions was higher due to the use of common elite group that assisted particles in using global information.
5	[48]	1. Makespan 2. Total weighted earliness 3. Total weighted tardiness	OR-Library benchmark problems	Weighted aggregative function	1. Performed evolutionary process in 2 stages. 2. Utilized multiple populations for independent evolution. 3. Employed migration strategies, re-initialization and local search.	1. Comparable to two-stage genetic algorithm (2ST-GA) and multi-stage genetic algorithm (MS-GA) in medium-sized problems. 2. Outperformed 2ST-GA and MS-GA in computational time and solution quality for large-sized problems.	1. Serial particle swarm with migrated particles accelerated convergence of solution through shared information of search experience from previous swarm. 2. Local search enhanced quality of solution by exploring better solutions around its neighbours. 3. Re-initialization strategy to diversify particles periodically was able to prevent the tendency of being stuck in local minima.
6	[39]	1. Makespan 2. Average flow time 3. Machine idle time	OR-Library benchmark problems	Greedy strategy	1. Employed dual-population hybrid PSO algorithm based on greedy strategy, where one population will lead another population to converge. 2. Performed mutation and crossover operations on individuals in the two populations. 3. Utilized simulated annealing as local search.	Outperformed MOPSO and Improved Multi-objective Particle Swarm Optical (IMPSO) in terms of solution quality and running time in all 31 problems of different scales.	1. Evolutionary speed was faster in early stage of searching. 2. Crossover and mutation operators enhanced diversity and convergence of both populations. 3. Local search strategy was effective to escape local minima and to guide searching process converging to optimal values.

Table 1. Comparison of standard JSP solved by MOPSO.

No.	Refs	Objective Functions	Test Problems	MO Optimization Methods	Scheduling Constraints	Strategies	Performances	Remarks
1	[49]	1. Mean fuzzy completion time 2. Maximum fuzzy completion time 3. Minimum agreement index	Selected problems from literature	Pareto	1. Fuzzy due date 2. Fuzzy processing time	1. Combined crowding measure-based archive maintenance with global best position selection. 2. Performed mutation on archive members.	1. Outperformed Pareto-dominance MOPSO (PDMOPSO) in 6 out of 8 problems. 2. Outperformed SPEA2 in 3 out of 8 problems and 4 out of 8 problems involving mutation. 3. Consumed less computational time than PDMOPSO and SPEA2.	1. Combination procedure resulted in archive members became uniformly distributed, each member led global best of at least 1 particle to take part in new search. 2. Introduction of mutation resulted in strong optimization capability in fuzzy JSP.
2	[43]	1. Sum of weighted tardiness and earliness costs 2. Mean weighted completion time	Self-designed test problems	Pareto	Sequence-dependent setup times	1. Employed genetic operators for particle update and VNS for particle improvement. 2. Combined crowding measure-based archive updating method with global best position selection. 3. Constructed initial solutions using new ETS method.	1. Obtained more non-dominated solutions with greater quality than NSGA-II. 2. Non-dominated solutions were more uniformly distributed than NSGA-II. 3. Outperformed NSGA-II in average values of diversification metric. 4. Outperformed NSGA-II in mean improvements of about 8% and 13% for 2 objectives.	1. Utilized 3 comparison metrics: diversity metric, spacing metric, and quality metric. 2. Found non-dominated solutions with higher diversity. 3. More time-consuming than NSGA-II.
3	[44]	1. Total penalties of earliness and tardiness 2. Weighted mean flow time	Self-designed test problems	Pareto	1. Sequence-dependent setup times 2. Ready times	1. Employed genetic operators for particle update and variable neighbourhood search (VNS) for particle improvement. 2. Applied characters of scatter search (SS) to choose a different swarm in each iteration. 3. Constructed initial solutions using new elite tabu search (ETS) method.	1. Outperformed SPEA-II and NSGA-II in every test instance. 2. Obtained more non-dominated solutions with greater quality than SPEA-II and NSGA-II. 3. Outperformed SPEA-II and NSGA-II in diversification metric.	1. Utilized 3 comparison metrics: diversity metric, spacing metric and quality metric. 2. Found non-dominated solutions with higher diversity. 3. More time-consuming, with greater increasing rate in computational times.

Table 2. Comparison of JSP with scheduling constraints solved by MOPSO.

Table 1 presents the related works implementing standard JSP which do not consider additional scheduling constraints such as sequence-dependent setup times, ready times, as well as uncertainty in due dates and processing times. Other than the standard JSP as portrayed in Table 1, there are also several variants of JSP solved by MOPSO. In the literature, some of the articles implemented JSP with scheduling constraints such as sequence-dependent setup times, ready times, as well as fuzzy due dates and fuzzy processing times. It may be more appropriate to take into account these considerations in order to reflect the real-life situations, as can be found in the related works presented in Table 2.

Based on the works described in Tables 1 and 2, MOPSO implementations for solving JSP have a number of similarities and differences from each other and this is discussed further in the next section.

VI. Discussion and Future Research Directions

In this section, a discussion on variations and improvements of MOPSO implementation in JSP is conducted. Based on Tables 1 and 2, there exists a variation of MOPSO in solving JSP, where certain preferences are more widely employed compared to the others, as described below:

- Objective functions: The standard objective function employed is makespan, followed by total tardiness and total machine idle time. Other than that, objective functions in terms of flow times, due dates and penalties/costs are typical choices too.
- Test problems: The standard test problems used is from the OR-Library. Few researchers use self-designed test problems and selected problems from the literature.
- MO optimization methods: The most popular method in solving the multi-objective cases for JSP is according to the concept of Pareto optimality, as compared to the other non-Pareto scheme that includes aggregation strategy and lexicographic ordering strategy.
- Scheduling constraints: A majority of MOPSO does not apply additional scheduling constraints to their algorithms to solve JSP. Nevertheless, the introduction of fuzzy variables into parameters such as due dates and processing times, as well as practical considerations such as sequence-dependent setup times and ready times, are other variations in MOPSO implementation to solve JSP.

Table 3 highlights MOPSO improvements and the aims of their implementation in solving JSP based on Tables 1 and 2. We can see from Table 3 that different improvement strategies can be adopted simultaneously in MOPSO algorithm for JSP. For instance, three improvement strategies were used by Pratchayaborirak and Kachitvichyanukul [48] to solve JSP: 1) a local search procedure is adopted to enhance exploitation ability and solution quality; 2) multiple populations are used to accelerate the convergence of solution through shared information of search experience; 3) a re-initialization strategy is employed to diversify particles periodically to avoid being trapped in local optima. The combination of these three improvement strategies resulted in their MOPSO algorithm for JSP has three advantages of better local exploitation capability, faster convergence speed and avoiding local optima.

No.	Improvements	Aims	Refs.
1	Combined global best position selection with crowding measure-based archive maintenance	Preserve solutions of the closest distance to the Pareto front with better diversity.	[43] [46] [49]
2	Genetic operators	Reduce stagnation of search process, enhance convergence and diversity.	[39] [43] [44] [46] [49]
3	Diversification procedure in maintenance of Pareto optima	Maintain and update non-dominated solution set to ensure diversity.	[37]
4	Orthogonal design method	Improve the widespread searching capability.	[47]
5	Local search procedures	Enhance exploitation ability and solution quality.	[48]
6	Particle movement strategies	Explore different potential areas, generate well-spread and better quality of Pareto front.	[38]
7	Multiple populations	Accelerate convergence of solution through shared information of search experience.	[39] [48]
8	Re-initialization strategy	Diversify particles periodically to avoid being trapped in local optima.	[48]
9	Hybrid with other metaheuristics: simulated annealing, variable neighbourhood search, scatter search, tabu search	- Escape local optima and guide searching process converging to optimal values. - Improve particles utilizing different neighbourhood search structures. - Assemble a new swarm from superior and diverse solutions in Pareto archive and new particles. - Construct a group of superior and diverse initial solutions.	[39] [43] [44]

Table 3. List of MOPSO improvements applied in JSP.

A hybrid of MOPSO with other metaheuristics is also

preferable because a combination of two or more algorithms facilitates the search process to be conducted more comprehensively. For instance, a hybrid with another metaheuristic such as Simulated Annealing proposed by Meng et al. [39] in solving JSP offers collective advantages of each individual algorithm, where MOPSO offers an efficient global search ability and a good convergence rate while Simulated Annealing offers a better local exploitation ability and an ability to escape local optima.

Certainly, these improvement strategies increase the computational cost for incorporating the additional components into MOPSO. For instance, as demonstrated by Tavakkoli-Moghaddam et al. in [43] and [44], their algorithms implemented multiple improvement strategies and had better performance in solving JSP compared to the competing algorithms, even though they took more time to achieve it. Therefore, it is generally suitable in the context that a fast computational speed is not necessary but the quality of the computational result is strongly demanded.

Based on the summary in Table 3, we consolidate these aspects of improvement strategies in the form of a proposed MOPSO model in solving JSP, as shown in Figure 1. The model illustrates the context of integrating the strategies for improvements within two phases of MOPSO, i.e. initialization and swarm-evolving phases. It also illustrates the multiple aspects of improvements that can be carried out during the swarm-evolving phase, i.e. in terms of solution diversity, exploitation (nearby) and exploration (wide-ranging) mechanisms, as well as premature convergence (convergence to a local optimum). 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. Besides, it is noted that there are strategies that incorporate a hybrid with other metaheuristics. In essence, this model represents the basis of our perspective on improvement strategies available in the literature.

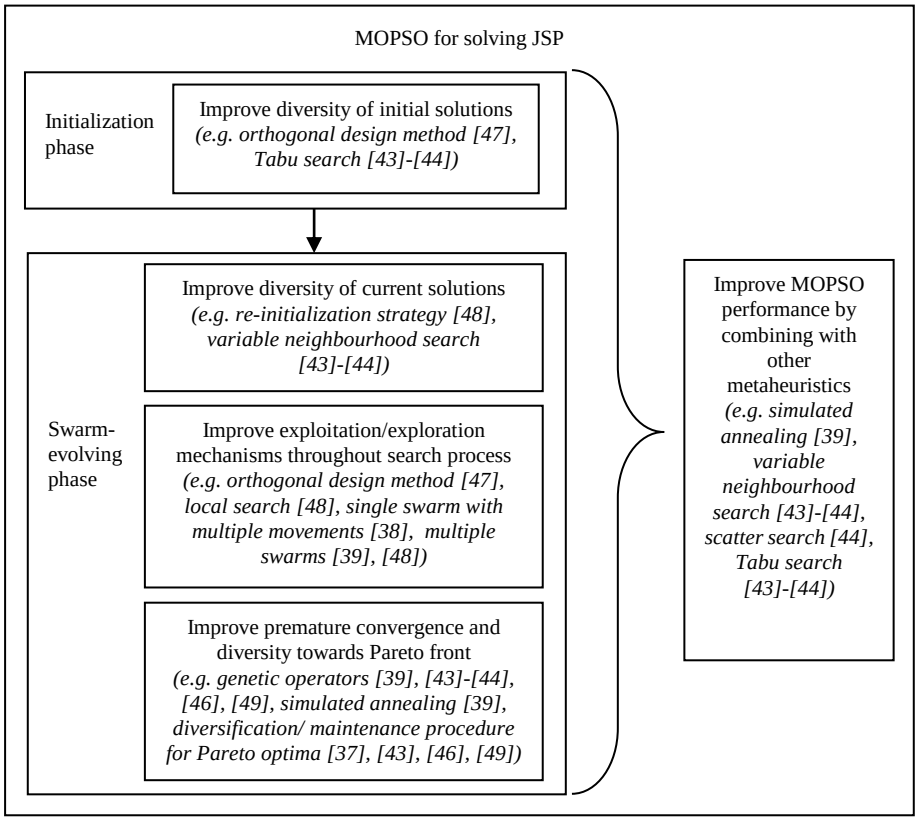

Figure 1. Proposed MOPSO model to solve JSP

Based on the proposed MOPSO model in Figure 1, we offer some insights regarding aspects of improvements that are merit exploring in the near future:

- Improvements in the diversity of the solutions, which are not only targeted to the non-dominated solutions on the Pareto front, but also include the diversity of the initial solutions and solutions of the swarm in general.
- Improvements in the exploitation (nearby) and exploration (wide-ranging) abilities throughout the search process, for example, using a single swarm with multiple movements or using multiple swarms.
- Improvements in the premature convergence (convergence to a local optimum), for example, using genetic operators, local search and re-initialization strategy.
- Improvements involving multiple aspects or strategies, for example, using a hybrid of MOPSO with other metaheuristics.

It is clear that MOPSO improvements generally requires additional computational cost. Thus, the trade-off between the computational cost and quality of solutions should be considered based on application requirements in JSP. This is imperative for practical purposes since the MOPSO algorithm may be implemented in real-life scheduling that requires decision-making to be carried out as soon as possible. Despite the ability of the technique to achieve better computational results, relatively long computation time will deem it inefficient in solving real-world scheduling problems.

VII. Conclusion

In this paper, the discussion of MOPSO and its application of solving JSP is carried out in detail. We first described the basic concept of multi-objective optimization and its general categorization, followed by the descriptions and algorithms of the standard PSO and multi-objective PSO. We also provided the background of JSP, along with the related objective functions. Afterwards, the MOPSO applications in solving JSP were presented, succeeded by the summary on variations and improvements of MOPSO implementation in JSP. Finally, we put forward a proposed MOPSO model to solve JSP and the particular aspects of MOPSO improvements that could contribute to the future directions of research in this area.

Based on the computational results of the test instances in existing woks, MOPSO exhibits superior performance over the other algorithms in solving JSP. It offers efficient optimization of complex multi-dimensional search spaces, flexible to various hybridization and integration with other useful techniques. However, despite showing promising results, the applications of MOPSO to JSP are still very limited. There has been a deficiency in the research work done and the development of MOPSO approach for JSP remains open, though not very active, research area. Further studies of developing effective MOPSO algorithm are required, and its challenges in employing JSP with multiple objectives need to be further investigated.

Acknowledgment

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] P. Pongchairerks, V. Kachitvichyanukul. “A Particle Swarm Optimization Algorithm on Job-Shop Scheduling Problems with Multi-Purpose Machines”, *Asia-Pacific Journal of Operational Research (APJOR)*, 26(2), pp. 161–184, 2009.

[2] A. S. Jain, S. Meeran. “Deterministic job-shop scheduling: Past, present and future”, *European Journal of Operational Research*, 113(2), pp. 390–434, 1999.

[3] P. Brucker. *Scheduling Algorithms*, 5th ed. Springer-Verlag Berlin Heidelberg, 2007.

[4] J. E. Beasley. “OR-Library: Distributing Test Problems by Electronic Mail”, *The Journal of the Operational Research Society*, 41(11), pp. 1069–1072, 1990.

[5] J. Carlier, E. Pinson. “An algorithm for solving the job-shop problem”, *Management Science*, 35(2), pp. 164–176, 1989.

[6] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*, 5th ed. Springer International Publishing, 2016.

[7] C. Rajendran. “Heuristics for scheduling in flowshop with multiple objectives”, *European Journal of Operational Research*, 82(3), pp. 540–555, 1995.

[8] B. Yagmahan, M. M. Yenisey. “A multi-objective ant colony system algorithm for flow shop scheduling problem”, *Expert Systems with Applications*, 37(2), pp. 1361–1368, 2010.

[9] R. T. Marler, J. S. Arora. “Survey of multi-objective optimization methods for engineering”, *Structural and Multidisciplinary Optimization*, 26(6), pp. 369–395, 2004.

[10] H. Afaq, S. Saini. “Swarm Intelligence based Soft Computing Techniques for the Solutions to Multiobjective Optimization Problems”, *International Journal of Computer Science Issues*, 8(3), pp. 498–510, 2011.

[11] P. Ngatchou, A. Zarei, M. A. El-Sharkawi. “Pareto Multi Objective Optimization”. In *Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems*, pp. 84–91, 2005.

[12] Y. Zhang, S. Wang, G. Ji. “A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications”, *Mathematical Problems in Engineering*, 2015, pp. 1–38, 2015.

[13] A. Banks, J. Vincent, C. Anyakoha. “A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications”, *Natural Computing*, 7(1), pp. 109–124, 2008.

[14] J. Moore, R. Chapman. “Application of particle swarm to multiobjective optimization”. Department of Computer Science and Software Engineering, Auburn University, 1999.

[15] C. A. Coello Coello, M. S. Lechuga. "MOPSO : A Proposal for Multiple Objective Particle Swarm Optimization". In *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 1051–1056, 2002.

[16] C. A. Coello Coello, G. T. Pulido, M. S. Lechuga. "Handling multiple objectives with particle swarm optimization", *IEEE Transactions on Evolutionary Computation*, 8(3), pp. 256–279, 2004.

[17] X. Hu, R. C. Eberhart. "Multiobjective optimization using dynamic neighborhood Particle Swarm Optimization". In *Proceedings of the IEEE congress on evolutionary computation (CEC 2002)*, pp. 1677–1681, 2002.

[18] X. Hu, R. C. Eberhart, Y. Shi. "Particle swarm with extended memory for multiobjective optimization". In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 193–197, 2003.

[19] K. E. Parsopoulos, M. N. Vrahatis. "Particle swarm optimization method in multiobjective problems". In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, pp. 603–607, 2002.

[20] K. E. Parsopoulos, M. N. Vrahatis. "Recent approaches to global optimization problems through Particle Swarm Optimization", *Natural Computing*, 1(2–3), pp. 235–306, 2002.

[21] J. D. Schaffer. "Multiple objective optimization with vector evaluated genetic algorithms". In *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pp. 93–100, 1985.

[22] K. E. Parsopoulos, D. K. Tasoulis, M. N. Vrahatis. "Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization". In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, pp. 823–828, 2004.

[23] J. E. Fieldsend, S. Singh. "A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence". In *Proceedings of UK Workshop on Computational Intelligence (UKCI'02)*, pp. 37–44, 2002.

[24] S. Lalwani, S. Singhal, R. Kumar, N. Gupta. "A comprehensive survey: applications of multi-objective particle swarm optimization (MOPSO) algorithm", *Transactions on Combinatorics*, 2(1), pp. 39–101, 2013.

[25] M. Reyes-Sierra, C. A. Coello Coello. "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art", *International Journal of Computational Intelligence Research*, 2(3), pp. 287–308, 2006.

[26] K. E. Parsopoulos, M. N. Vrahatis. "Multi-Objective Particles Swarm Optimization Approaches", in *Multi-Objective Optimization in Computational Intelligence*, pp. 20–42, 2008.

[27] J. E. Fieldsend. "Multi-objective particle swarm optimisation methods". Department of Computer Science, University of Exeter, 2004.

[28] M.-P. Song, G.-C. Gu. "Research on particle swarm optimization: a review". In *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, pp. 2236–2241, 2004.

[29] T. Li, B. Yang. "A Review of Multi-objective Particle Swarm Optimization Algorithms in Power System Economic Dispatch", *International Journal of Simulation -- Systems, Science & Technology*, 17(27), pp. 1–5, 2016.

[30] E. Zitzler, L. Thiele. "Multiobjective Evolutionary Algorithms - A Comparative Case Study and the Strength Pareto Approach", *IEEE Transactions on Evolutionary Computation*, 3(4), pp. 257–271, 1999.

[31] E. Zitzler, M. Laumanns, S. Bleuler. "A Tutorial on Evolutionary Multiobjective Optimization", in *Metaheuristics for Multiobjective Optimisation*, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, (eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 3–37, 2004.

[32] N. Gunantara. "A review of multi-objective optimization: Methods and its applications", *Cogent Engineering*, 5(1), pp. 1–16, 2018.

[33] U. Baumgartner, C. Magele, W. Renhart. "Pareto optimality and particle swarm optimization", *IEEE Transactions on Magnetics*, 40(2), pp. 1172–1175, 2004.

[34] C. Chi-kin, T. Hung-tat. "Autonomous agent response learning by a multi-species particle swarm optimization". In *Proceedings of the 2004 Congress on Evolutionary Computation*, pp. 778–785, 2004.

[35] J. Kennedy, R. Eberhart. "Particle swarm optimization". In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[36] N. Padhye, J. Branke, S. Mostaghim. "Empirical comparison of MOPSO methods-guide selection and diversity preservation". In *2009 IEEE Congress on Evolutionary Computation*, pp. 2516–2523, 2009.

[37] D. Y. Sha, H. H. Lin. "A multi-objective PSO for job-shop scheduling problems", *Expert Systems with Applications*, 37(2), pp. 1065–1070, 2010.

[38] W. Wisittipanich, V. Kachitvichyanukul. "An Efficient PSO Algorithm for Finding Pareto-Frontier in Multi-Objective Job Shop Scheduling Problems", *Industrial Engineering and Management Systems*, 12(2), pp. 151–160, 2013.

[39] Q. Meng, L. Zhang, Y. Fan. "Research on Multi-objective Job Shop Scheduling with Dual Particle Swarm Algorithm Based on Greedy Strategy", *Wireless Personal Communications*, 103(1), pp. 255–274, 2018.

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

[41] Y. Chen, Z. Guan, X. Shao. "A comparative analysis of job scheduling algorithm". In *2011 International Conference on Management Science and Industrial Engineering*, pp. 1091–1095, 2011.

[42] J. Błażewicz, W. Domschke, E. Pesch. "The job shop scheduling problem: Conventional and new solution techniques", *European Journal of Operational Research*, 93(1), pp. 1–33, 1996.

[43] R. Tavakkoli-Moghaddam, M. Azarkish, 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, 53(5–8), pp. 733–750, 2011.

[44] R. Tavakkoli-Moghaddam, M. Azarkish, A. Sadeghnejad-Barkousaraie. "A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem", *Expert Systems with Applications*, 38(9), pp. 10812–10821, 2011.

[45] E. O. Oyetunji. "Some common performance measures in scheduling problems: Review article", *Research Journal of Applied Sciences, Engineering and Technology*, 1(2), pp. 6–9, 2009.

[46] D. Lei. "A Pareto archive particle swarm optimization for multi-objective job shop scheduling", *Computers and Industrial Engineering*, 54(4), pp. 960–971, 2008.

[47] M. Feng *et al.* "Orthogonal particle swarm optimization for multi-objective job shop scheduling problems". In *2010 2nd International Conference on Computational Intelligence and Natural Computing*, pp. 256–260, 2010.

[48] T. Pratchayaborirak, V. Kachitvichyanukul. "A two-stage PSO algorithm for job shop scheduling problem", *International Journal of Management Science and Engineering Management*, 6(2), pp. 83–92, 2011.

[49] D. Lei. "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems", *International Journal of Advanced Manufacturing Technology*, 37(1–2), pp. 157–165, 2008.

Author Biographies

Nurul Izah Anuar received her Bachelor of Electronic Engineering (Computer) and Master of Engineering (Advanced Manufacturing Management) degrees from Multimedia University (MMU), Malaysia in 2006 and 2011, respectively, where she is currently a senior lecturer in the university. She is currently pursuing a PhD programme in Universiti Teknikal Malaysia Melaka (UTeM), Malaysia. Her research interests include exploring the application of computational intelligence in production scheduling problems.

Muhammad Hafidz Fazli bin Md Fauadi is an associate professor in the Universiti Teknikal Malaysia Melaka (UTeM), Malaysia. He obtained the Bachelor degree in Information Technology from Universiti Kebangsaan Malaysia (UKM) in 2004 and Master degree in Mechanical Engineering (Advanced Manufacturing Technology) from Universiti Teknologi Malaysia (UTM) in 2006. Then, he received the Doctor of Engineering (Information, Production and System Engineering) degree from Waseda University, Japan in 2012. His specializations are in intelligent manufacturing system, cloud manufacturing and Internet of Things.