A Hybrid Differential Particle Swarm Optimization Approach to Solve a Multi-objective Parallel Machine Scheduling Problem

Payam Chiniforooshan, Shahrooz Shahparvari

Abstract - This paper addresses the unrelated parallel machine scheduling problem with sequence-dependant setup times for minimizing total tardiness and workload imbalance. The mixed integer linear programming is proposed to model the studied problem. This problem is shown to be NP-hard in the strong sense. Thus, we propose a hybrid algorithm that combines differential evolution with particle swarm optimization, namely HDEPSO, in order to solve the given problem. Our objective is to achieve faster convergence rate and obtain better pareto optimal solutions. In order to demonstrate the efficiency and reliability of the proposed algorithm, a number of test problems are solved. The HDEPSO results are compared with two well-known multi-objective genetic algorithms in the literature, i.e. NSGA-II and SPEA-II based on some comparison metrics. Computational experiments indicate the superiority of the HDEPSO compared to these two genetic algorithms.

Keywords: Parallel Machine Scheduling, Multi-objective optimization, Differential evolution, Particle swarm optimization.

1. Introduction

Scheduling is concerned primarily with allocating scarce resources, e.g. machines, to particular tasks that have to be done over given periods, with the objective of optimizing some performance measures [1]. Scheduling problems are used in many applications such as production and procurement, transportation and distribution, and communication. In real production, parallel machine scheduling (PMS) is one of the most basic and important types of the scheduling problems. A lot of research has been made on parallel machines, independent of the machine environment, i.e. on identical machines, unrelated, etc.

Manuscript received June 17, 2012

Chiniforooshan. P. is with the Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, (e-mail: .Chiniforooshan@usc.ac.ir).
Shahparvari. Sh is with Department of Industrial Engineering, Kish International Campus, University of Tehran, Tehran, Iran (corresponding author to provide phone: Tel: +989122392343,Fax:+982166517077; e-mail: Shahparvari@ut.ac.ir).

In a general PMS problem, a set of *n* independent jobs are to be scheduled on *m* parallel machines to meet certain job completion time related objectives [2]. A well respected paper which is presented by Cheng et al. (1990) relates to the whole class of scheduling problem that consists of parallel machines in different environment. In unrelated parallel machines, there is no particular relationship among machines' capabilities. In other words, the processing time of each job differs in terms of processing on each machine. Therefore, the unrelated parallel machine scheduling (UPMS) becomes the most complex problem and has attracted more researchers [3,4,5,6].

During many years, it was common practice to take only one performance criterion into account in the objective function. However, scheduling problems in real life applications generally involve optimization of more than one criterion. Although past literature in UPMS addresses a variety of research papers, all of them have typically considered a single criterion. UPMS often involve more than one aspect and require multi-objective analysis. Therefore, it is more amenable to solving multi-objective scheduling problem.

Minimizing total tardiness is one of the most important criteria in many manufacturing systems, especially in the current situation where competition is becoming more intensive. Pfund et al. (2004) [7] surveyed the literature in solving UPMS involving single and multiple objectives. According to this research, while minimization of makespan has been fairly widely studied, problems of minimizing the number of tardy jobs, weighted number of tardy jobs, total tardiness, and total weighted tardiness remain largely unstudied.

Another interesting objective function is to balance the workload across machines as evenly as possible. In PMS environments, by distributing the available workload among the available machines as equally as possible, bottlenecks can be eliminated, throughput can be maximized, and work in progress, finished goods inventory and operating costs can be lowered. Rajakumar et al. (2004) addressed the UPMS problem without setup times and with the objective of load balancing [8]. Yu et al. (2002) studied UPMS problem in a printed wiring board manufacturing line for a set of performance measures included load balancing. They used integer programming, lagrangian relaxation, and heuristics to solve the problem [9]. Sun et al. proposed a genetic algorithm for UPMS to minimize number of pickups and workload imbalance [10]. Keskinturk et al. (2010) proposed a mathematical model for the PMS problem to minimize average relative percentage of imbalance, taking into account sequence-dependant set up times [11].

The studies with bi-criteria goals attempt to consider the coordination of the producer and customers [12]. These criteria are often conflicting in nature and are quit complex. In this research, we address the bi-objective unrelated parallel machine scheduling with sequenced-dependant setup times subject to minimization of the total weighted tardiness and machines workload imbalances. Minimizing both objectives can help assuring a high utilization of the production system as well as a high level of service towards the costumers. Scheduling problems with sequence-dependant setup times has received the attention of many researchers.

Lee et al. (1997) [13] considered PMS with sequence-dependent setup time that minimizes total weighted tardiness. Another paper on UPMS with setup times is studied by Kim et al. (2002). The paper deals with real problem encountered in a semiconductor production facility where a part of the manufacturing process suffers from bottleneck. As criterion, minimization of total tardiness is used [14]. An interesting paper, which considers setup times, is recently presented by Allahveredi et al. (2008) [15], where all classes of problems are mentioned and the techniques used to solve them.

Lenstra et al. [16] showed that a single machine scheduling with total weighted tardiness minimization problem is NP-hard in strong sense. Clearly, the single machine is a special case of the sequence-dependent unrelated parallel machine scheduling problem considered in this paper. Therefore, the problem investigated in this paper is also strongly NP-hard. We refer to Logendran et al. [3] for further knowledge and recent findings regarding the UPMS problem. Thus, meta-heuristic algorithms, such as genetic algorithm (GA) [17], tabu search (TS) [18], ant colony (ACO) [19], simulated annealing (SA) [20], and variable neighborhood search (VNS) [21], are extensively used to find a good solution for such a hard problem.

Differential evolution (DE) is a novel evolutionary algorithm recently introduced by Storn and Price [22] for optimization problems over continuous spaces. Due to its ease of use, fast convergence and robustness, DE has successfully been applied to diverse domain of science and engineering [23]. Besides other applications, all of the DE implementations in scheduling literature focused on the single-machine scheduling problem [24], flow shop scheduling problem [25] and job shop scheduling problem [26]. Particle swarm optimization (PSO), first introduced by Kennedy and Eberhart [27], is an another most recent evolutionary algorithms that has been applied to wide range of applications such as power and voltage control [28], task assignment [29], project scheduling [30], cell formation problem [31], flow shop sequencing problem [32].

In this paper, we devise a hybrid approach by combining the searching ability of DE and PSO, namely HDEPSO, to solve multi-objective unrelated parallel machine scheduling with sequence-dependant setup time. This hybridization enhances the exploration ability of the DE by the vibrancy and explorative nature of PSO. The optimization criteria are minimizing total weighted tardiness and workload imbalance. To investigate the effectiveness of our proposed approach, computational experiments are conducted and comparison results with two well-known multi-objective genetic algorithms, namely NSGAII [33] and SPEA-II [34], are provided. The results clearly show that our HDEPSO significantly outperforms the abovementioned algorithms.

The remaining contents of this paper are partitioned into five sections. In section 2, we propose the formulation of multi-objective unrelated PMS problem. This is followed by a brief overview of DE and PSO in the section 3. In section 4, we describe the proposed HDEPSO algorithm. In section 5, we show the experimental results obtained by the proposed solution algorithm and then compare these results with two multi-objective genetic algorithms, called NSGAII and SPEA-II. Finally, conclusion is presented in section 6.

2. Problem definition and mathematical modeling

As described earlier, our problem involves scheduling *n* jobs on *m* unrelated parallel machines to minimize total tardiness and workload imbalance as objective functions. A classical UPMS problem can be stated as follows: a set of *n* independent jobs to be processed on *m* available unrelated parallel machines. Each machine can process only one job at a specific time, and each job can be processed on only one machine. Two operations are not allowed to overlap each other. Each job is ready at the beginning of the scheduling horizon and has a distinct processing time and due date. Once started, an operation cannot be interrupted until it is completed. Each machine is continuously available. No idle time in the machines between operations is allowed. Also, we assume that there are sequence-dependent setup times. The notation for parameters and variables used in the model are as follows:

Parameters:

M	number of machines available
N	number of jobs to be scheduled
T_i	length of time job i is tardy
d_i	due date for job i
B	a large number
P_{im}	processing time for job i using machine m
S_{ji}	setup time for job i when it immediately follows job j
S_{0i}	setup time for job i when it is the first in the queue

Decision variables:

C_i	completion time of job i
Y_{ijm}	1, if job i is immediately followed by job j in sequence on machine m; 0, otherwise
Z_{im}	1, if job i is assigned to machine m; 0, otherwise
W_{min}	minimum workload of all machines
W_{max}	maximum workload of all machines

The following MILP model is proposed for our problem and is intended to minimize the ET objective function and the WB objective function with respect to some constraints.

$$Min\ Z_1 = \sum_{i=1}^{N} T_i \qquad (1)$$

$$Min\ Z_2 = W_{max} - W_{min} \qquad (2)$$

Subject to.

$$T_i \geq C_i - d_i \qquad i=1,\ldots,N \qquad (3)$$

$$E_i \leq d_i - C_i \qquad i=1,\ldots,N \qquad (4)$$

$$\sum_{m=1}^{M} Z_{im} = 1 \qquad i=1,\ldots,N \qquad (5)$$

$$\sum_{j=1}^{N} Y_{ijm} \leq Z_{im} \qquad i \neq j,\ i=0,1,\ldots,N,\ m=1,\ldots,M \qquad (6)$$

$$\sum_{j=0}^{N} Y_{ijm} = Z_{im} \qquad i \neq j,\ j=1,\ldots,N,\ m=1,\ldots,M \qquad (7)$$

$$C_i - C_j - BY_{ijm} \geq P_{im} + S_{ij} - B \qquad i \neq j,\ i=1,\ldots,N,\ j=0,1,\ldots,N,\ m=1,\ldots,M \qquad (8)$$

$$\sum_{i=1}^{N} P_{im} Z_{im} \geq W_{min} \qquad m=1,\ldots,M \qquad (9)$$

$$\sum_{i=1}^{N} P_{im} Z_{im} \leq W_{max} \qquad m=1,\ldots,M \qquad (10)$$

$$Y_{ijm}, Z_{im} \in \{0,1\} \qquad (11)$$

$$C_i, T_i, E_i, W_{min}, W_{max} \geq 0 \qquad (12)$$

The objective functions are (1) minimizing total tardiness and (2) minimizing total workload imbalance. A fictitious job *0* is introduced to simplify the understanding and the writing of the constraints, which obviously results in $Z_{0m} = 1$ and $C_0 = 0$. The defining constraints (3 and 4) measure the degree to which each job is tardy or early. Constraint (5) ensures that each job is processed on one and only one machine. Constraints (6) and (7) ensure that each job must come immediately before, and immediately after, only one other job. Constraint (8) ensures that the completion time of job i is far enough after that of job j to include the processing time and setup time for job i. Two defining constraints (9) and (10) measure the minimum and maximum workload of machines, respectively. Lastly, constraints (11) and (12) indicate the types of model's variables.

3. Overview of DE and PSO

Differential Evolution (DE) algorithm, introduced by Storn and Price (1995) is a powerful population-based evolutionary algorithm for optimization algorithm over continuous spaces. DE starts with an initial population vector, which is randomly generated when no preliminary knowledge about the solution space is available [35]. The basic scheme of DE, which is denoted as *DE/rand/1/bin*, can be summarized as follows:

At every generation G, DE maintains a population $\boldsymbol{P}^{(G)}$ of NP (population size) vectors of solutions which evolve through the optimization process to find global solution:

$$\boldsymbol{P}^{(G)} = [\boldsymbol{X}_1^G, \ldots, \boldsymbol{X}_{NP}^G] \quad (13)$$

The population size, NP, is constant during the optimization process. The dimension of each vector of candidate solutions correspond to the number of the decision parameters, D, to be optimized. Therefore,

$$\boldsymbol{X}_i^{(G)} = \left[X_{1,i}^{(G)}, \ldots, X_{D,i}^{(G)}\right], i = 1,2,\ldots,NP \quad (14)$$

After that the initial population is created, it evolves through the operation of mutation, crossover and selection. At every generation *G*, each vector in the population has to serve once a target vector. For each target vector, a mutant vector $\boldsymbol{V}_i^{(G)}$ is defined by:

$$\boldsymbol{V}_i^{(G)} = \boldsymbol{X}_a^{(G)} + F\left(\boldsymbol{X}_b^{(G)} - \boldsymbol{X}_c^{(G)}\right) \quad (15)$$

with random indexes $a, b, c \in \{1,2,\dots,Np\}$, integer, mutually different, and different to the target vector. *F* is a user defined constant (also known mutation scaling factor), which is typically chosen from the range (0,2] ([35]). Larger values for *F* result in higher diversity in the generated population and lower values cause faster convergence.

DE utilizes the crossover operation to generate new solutions by shuffling competing vectors and also to increase the diversity of the population. To this end, the trial vector, i.e., $\boldsymbol{U}_i^{(G)} = \left[U_{1,i}^{(G)}, \dots, U_{D,i}^{(G)}\right]$ is formed, where

$$U_{j,i}^{(G)} = \begin{cases} V_{j,i}^{(G)} & if\ rand_j \le C_R\ or\ j = k \\ X_{j,i}^{(G)} & Otherwise \end{cases},$$

$$i = 1,2,\dots,NP, k = 1,2,\dots,D \quad (16)$$

In (16), *rand_j* is the *j*th evaluation of a uniform random number generator with outcome between 0 and 1. *CR* is the crossover rate constant and is a user-defined parameter within the range [0,1]. Large *CR* usually increases the convergence rate. *K* is a random parameter index chosen from the set {1, …, *D*}, which is used to make sure that at least one parameter is always selected from a $\boldsymbol{V}_i^{(G)}$. The crossover procedure is illustrated in Figure 1.

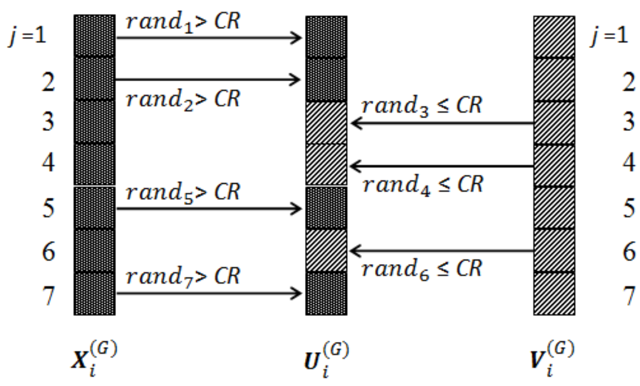

Figure 1 Crossover process with an example with 7 jobs (7-dimension)

In order to decide which vector $(\boldsymbol{U}_i^{(G)}, \boldsymbol{X}_i^{(G)})$ should become a member of generation *G*+1, the trial vector is compared to the target vector using a greedy criterion. For a minimization problem, the vector with the lower value of objective function is chosen. As a result, all individuals of the next generation are as good as or better than the individuals of the current generation. Comprehensive history and development of DE is presented in the[36]].

As other evolutionary algorithms, Particle swarm optimization (PSO) is a population-based optimization algorithm inspired by the behavior of a bird flock. The individuals in a PSO are denoted as particles. The PSO algorithm represents each potential solution by the position of a particle in multi-dimensional hyperspace. Throughout the optimization process velocity and displacement updates are applied to each particle to move it to a different position and thus a different solution in the search space. PSO refines its search by attracting the particles to positions with good solutions. PSO remembers the best position found by any particle (*gbest*). Also, each particle remembers its own previously best found position (*pbest*). Suppose that the position of the particle *i* at the *G*th iteration is represented by $\boldsymbol{X}_i^{(G)}$. Then, the velocity vector of particle *i* at iteration *G+1*

$(\boldsymbol{V}_i^{(G)})$ is updated by the Eq. (17).

$$\boldsymbol{V}_i^{(G+1)} = \omega \boldsymbol{V}_i^{(G)} + c_1 \times rand() \times \left(\boldsymbol{Pb}_i^G - \boldsymbol{X}_i^G\right) + c_2 \times rand() \times \left(\boldsymbol{Pg}^G - \boldsymbol{X}_i^G\right) \quad (17)$$

Where ω is the inertial weight which is introduced to balance between the global and local search abilities, $\boldsymbol{Pb}_i^G$ is the best found position of the *i*th particle at the *t*th iteration and $\boldsymbol{Pg}^G$ is the best position known for all particles. c_1 and c_2 are the cognitive and social acceleration constants, and $rand()$ is a random number generator with a uniform distribution over [0,1]. The position of each particle is updated in each iteration by adding the velocity vector to the position vector according to Eq. (18).

$$\boldsymbol{X}_i^{G+1} = \boldsymbol{X}_i^G + \boldsymbol{V}_i^{(G+1)} \quad (18)$$

This simultaneous movement of particles towards their own previous best solutions and the best solution found by the entire swarm results in the particles converging to one or more good solutions in the solution space.

4. Problem Solving Algorithm

As an aforementioned, the UPMS problem with sequence-dependant setup time is strongly NP-hard. Beside, finding a desirable solution for multi-objective combinatorial problems make them even more complex. A multi-objective problem causes the single optimal solution to convert into a set of optimal solutions named pareto optimal solutions. Thus, meta-heuristic algorithms are used to find a good solution for such a complex problems. These algorithms provide good solutions in a reasonable amount of time, making them more practicable and thus useful to the industries.

Meta-heuristic algorithms such as DE and PSO are proper methods to solve complex problems. However, these methods are not without limitations. To overcome these limitations, in this paper, we combine DE with PSO in a hybrid algorithm to solve multi-objective UPMS with sequence-dependant setup time. Since Both DE and PSO algorithms are population based, hybridization the searching abilities of both methods seem to be a good approach. Our objective is to achieve faster convergence rate and obtain better pareto optimal solutions.

Since DE and PSO algorithms are originally designed to solve problems with continuous variables, they cannot be used directly to solve discrete problems. Therefore, the proposed HDEPSO approach uses random key representation, introduced by bean [37], to encode solutions. Based on this representation scheme, the sequence of jobs on machine can be converted to continuous position values. Each position is a vector of uniform random number between 0 and 1. Thus, each solution is encoded as a vector of random keys. The proposed HDEPSO algorithm to solve the multi-objective UPMS problem consists of following steps:

Step 1: In order to establish a starting point for the optimization process, each decision parameter in every vector of the initial population is assigned a randomly chosen value from within its corresponding feasible bounds:

$$X_{j,i}^{(1)} = X_j^L + rand_j\left(X_j^U - X_j^L\right), \quad i = 1,2,\dots,NP, j = 1,2,\dots,D \qquad (19)$$

where X_j^L and X_j^U is considered between [0,1], and $rand_j$ represent a uniformly distributed random value that ranges from 0 to 1. This initial population set $P^{(t)} = \left\{X_{j,1}^{(t)}, X_{j,2}^{(t)}, \dots, X_{j,NP}^{(t)}\right\}$ consists of 2N solutions, which are randomly generated.

Step 2: The values of objective functions for each vector are evaluated: total tardiness (F_1), total workload imbalance (F_2) according to equations (1) and (2). Eliminate dominated solutions from the feasible set $P^{(t)}$.

Step 3: Compute average total tardiness ($\bar{F}_1$) and average workload imbalance ($\bar{F}_2$) in the updated$P^{(k)}$. Then for each solution vector, compute the normalized distance (D) in a two-dimensional objective space from the origin according to equation (20).

$$D = \sqrt{(F_1/\bar{F}_1)^2 + (F_2/\bar{F}_2)^2} \qquad (20)$$

Step 4: Order solution vectors in $P^{(t)}$ in descendant order. Split the ordered population set into two solution subsets: lower-half and upper-half.

Step 5: Apply the mutation and crossover operators according to equations (15) and (16) to the current lower-half subset solutions.

Step 6: Apply the movement operator of PSO according to equations (17) and (18) to the current upper-half subset solutions. These solutions belong to the next iteration population set$P^{(t+1)}$.

Step 7: Update the *pbest* and *gbest* values in the next iteration when the following corresponding conditions are met:

$$pbest_i^{(t+1)} = \begin{cases} pbest_i^{(t)} & if\ D\left(P_i^{(t+1)}\right) > D\left(P_i^{(t)}\right) \\ pbest_i^{(t+1)} & if\ D\left(P_i^{(t+1)}\right) \le D\left(P_i^{(t)}\right) \end{cases} \qquad (21)$$

$$gbest_i^{(t+1)} = \begin{cases} gbest_i^{(t)} & if\ D\left(pbest_l^{(t+1)}\right) > D\left(pbest_l^{(t)}\right) \\ gbest_i^{(t+1)} & if\ D\left(pbest_l^{(t+1)}\right) \le D\left(pbest_l^{(t)}\right) \end{cases} \qquad (22)$$

where $pbest_l^{(t)}$ is the fittest among all *pbest* in the iteration *t*th.

Step 8: the algorithm is repeated from Step (2) to Step (7) until the termination condition is met.

5. Computational Experiments

This section gives experimentation results on the performance of proposed HDEPSO to solve the considered problem. Also, the performance of the HDEPSO is compared with two well-known multi-objective genetic algorithms in the literature, namely NSGA-II and SPEA-II. All algorithms are coded in C++ programming language and executed on an Intel® Core 2 DuoE4500 at 2.20 GHz with 2.0GB of RAM.

Because of the novelty of our problem, a number of test problems are randomly generated in small, and large sizes. The processing times and setup times for jobs are randomly selected from uniform distribution between (1,20) and (1,7), respectively. In the case of setup times, after generating from the uniform distribution, the amount of setup times is corrected based on the triangular inequality $S_{ijm} + S_{jkm} \geq S_{ikm}$. The due dates are randomly picked from a uniform distribution on the interval [(*SUMP/2M*).(1-*TF*-*RDD/2M*),(*SUMP/2M*).(1-*TF*+*RDD/2M*)] where *SUMP*= $\sum_{m=1}^{M}\sum_{i=1}^{N} P_{im}$, *TF* is the tardiness factor (say 0.6), and *RDD* is the relative range of the due dates (say 0.8).

In order to determine appropriate values for the parameters required by HDEPSO, we perform extensive preliminary experiments with different set of parameters for small and large instances. Based on these experiments, we consider the following values. The initial population size *NP* is set to 50 and 100 for small and large-sized problems, respectively. In addition, the crossover parameter *CR*, the mutation parameter *F*, and the inertial weight are set to 0.3, 0.6, and 0.9. Also the social acceleration constants are considered the same and equal to 1.5.

In order to make a fair comparison between algorithms, CPU time is chosen as a stopping criterion. The computational time limit for all meta-heuristics is calculated according to $(N+M)\times\Omega$, where Ω is a constant coefficient. While different limits could be obtained by different values of Ω, the preliminary tests showed its proper amount as 0.5.

We have considered two sets of the test problems. The first set consists of 8 classes of problems called small problems, and each class contains 10 randomly generated problem instances. Therefore, 80 problem instances are considered for the small size problems. The algorithms are replicated five times on each one of the instances. In small size, the comparisons of algorithms are made in terms of the solution quality. The computational results of these tests are summarized in Table 1. In this table, the solution quality of each objective is measured by average gap of two objectives given in equations (1) and (2) between the optimal solution and the results obtained by algorithms. To be more specific average gap is computed as follows:

$$Gap_{F_i} = \sum_{r=1}^{R}\left(\frac{Method_{Sol} - Opt_{Sol}}{Opt_{Sol}} \times 100\right)\Big/R \qquad (23)$$

Where $Method_{Sol}$ is the value of the objective function F_1 and F_2 found by any of algorithms (i.e., HDEPSO, NSGA-II, and SPEA-II), and Opt_{Sol} is the corresponding optimal solution obtained by solving MILP model, and *R* is total number of replications. In order to solve proposed MILP model, we use CPLEX solver. The average gaps of the results of HDEPSO for F_1 and F_2 are less than the results obtained by NSGA-II and SPEA-II.

The second set called large size problems includes 10 classes of problems. Each class of this set contains 10 randomly generated problems, and a total of 100 problem instances are considered as large size problems. To validate the reliability and performance of the proposed HDEPSO, the following comparison metrics are used.

1. Quality metrics: This metric is simply measured by putting together the non-dominated solutions found by algorithm and the ratios between non-dominated solutions are achieved.
2. Spacing metric: we use spacing metric that provides a measure of uniformity of the spread of non-dominated solutions. This metric is given by equation (24).

$$SM = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \qquad (24)$$

where

$$d_i = \min_{j\in NDS \wedge j\neq i} \sum_{k=1}^{K} \left|F_k^i - F_k^j\right|$$

and $\bar{d}$ is the mean of all d_i, *n* is the size of obtained non-dominated solutions and f_k^i is the function value of the *k*-th objective function for solution *i*. The lower values of the SM are preferable.

Table 2 reports the related computational results for large size instances. The results reveal that the proposed HDEPSO can achieve a greater number of pareto optimal solutions with higher qualities than NSGA-II and SPEA-II. Moreover, the finding resulted from algorithms' implementations indicate that the proposed algorithm provides non-dominated solutions that have less average values of the spacing metric, also SPEA-II is superior to NSGA-II regarding to this metric.

6. Conclusion

In this paper, we have dealt with the unrelated parallel machine scheduling with sequence-dependant setup times. We have considered this problem as a bi-objective case that minimizes both tardiness and total workload imbalance simultaneously. Furthermore, we have proposed a hybrid approach by combining the searching ability of DE and PSO, namely HDEPSO, to solve this NP-hard problem. This hybridization enhances the exploration ability of the DE by the vibrancy and explorative nature of PSO. To investigate the effectiveness of our proposed approach, computational experiments were conducted and comparison results with two well-known multi-objective genetic algorithms, namely NSGAII and SPEA-II, were provided based on two comparison metrics. The results clearly show that our HDEPSO significantly outperforms the abovementioned algorithms.

Table 1- Comparison Results for small size instances

Problems	Average Gap for F_1(%)			Average Gap for F_2(%)		
	HDEPSO	NSGA-II	SPEA-II	HDEPSO	NSGA-II	SPEA-II
6×2	0.81	0.81	0.81	0.00	0.00	0.00
6×3	1.17	1.66	1.17	3.86	2.55	3.86
8×2	1.14	0.74	0.74	0.00	1.20	1.20
8×3	2.23	4.37	2.56	4.01	4.01	3.50
10×2	2.39	3.03	2.39	4.74	5.63	5.19
10×3	3.21	4.13	4.94	4.36	5.30	5.87
12×2	2.45	4.27	4.05	1.06	2.63	2.65
12×3	3.95	5.14	5.60	4.17	4.22	4.90

Table 2- Comparison Results for large size instances

Problems	Quality metric			Spacing metric		
	HDEPSO	NSGA-II	SPEA-II	HDEPSO	NSGA-II	SPEA-II
50×5	0.78	0.15	0.08	0.29	0.40	0.36
50×10	0.66	0.21	0.13	0.31	0.43	0.39
100×5	0.78	0.17	0.05	0.31	0.41	0.38
100×10	0.80	0.17	0.03	0.33	0.45	0.42
150×10	0.74	0.18	0.08	0.34	0.44	0.42
150×15	0.81	0.13	0.06	0.38	0.48	0.42
200×10	0.74	0.14	0.12	0.36	0.44	0.45
200×15	0.76	0.14	0.11	0.41	0.49	0.44
250×10	0.82	0.12	0.06	0.41	0.53	0.45
250×15	0.83	0.10	0.08	0.40	0.51	0.44

References

[1] M. Pinedo, *Scheduling Theory, Algorithms and Systems*, 3rd ed. New York: Springer Science+Business Media, LLC, 2008.

[2] F. ,. U. G. Sivrikaya, "Parallel machine scheduling with earliness and tardiness penalties," *Computer and Operations Research*, vol. 26, pp. 773-787, 1999.

[3] R. ,. M. B. ,. S. B. Logendran, "Scheduling unrelated parallel machines with sequence-dependent setups," *Computers and Operations Research*, vol. 34, pp. 3420-3438, 2007.

[4] M. ,. K. O. Azizoglu, "Scheduling jobs on unrelated parallel machines to minimize regular total cost functions," *IIE Transactions*, vol. 31, pp. 153-159, 1999.

[5] R. ,. S. F. Logendran, "Unrelated parallel machine scheduling with job splitting," *IIE Transactions*, vol. 36, pp. 359-372, 2004.

[6] V. ,. C. D. Suresh, "Minimizing maximum tardiness for unrelated parallel machines," *International Journal of Production Economics*, vol. 32, pp. 223-229, 1994.

[7] M. ,. F. J. ,. G. J. ,. Pfund, "A survey of algorithms for single and multi-objective unrelated parallel machine deterministic scheduling problems," *Journal of the Chinese Institute of Industrial Engineers*, vol. 21, pp. 230-241, 2004.

[8] S. Rajakumar, V. P. Arunachalam, and V. Selladurai, "Workflow balancing strategies in parallel machine scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 23, p. 366–374, 2004.

[9] L. ,. S. M. H. ,. P. M. ,. C. M. W. ,. F. W. J. Yu, "Scheduling of unrelated parallel machines: an application to PWB manufacturing," *IIE*

Transactions, vol. 34, pp. 921-931, 2002.

[10] D. S. ,. L. T. E. ,. K. K. H. Sun, "Component allocation and feeder arrangement for a dual-gantry multi-head surface mounting placement tool," *International Journal of production Economics*, vol. 95, pp. 245-264, 2005.

[11] T. ,. Y. M. B. ,. B. M. Keskinturk, "An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times," *Computers & Operations Research*, vol. 39, pp. 1225-1235, 2012.

[12] M. ,. K. A. B. Koksalan, "Using genetic algorithms for single machine bicriteria scheduling problems," *European Journal of Operational Research*, vol. 145, pp. 543-556, 2003.

[13] Y. H. ,. P. M. Lee, "Scheduling jobs on parallel machines with sequence dependant setup times," *European Journal of Operational Research*, vol. 100, pp. 464-474, 1997.

[14] D. W. ,. K. K. H. ,. J. W. ,. C. F. F. Kim, "Unrelated parallel machine scheduling with setup times using simulated annealing," *Robotics and Computer Integrated Manufacturing*, vol. 18, pp. 223-231, 2002.

[15] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, p. 985–1032, 2008.

[16] J. K. ,. R. K. A. H. G. ,. B. P. Lenstra, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343-362, 1977.

[17] S. Balin, "Non-identical Parallel Machine Scheduling Using Genetic Algorithm.," *Expert Systems with Applications*, vol. 38, p. 6814–6821, 2011.

[18] M. ,. R. G. ,. A.-S. A. Helal, "A Tabu Search Algorithm to Minimize the Makespan for the Unrelated Parallel Machines Scheduling Problem with Setup Times," *International Journal of Operations Research*, vol. 3, pp. 182-192, 2006.

[19] J. P. ,. R. G. ,. M. R. Arnaout, "A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times," *Journal of Intelligent Manufacturing*, vol. 21, pp. 693-701, 2010.

[20] K. C. ,. L. Z. .-J. ,. L. S. W. Ying, "Makespan minimization for scheduling unrelated parallel machines with setup times," *Journal of Intelligent Manufacturing*, pp. DOI:101007/s10845-010-0483-3, 2010.

[21] A. ,. M. L. Bilyk, "A Variable Neighborhood Search Approach for Planning and Scheduling of Jobs on Unrelated Parallel Machines," *Journal of Intelligent Manufacturing*, pp. doi:101007/s10845-010-0464-6, 2010.

[22] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Global Optimization*, vol. 11, pp. 341-359, 1997.

[23] R. Poli, "An analysis of publications on particle swarm optimisation applications," 2007.

[24] M. F. Tasgetiren, Y. Liang, M. Sevkli, and G. Gencyilmaz, "Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem," *International Journal of Production Research*, vol. 44, no. 22, pp. 4737-4754, 2006.

[25] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, p. 674–692, 2006.

[26] A. C. Nearchou, "A differential evolution approach for the common due date early/tardy job scheduling problem," *Computers & Operations Research*, vol. 35, p. 1329–1343, 2008.

[27] J. ,. E. R. C. Kennedy, "Particle swarm optimization," in *IEEE international conference on neural networks*, Piscataway, 1995, p. 1942–1948.

[28] H. K. K. ,. F. Y. ,. N. Y. Yoshida, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, p. 1232–1239, 2000.

[29] A. ,. A. I. ,. A.-M. S. Salman, "Particle swarm optimization for tast assignment problem," *Microprocessors and Microsystems*, vol. 26, p. 363–371, 2003.

[30] H. ,. L. H. ,. T. C. M. Zhang, "Particle swarm optimization for resource-constrained project scheduling," *International Journal of Project Management*, vol. 24, p. 83–92, 2006.

[31] C. ,. L. S. Andres, "A particle swarm optimization algorithm for part–machine grouping," *Robotics and Computer-Integrated Manufacturing*, vol. 22, p. 468–474, 2006.

[32] M. F. ,. L. Y. C. ,. S. M. ,. G. G. Tasgetiren, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, p. 1930–1947, 2007.

[33] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182-197, 2002.

[34] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," in *Evolutionary methods for design, optimization and control with applications to industrial problems*, Athens, 2001, pp. 95-100.

[35] R. ,. P. K. Storn, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

[36] V. Feoktistov, *Differential Evolution: In Search of Solutions*. USA: Springer, 2006.

[37] J. C. Bean, "Genetics and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 6, pp. 154-160, 1994.