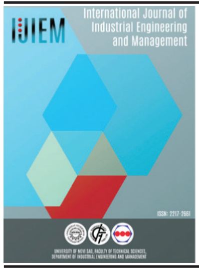
ISSN 2683-345X

journal homepage: http://ijiemjournal.uns.ac.rs/

International Journal of Industrial Engineering and Management

2024 / ARTICLE-IN-PRESS / 362



Original research article

Tri-objective parallel machine with job splitting and sequence dependent setup times using differential evolution and particle swarm optimization

W. Wisittipanich[a] 0000-0003-4515-3273, N. Wisittipanit[b,*] 0000-0002-8170-0230

[a] *Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand;*
[b] *Department of Materials Engineering, School of Science, Mae Fah Luang University, Chiang Rai, Thailand*

ABSTRACT

Parallel machines scheduling problems (PMSPs) exist in the industry since most manufacturing operations aim to produce lots of similar products in a defined time period. Some incoming jobs have different sizes and due dates; plus, the production capacity, setup time, job processing time and energy requirement of each machine can be different, possibly due to distinct models and brands. In addition, jobs can be split into sublots and processed independently on any machine; and the setup times of machines also depend on job sequences. As such, the production management involving those machines becomes exceedingly complex, particularly when the problem has multiple objectives. To obtain optimum solutions, it would require complicated mathematical model along with a solver software; however, metaheuristic algorithms might be needed if a problem becomes too large. This study applied two metaheuristic algorithms, namely differential evolution (DE) and particle swarm optimization (PSO) to the tri-objective PMSP with job splitting and sequence dependent setup times (PMSP-JSSDST) in order to obtain solutions with simultaneously minimized makespan, tardiness and total energy consumption. Both algorithms were used to solve the PMSP-JSSDST instances with some small instances being run on a commercial solver for control purpose. Then, the performances of DE and PSO were compared using hypervolume indicator. The results showed that the performances of both algorithms almost matched to those of the commercial solver for the small instances. And for the large instances, DE algorithm offers superior performances compared to those of PSO algorithm, having significantly higher values of the hypervolume indicator.

ARTICLE INFO

Article history:

Received November 16, 2023
Revised May 17, 2024
Accepted August 6, 2024
Published online September 17, 2024

Keywords:
Parallel machine;
Optimization;
Differential evolution;
Particle swarm optimization;
Hypervolume indicator

*Corresponding author:
Nuttachat Wisittipanit
nuttachat.wis@mfu.ac.th

1. Introduction

Industrial machines, when placed to operate a group of identical or similar jobs, are generally assigned to perform those operations at the same time duration in order to increase output capacity. Determining optimal ways to schedule jobs for those machines is called parallel machine scheduling problem (PMSP) which is known to be NP-Hard (Non-deterministic Polynomial-Hard). Specifically, PMSP is concerned with allocating a set of jobs to a number of parallel machines in order to optimize some measures of effectiveness e.g. minimization of makespan and cost under some production constraints e.g. resource capacities, operation procedures and time. There are three major categories of PMSP: (1) identical PMSP (2) uniform PMSP, and (3) unrelated PMSP [1]. Identical and uniform PMSP are defined as machines processing the same jobs with identical and different speeds, respectively, while unrelated PMSP is defined as machines processing different

https://doi.org/10.24867/IJIEM-2024-4-362

jobs with different speeds. Moreover, PMSP might include sequence dependent setup times (SDST) as an additional feature where the setup times of machines depend on jobs' orders of operation [2]; particularly, this kind of PMST is called PMSP-SDST. Job splitting is another feature that increases the complexity of PMSP. In this case, each job can be split into sublots and independently processed on any of parallel machines. Although job splitting reduces completion times of certain jobs, it increases overall setup frequency and delays completion of other jobs [3]. Thus, it is very important to find an appropriate number of sublots to be split from each job in order to meet desired objectives. Most studies on PMSP focus on a single objective problem especially on the minimization of makespan; however, PMSP with a single objective might not actually reflect the real-world challenges on parallel machine production where time, energy, manpower costs and material resources are also crucial factors. As such, PMSP that encompasses multiple objectives such as minimization of flow time, total tardiness, power cost, and makespan could be more suitable in the actual parallel machine production line.

This research considers the PMSP with job splitting and sequence dependent setup times, called PMSP-JSSDST, having triple objectives of minimizing makespan, tardiness and energy usage. Essentially, the study utilized the pareto-based optimization technique for this multi-objective problem where non-dominated solutions were determined in a single experimental run. The particular problem also takes into account the inequality of job sizes which can be split into sublots and distributed to be processed on available machines with different capacities, setup times, job processing times, sequence dependent setup times, energy usage rates and startup energy rates (energy usage spike at startup time). The concept of including startup energy as one of the essential parameters is that it could discourage the deployment of some machines unless totally necessary since high cost is assumed when a machine starts. Specifically, in this research, criteria of the PMSP-JSSDST are defined as follows: (1) jobs can be split to be processed on any parallel machines (2) setup times depend on the job sequence (3) production capability of each machine is not equal (4) energy usage of each machine is not equal (including startup energy), and (5) production time does not depend on job sizes. Two metaheuristics, differential evolution (DE) and particle swarm optimization (PSO) have been applied to solve such PMST-JSSDST. The solution representation of encoding and decoding procedure to transform a solution of continuous values into a practical schedule is presented. Then, the optimization performances between those two metaheuristics algorithms were compared using hypervolume indicator [4]. Moreover, parameter optimization procedures were conducted for both DE and PSO algorithms before the actual experimental runs such that suitable parameter settings were obtained. There were also several simple problems instances (PMST-JSSDST having low number of jobs and machines) that were used as a control group. Those problem instances were intended to be solved by LINGO [5] - a commercial solver software – and also by both metaheuristics algorithms in order to ensure reliability and efficiency of the optimization program.

The proposed model and solution of PMSP-JSSDST can be applied to, for instance, the heat treatment process of metal parts in which the parts can be split to various heat treatment machines and setup times depend on job sequence where different jobs require distinct temperature profiles; for examples, setup times from Job1 to Job2, where Job2 requires higher temperature profile, would be faster than those from Job2 to Job1 since heating is generally faster than cooling. Moreover, the production time of heat treatment process of each machine does not depend on job sizes since all parts in the same machine must undergo the same temperature profile; and the production capability and energy usage of each machine might not be equal. The main contribution of this research is the presentation of mathematical model for tri-objective PMSP-JSSDST, the development of tri-objective pareto-based optimization (TOPO) framework to deal with multi-objective (MO) problem, the application of metaheuristics to the problem with new proposed solution representation of encoding and decoding procedures, and the use of hypervolume indicator to evaluate metaheuristic's performances.

The rest of the article is organized as follows. Section 2 presents literature reviews related to PMSP and algorithms used in this research. Data, mathematical model, method frameworks, and solution description are given in section 3. Finally, experimental results and conclusion are provided in section 4 and 5, respectively.

2. Literature Review

Parallel machine scheduling problem (PMSP) refers to a problem that attempts to assign a collection of jobs such as machining and painting of automobile parts to a defined number of parallel machines – machines performing similar operations simultaneously – in order that some essential operational parameters e.g. makespan, mean flow time and number of tardy jobs could be optimized. PMSP also has to take into account any constraint that might be present in the operation; for instances, process criteria and resource availability. There are many variants of PMSP, which vary in the objectives, number of objectives, operational conditions and constraints. The One of the less studied variants is the PMSP with sequence dependent setup times (PMSP-SDST) which is considered to be relatively new in the field [6]. SDST refers to the setup times that depend on the sequence of jobs. Some jobs could associate with different products and the setup time for a process change from product A to product B could be different to that from product B to product A, for instance. Moreover, the setup times also rely on the technological similarities between jobs [7] in which more identical technological requirements lead to less setup times. A good example of SDST would be the tool changes on a CNC machine where the change from a simple part demanding a few cutting tools to a complex part demanding a large number of tools would require a large setup time; and it would require less setup time the other way around [8]. SDST, when related, is a major aspect in production scheduling problems and problems with SDST are considered to be relatively challenging to solve [2]; even the scheduling problems with SDST having a single machine is NP-hard [9]. Therefore, PMSP-SDST with the single objective of minimizing makespan is also NP-hard [10]. Another variant of PMSP might even include both SDST and JS (job splitting) in the same problem, called PMSP-JSSDST. Sethanan et al [11] employed two metaheuristics algorithms, DE and PSO, to solve a PMSP-JSSDST with the single objective i.e. makespan, where the problem was based on the machine operations in a fruit beverage factory. They found that the performances of DE were unambiguously better than those of PSO. Furthermore, PMSP-JSSDST was applied to the job-shop scheduling problem with a single objective that tried to minimize total tardiness of all jobs [3]; and they used Tabu Search and Simulated Annealing algorithms - both metaheuristics – to determine the optimized solutions.

Most of the PMSP studies emphasize on the single objective (SO) optimization especially the minimization of makespan [11], [12]. However, in practical use cases, there are other essential objectives such as total tardiness, power cost and minimization of machine load variation as well. A study by Torabi et al [13] explored the unrelated-PMSP (UPMSP) problem where a novel fuzzy model was created to optimize three objectives: total weighted tardiness, machine load variation, and total weighted flow time. In addition, such optimization problem was inspired by the manufacturing of various cables and wires in which specific machines and resources are required to process each job. Fang and Lin [14] tried to solve a PMSP problem using EDD (Earliest Due Date) method with double objectives – minimization of job tardiness penalty and total energy cost; in addition, the problem allowed the adjustment of machine processing power which, in this case, is the CPU frequency during workload to pursue better equilibrium between energy consumption and job processing time. Another PMSP problem which dealt with the minimization of energy usage and total tardiness employed the Ant optimization algorithm based on ATC heuristic rule (ATC-ATO method) for optimization operation and Taguchi method for determination of optimal parameters [15].

Wisittipanich and Kachitvichyanukul [16] presented a new multi-objective differential evolution (MODE) algorithm which employs data in the elite collection (non-dominated/Pareto solutions) in order to devise new mutation strategies based on the differential evolution algorithm. The optimization models based on the use of elite group are quite effective for the multi-objective problems. Mihaly and Kulcsar [17] used a novel hybrid algorithm to solve multi-objective multi-project scheduling problems. DE algorithm can also be applied to several PMSPs. Li et al [18] applied DE embedded with chaos theory to the PMSP in a real industrial lace dyeing process to minimize total tardiness and proved that the proposed algorithm performed better than the actual industrial scheduling system. Wang et al [19] investigated the PMSP using DE with the new crossover/mutation methods combined with local search in order to minimize makespan and claimed that their hybrid DE algorithm was efficient and viable. Moreover, another metaheuristics algorithm called particle swarm optimization or PSO was combined with clonal selection algorithm (CSPSO) in order to solve the parallel machine problem that tried to minimize the total tardiness [20]. Furthermore, Alharkan et al [21] utilized PSO and tabu search algorithms to optimize the scheduling of parallel machines with a single server (could be an operator or robot) in order to minimize makespan.

PSO, proposed by Kennedy and Eberhart in 1995 [22], is a population-based random search technique that imitates the behavior of fish schooling or birds flocking. In PSO, a solution is represented by a particle, and a swarm of NP particles forms the population of PSO. Each particle consists of two main attributes which are position and velocity. The evolutionary concept of PSO is that each particle applies the cognitive knowledge of its experiences (personal best, *pbest*) and the social knowledge of the swarm (global best, *gbest*) to guide itself toward the better position. In particular, the new position of a particle is governed by three main control parameters; inertia weight, personal best coefficient (*cp*), and global best coefficient (*cg*). These processes are repeated until a stopping criterion is met.

DE, proposed by Storn and Price in 1997 [23], is a population-based random search technique for global optimization over a continuous search space. A solution in DE is represented by the *D*-dimensional vector, and the population of DE consists of *NV* vectors. The key idea of DE evolutionary process is its distinct mechanism for generating new vectors through repeated cycles of three main operations: mutation, crossover, and selection. The use of few control variables; Scale Factor (*F*) and Cross-over rate (*Cr*) constitutes DE to search efficiently and fast. In selection operation, the re-placement of an individual vector occurs only if the better solution is found. As a consequence, DE is able to generate better diverse solutions since the best solution in the population does not exert any influence on the other solutions in the population.

To evaluate the quality of non-dominated solutions from a multi-objective optimization problem, recently the hypervolume indicator has been utilized extensively. The hypervolume indicator simply refers to the hyper-volume between a given reference point and a non-dominated front (Pareto front) [24]. Considering minimization of all objectives, the indicator basically maps the solution point sets in the hyperspace to the measure of the volume body which is dominated by those point sets and bounded by a chosen reference point. Sometimes, this indicator is referred to as "the size of dominated hyperspace" [25].

3. Materials and Methods

3.1 Dataset

There were a total of 37 problem instances for the experiment, and all the instances were randomly generated with the number of jobs/machines ranging from low to high. The first seven instances were quite small where the number of jobs and machines were relatively low; and these small problem instances were intended to be additionally solved with the commercial solver – LINGO. The rest of problem instances (30 instances) are larger instances where the number of jobs and machines were relatively high.

3.2 Mathematical Model

In this study, the PMSP-JSSDST is modelled as a mixed integer linear programing (MILP). The problem assumptions are listed as followed.

1. All data used in the experiment such as processing time, setup time, changeover time, and energy consumption rate are deterministic.
2. Each job can be split into several sublots, and the maximum number of sublots is determined according to the minimum machine capacity.
3. The changeover time of jobs are sequence-dependent.
4. Energy usage is based on electrical usage only.
5. All jobs are equally important.
6. Preemption of jobs is not allowed.
7. Machine breakdown are not considered.

Indices, parameters, and decision variables used in the model are defined as follows.

Indices

i, j	: Job index (i, j = 1, 2, 3, ..., Nn)
m	: Machine index (m = 1, 2, ..., Nm)
t,k	: Job sublot index (t,k = 1, 2, ..., Nt)

Parameters

Nn	: Number of jobs
Nm	: Number of machines
Nt	: Number of maximum sublots of each job
PT_{im}	: Processing time of job i on machine m
SS_{ij}	: Changeover time of job i to job j on machine m
ST_m	: Setup time of machine m
Cap_m	: Capacity of machine m
Q_i	: Production quantity of job i
SR_m	: Energy usage for start-up machine m
R_{im}	: Energy usage for processing job i on machine m
G	: Big number

Decision Variable

Q_{itm}	: Production quantity of job i sublot t on machine m
H_{itm}	: Energy usage of job i sublot t on machine m

C_{Titm} : Completion time of job *i* sublot *t* on machine *m*

Z : Production completion time or makespan

X_{itm} = 1 when job *i* sublot *t* is processed on machine *m* or 0 otherwise

Y_{itjkm} = 1 when job *i* sublot *t* is immediately processed after job *j* sublot *k* on machine *m* or 0 otherwise

W_{it1lm} = 1 when job *i* sublot *t* is the first operation on machine *m* or 0 otherwise

U_{1ljkm} = 1when job *j* sublot *k* is the last operation on machine *m* or 0 otherwise

This study focuses on three objectives which are 1) minimization of makespan 2) minimization of total tardiness and 3) minimization of total energy usage. The mathematical models are shown as the following.

Mathematical model 1: Minimization of makespan

Objective Function:

$$Min\ Z \qquad (1)$$

Constraints:

$$Q_{itm} \leq Cap_m - (Cap_m * (1 - X_{itm})) \qquad \forall itm \qquad (2)$$

$$\sum_{m=1}^{Nm}\sum_{t=1}^{Ns} Q_{itm} = Q_i \qquad \forall i \qquad (3)$$

$$X_{jkm} - U_{itjkm} - \sum_{i=1}^{Nn}\sum_{t=1}^{Nt} Y_{itjkm} = 0 \qquad \forall jkm, i = 1, t = 1 \qquad (4)$$

$$X_{jtm} - W_{itjkm} - \sum_{j=1}^{Nn}\sum_{k=1}^{Nt} Y_{itjkm} = 0 \qquad \forall itm, j = 1, k = 1 \qquad (5)$$

$$\sum_{j=1}^{Nn}\sum_{k=1}^{Nt} U_{itjkm} \leq 1 \qquad \forall m\ ;\ i = 1, t = 1 \qquad (6)$$

$$\sum_{j=1}^{Nn}\sum_{k=1}^{Nt} W_{itjkm} \leq 1 \qquad \forall m\ ;\ j = 1, k = 1 \qquad (7)$$

$$\sum_{j=1}^{Nm} x_{itm} \leq 1 \qquad \forall it \qquad (8)$$

$$CT_{itm} \geq PT_{im} + ST_m \qquad \forall im, t = 1 \qquad (9)$$

$$C_{itm} \geq C_{jkm} + (PT_{im} * X_{itm}) + (SS_{ij} * X_{itm}) \qquad \forall itjm, k = t - 1 \qquad (10)$$

$$Z \geq CT_{itm} \qquad \forall itm \qquad (11)$$

$$Q_{itm} \geq 0 \qquad \forall itm \qquad (12)$$

$$X_{itm}, Y_{itjkm}, U_{itjkm}, W_{itjkm} \in \{0,1\} \quad \forall itm \qquad (13)$$

The objective function of minimizing makespan is shown in equation (1). Equation (2) indicates that the quantity of each job sublot cannot exceed the machine capacity. Equation (3) ensures that the total production quantity of any job is processed. Equation (4) states that, besides the first sublot on a machine, there must be a sublot that precedes other sublots on the same machine. Equation (5) states that besides the last sublot on a machine, there must be a sublot that succeeds other sublots on the same machine. Equation (6) and (7) guarantee that each machine can operates at most one first sublot and one last sublot. Equation (8) ensures that each job sublot is assigned to at most one machine. Equation (9) and (10) illustrate the relationship of completion time of jobs on each machine. Equation (11) calculates the makespan. Equation (12) states that the quantity of job sublot is a non-negative. And Equation (3) specifies the binary decision variables.

Mathematical model 2: Minimization of total tardiness

Objective Function:

$$Min \sum_{m=1}^{Nm} TD_i \qquad (14)$$

Constraints:

$$TD_i \geq CT_{itm} - DD_i - (G * (1 - X_{itm}) \quad \forall itm \qquad (15)$$

$$TD_i \geq 0 \qquad \forall i \qquad (16)$$

The objective function of minimizing total tardiness of jobs is illustrated in equation (14). In addition to constraints (2) to (13), two constraints regarding to job tardiness are added in this model. Equation (15) and (16) represent the tardiness calculation of each job which cannot be negative values.

Mathematical model 3: Minimization of total energy usage

Objective Function

$$Min \sum_{m=1}^{Nm} HMAX_m \tag{17}$$

Constraints

$$H_{itm} \geq (SR_m + R_{im}) * X_{itm} \qquad \forall itm \tag{18}$$

$$H_{itm} \geq H_{i,t-1,m} + (R_{im} * X_{itm}) \qquad \forall itm, t \neq 1 \tag{19}$$

$$HMAX_m \geq H_{itm} \qquad \forall itm \tag{20}$$

The objective function of minimizing total energy usage is shown in equation (17). Additional constraints related to energy usage are added. Equation (18) and (19) ensure the relationship of the energy usage for job i on each machine must be greater than or equal to energy usage for starting-up a machine and operating a job sublot on that machine. Equation (20) determines the total energy usage on machine m.

3.3 Tri-objective pareto-based optimization framework

Most traditional optimization techniques have been developed for optimizing only one single objective. When multiple objectives are considered in the problem, most research often simplify the problem by linearly combining different objectives into one objective with priority weights; and a solution is obtained based on predetermined weights. The difficulties of this approach are that a solution highly depends on given weights of a decision maker, and different decision makers are subjected to different perspectives.

Different from single-objective optimization, the pareto-based optimization is a weight-free method which aims to search for a set of non-dominated solutions instead of one single solution. Consequently, it allows decision makers to simultaneously find the trade-offs or non-dominated solutions on the Pareto front in a single run without prejudice.

Unfortunately, the approach of pareto-based optimization is more difficult and different from single objective for several aspects. First, the mechanism to select the best member, and, second, the way to evaluate the quality of solutions. Therefore, an efficient algorithm is required in order to find a set of high quality non-dominated solutions (Introduction). This study proposes a tri-objective pareto-based optimization (TOPO) framework to deal with multi-objective (MO) problem as shown in Figure 1.

The TOPO framework begins with initializing population in a random manner. Next, each population member is evaluated with three different objectives. Then, similar to the Elitist structure in NSGA-II [26] and other pareto optimization frameworks, the population experience is stored in an external archive, called Elite group, as a set of non-dominated solutions. In TOPO, instead of applying the sorting procedure in Elite group to every single evolution of a member, the sorting is only performed on the set of newly generated population in order to identify the group of new non-dominated solutions. This sorting procedure applies to the group of new solutions and current solutions in the external archive and store only non-dominated solutions into an archive for the Elite group. Then, Elite group screens its solutions to eliminate inferior solutions using hypervolume indicator. These processes are repeated until a stopping criterion is met. As a consequence, the Elite group in the archive contains only the best non-dominated solutions found so far in the searching process.

3.4 Solution representation

To apply PSO and DE for a combinatorial problem of parallel machine problem, the solution representation is required to transform the real number of D-dimensional space into practical schedule. Figure 2 presents the procedures of solution representation with encoding and decoding scheme. An example of parallel machine with 2 jobs and 2 machines in Table 1 is used to illustrate the solution representation procedures in this study.

The solution representation procedure starts with the encoding process which aims to represent a solution as a string of dimensions. First, the number of possible maximum sublots of each job is determined by dividing the production quantity of job by minimum machine capacity. Then, the number of possible maximum sublots of all jobs are summarized as *No. maximum sublots*. The number of dimensions (D) is calculated as equation (21), and each value in a dimension is initially filled with a uniform random number in the range [0, 1].

$$No.dimensions = No.maximum\, sublots * (Nm + 1) \tag{21}$$

In this example, the maximum sublots of J1 is 30/20 = 2 sublots and the maximum splits of J2 is

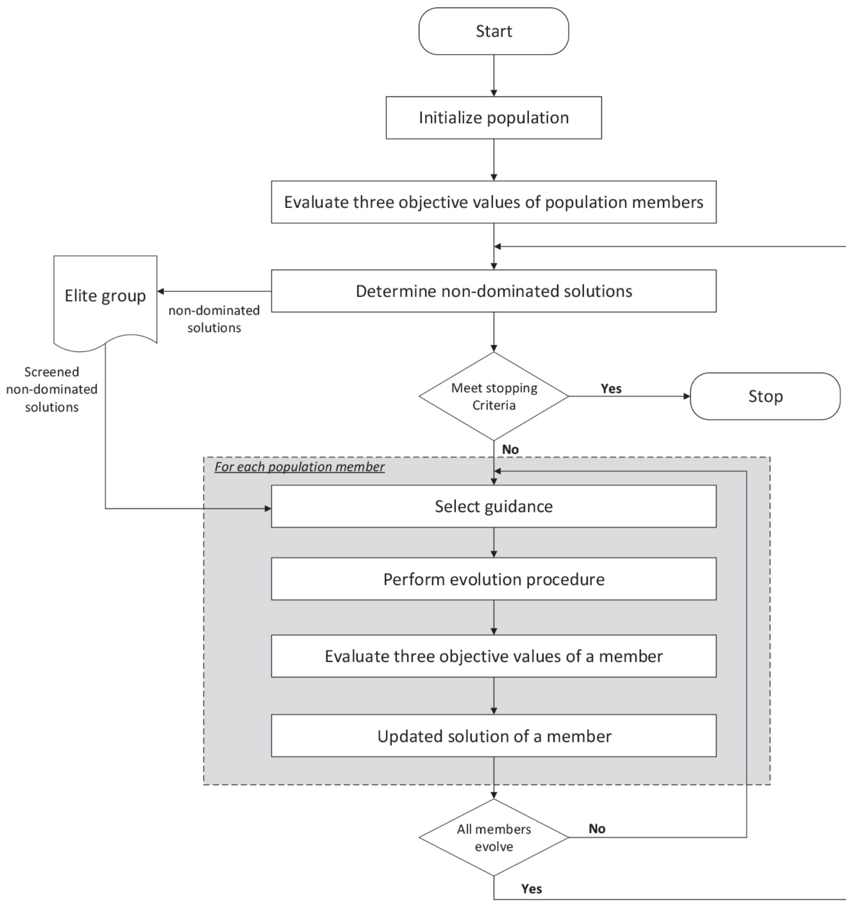

Start
Initialize population
Evaluate three objective values of population members
Determine non-dominated solutions
Elite group
non-dominated solutions
Screened non-dominated solutions
Meet stopping Criteria
Yes
Stop
No
For each population member
Select guidance
Perform evolution procedure
Evaluate three objective values of a member
Updated solution of a member
All members evolve
No
Yes

Figure 1. The tri-objective pareto-based optimization (TOPO) framework dealing with multi-objective (MO) problem

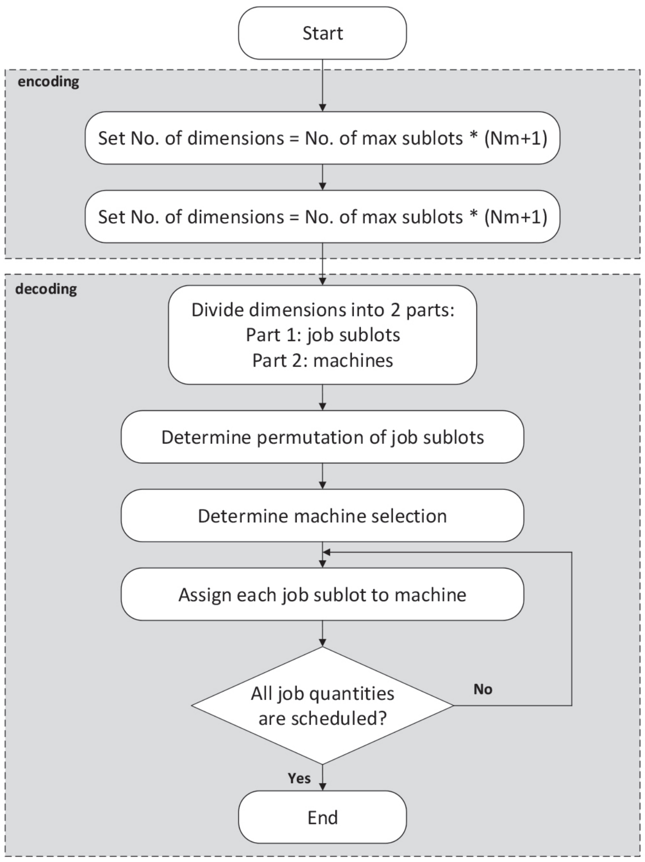

Start
encoding
Set No. of dimensions = No. of max sublots * (Nm+1)
Set No. of dimensions = No. of max sublots * (Nm+1)
decoding
Divide dimensions into 2 parts:
Part 1: job sublots
Part 2: machines
Determine permutation of job sublots
Determine machine selection
Assign each job sublot to machine
All job quantities are scheduled?
No
Yes
End

Figure 2. Solution representation procedures

Table 1. Data example of parallel machine with 2 jobs and 2 machines

Parameter	Value	
	Job 1	**Job 2**
Job Quantity	30	50
Due Date	50	100
	Machine1	**Machine2**
Machine Capacity	20	40
Machine Setup Time	5	4
Job Processing Time $\begin{bmatrix} P11 & P12 \\ P21 & P22 \end{bmatrix}$	$\begin{bmatrix} 54 & 56 \\ 43 & 41 \end{bmatrix}$	
Sequence Dependent Setup Time $\begin{bmatrix} S11 & S12 \\ S21 & S22 \end{bmatrix}$	$\begin{bmatrix} 0 & 5 \\ 15 & 0 \end{bmatrix}$	
Startup Energy [M1 M2]	[0.8715 0.5987]	
Energy Usage $\begin{bmatrix} R11 & R12 \\ R21 & R22 \end{bmatrix}$	$\begin{bmatrix} 9.561 & 5.956 \\ 7.869 & 6.018 \end{bmatrix}$	

50/30 = 3 sublots. Thus, the number of dimensions is 5*(2+1) = 15 dimensions. Supposed that each value in a dimension is initially generated with a uniform random number between 0 and 1, a random key representation of a *D*-Dimensional string is shown in Figure 3.

For the decoding process, a *D*-dimensional string is divided into 2 main parts. The first part denoted the possible maximum job sublots, and the second part is related to the selection of machines. In this example, as shown in Figure 4, dimension 1 to 5 represents 5 maximum sublots, dimension 6 to 10 stands for machine 1, and dimension 11 to 15 stands for machine2.

Next, the process to transform a *D*-Dimensional string to a solution comprises three main stages: (1) the determine the permutation of job sublots (2) determining the machine selection and (3) the assignment of each sublot to a machine. The permutation of job sublots is performed according to the permutation of n-repetition of n jobs with ascending sort rule. The machine selection is decided using the maximum-value rule. Then, in the assignment stage, each job sublot is assigned to a machine accordingly. Figure 5(a) illustrate the permutation process. Figure 5(b) show and Figure 5(c) show the assignment of each sublots to a machine to process according to the dimension value.

Then, the parallel machine schedule is obtained by taking the first sublot from the sequential orders, then the second sublot, and so on until all job quantities are considered. In the process of generating a schedule, each sublot is processed on the selected machine in the best available position without delay-

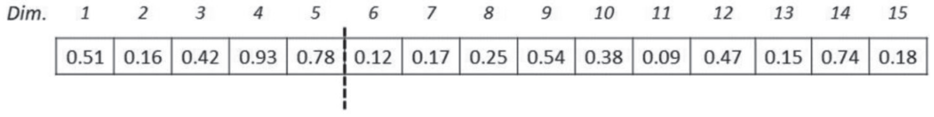


Figure 3. Random key representation of a D-Dimensional string

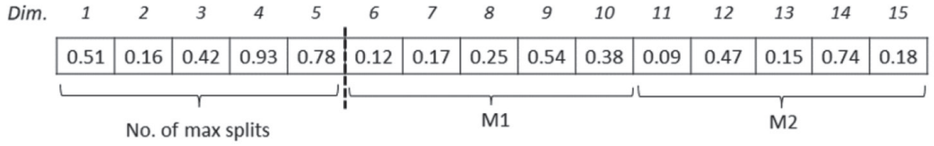
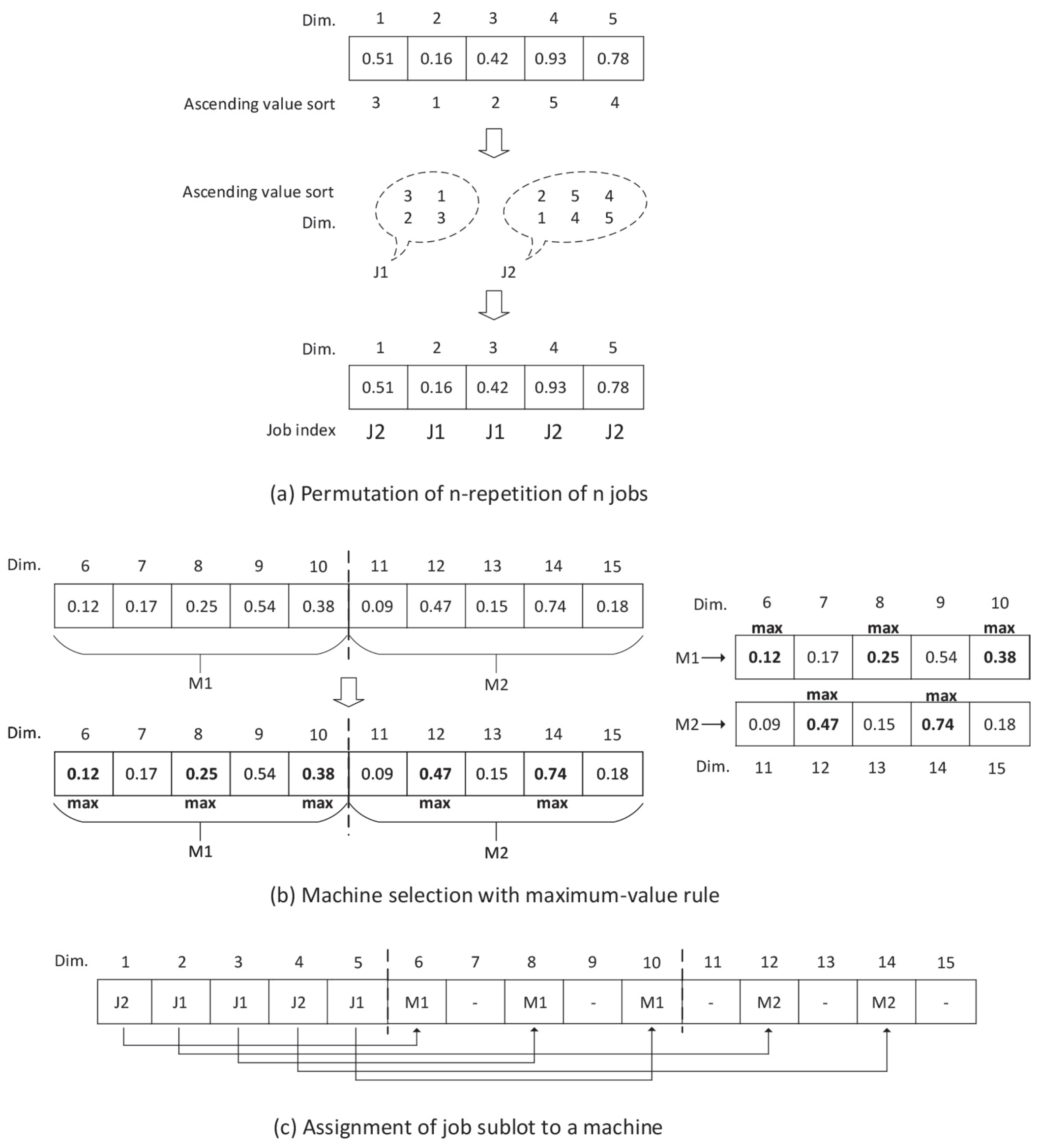


Figure 4. Representation of a D-Dimensional string to an example


Dim. 1 2 3 4 5
0.51 0.16 0.42 0.93 0.78
Ascending value sort 3 1 2 5 4
Ascending value sort 3 1 / 2 5 4
Dim. 2 3 / 1 4 5
J1
J2
Dim. 1 2 3 4 5
0.51 0.16 0.42 0.93 0.78
Job index J2 J1 J1 J2 J2
Dim. 6 7 8 9 10 11 12 13 14 15
0.12 0.17 0.25 0.54 0.38 0.09 0.47 0.15 0.74 0.18
M1
M2
max
M1⟶ 0.12 0.17 0.25 0.54 0.38
M2⟶ 0.09 0.47 0.15 0.74 0.18
Dim. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
J2 J1 J1 J2 J1 M1 - M1 - M1 - M2 - M2 -

(a) Permutation of n-repetition of n jobs

(b) Machine selection with maximum-value rule

(c) Assignment of job sublot to a machine

Figure 5. Decoding procedures to transform D-dimensional solution

ing other scheduled sublots. Thus, this procedure always results in an active schedule. According to this example, the final schedule is obtained as Figure 6.

Then, three objectives are evaluated accordingly; Makespan = 106, Tardiness = 16, and Energy usage = 21.3132.

It is worth noting that although the proposed decoding method always generate a feasible and active schedule, there are some limitations. In the decoding stage of machine selection, each job sublot is assigned to a machine according to the maximum-value rule, which often results in diversity of solutions without any bias. However, there are also disadvantages with this rule because the machine assignment is pre-determined without considering the current scheduled sublots. In addition, the schedule is not dynamically generated given the multi-objectives. As a consequence, this method sometimes may not result in generating optimal schedule or it may miss a chance to obtain better schedule.

4. Experimental Results

4.1 Parameter settings of metaheuristics and test instances

Generally, the settings and adjustments of crucial parameters of any metaheuristic algorithms most likely affect its optimization performances, more or less. Setting appropriate parameters could also decide how well an algorithm performs and also how fast the

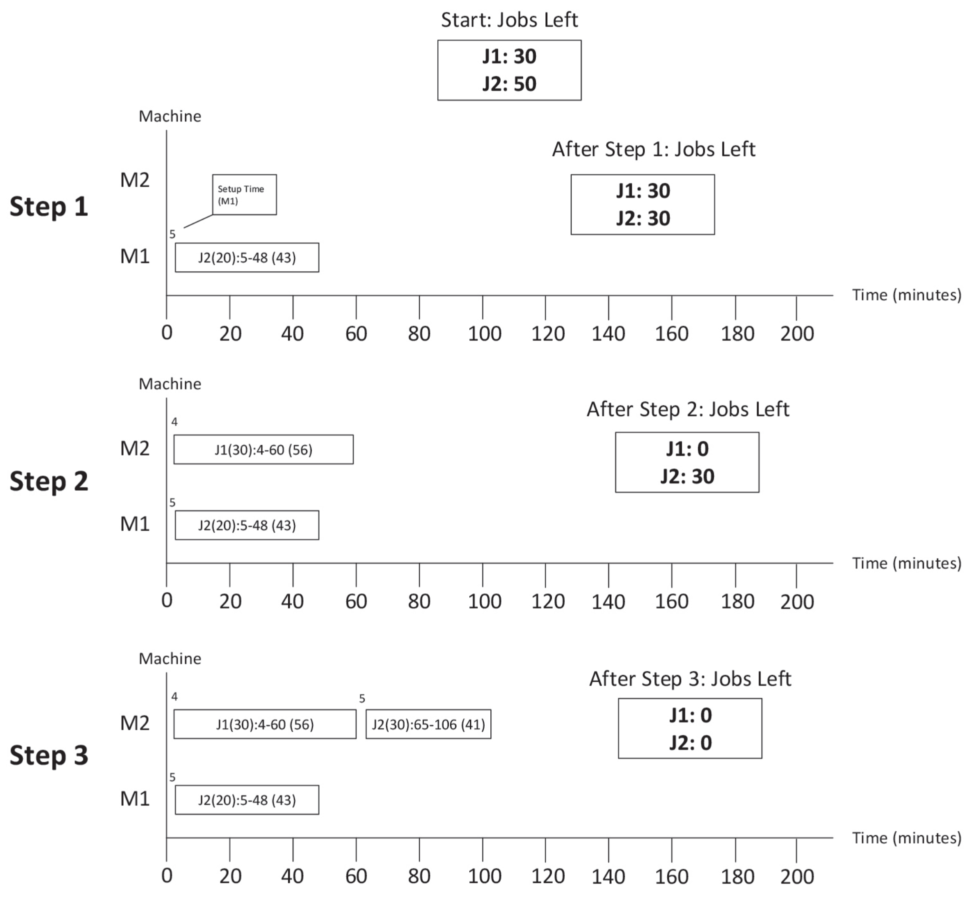


Figure 6. Scheduling of jobs

solutions could converge. The performance of the proposed methods was tested using 37 randomly generated test instances. The problem sizes are classified as small, medium, and large depending on the maximum number of possible sublots. The parameters of each problem size are shown in Table 2.

In this study, parameter optimization processes were performed for both DE and PSO algorithms. However, there was one common parameter not directly related to the core models in both algorithms which was the “PercentRandom” – the percentage of iterations that use random vector members to evolve a predecessor vector (otherwise it would use a vector from the elite group.). Thus, for running the DE algorithm, there were three parameter sets: (1) Fmax/Fmin (2) CRmax/CRmin, and (3) PercentRandom

Table 2. Parameters for test instances

Parameters	**Problem Size**		
	Small	**Medium**	**Large**
Sublot size	< 30	30 – 220	> 220
Nn	2 - 4	4 – 25	25 - 30
Nm	2 - 3	3 - 20	20 - 35
Cap_m	10, 20, 40	10, 20, 40, 60	10, 20, 40, 60
PT_{im}	U[10, 50]	U[10, 50]	U[10, 50]
SS_{ij}	U[0.1(PT), 0.25(PT)]	U[0.1(PT), 0.25(PT)]	U[0.1(PT), 0.25(PT)]
ST_m	U[0.1(PT), 0.15(PT)]	U[0.1(PT), 0.15(PT)]	U[0.1(PT), 0.15(PT)]
SR_m	U[2, 2.5]	U[2, 2.5]	U[2, 2.5]
R_{im}	U[4, 35]	U[4, 35]	U[4, 35]

which were altered in experiment trials during the parameter optimization process. And each parameter set comprises 3 values as shown in Table 3. As such, for each data instance, there were a total of 27 trial runs (equal to the number of parameter combinations i.e. 3^3). However, those trial runs were only applied to the problem with medium size: J20M10 (number of jobs =20, number of machines =10). Then, a parameter combination yielding the best optimization performance was selected for all of the problem instances in the experimental runs. And for PSO algorithm, there were also three parameter sets: (1) wMax/wMin (2) TopEP/BotEP, and (3) PercentRandom as shown in Table 4. And the procedure to find the best parameter combination was similar to that of DE algorithm. Those methods for parameter optimization resulted in parameter selection as follows: for DE| Fmax/Fmin = 1.5/0.5|CRmax/CRmin = 0.5/0.1|PercentRandom = 60, for PSO| wMax/wMin = 0.4/0.9|TopEP/BotEP = 0.1/0.2| PercentRandom = 60. It is noted that these selected parameters are underlined in both Table 3 and Table 4.

For both algorithms in each run of a problem instance, the population/swarm sizes (DE utilizes population size while PSO uses swarm size) depended on how large a problem was; note that a problem size equals to the number of maximum job sublots. For examples, a small problem (sublot size <30) matches with the population/swarm size of 200, and a medium problem (30≤ sublot size ≤220) matches with the population/swarm size of 500. However, the number of iterations remain constant regardless of the problem size; and both algorithms employed the same encoding/decoding procedure regardless of the problem sizes. Tables 5 shows the population or swarm size and number of iterations used for different problem sizes in the numerical experiment of metaheuristics. It is noted that these settings were empirically determined from preliminary experiments.

4.2 Computational results

For this research, the metaheuristics algorithms were implemented by the Python programming language [27] under PyCharm IDE [28] version 2020.3.1 in Ubuntu operating system version 20.04.5 LTS. The optimization application ran on the platform Intel® CoreTM i7-10750H CPU 2.60 GHz with 32 GB of RAM. The performance comparisons between DE and PSO algorithms were evaluated using the hypervolume indicator provided by the Python library pygmo version 2.18 [29]. There were a total of 37 instances in which the first seven instances were solved with DE and PSO algorithms as well as the solver software – LINGO; and the other 30 data instances were solved with only DE and PSO algorithms. LINGO is an optimization tool designed to solve both linear and non-linear models which aims to find the exact solution for each optimization problem. However, for some problems that are quite large, LINGO might take a long time to determine the solution (in some cases, LINGO could take the

Table 3. Parameter testing for DE algorithm

Parameter Name	Value 1	Value 2	Value 3
Fmax/Fmin	2/1	2.5/1.5	1.5/0.5
CRmax/CRmin	0.9/0.1	0.9/0.5	0.5/0.1
PercentRandom	20	40	60

Table 4. Parameter testing for PSO algorithm

Parameter Name	Value 1	Value 2	Value 3
wMax/wMin	0.1/0.4	0.25/0.7	0.4/0.9
TopEP/BotEP	0.1/0.2	0.2/0.3	0.3/0.4
PercentRandom	20	40	60

Table 5. Population/Swarm size and number of iterations for different problem sizes in the numerical experiment of DE/PSO

Problem Size	Sublot Size	Population/Swarm Size	Iteration
Small	< 30	200	1000
Medium	30 – 220	500	1000
Large	> 220	1000	1000

infinite time since a problem is too large). For each data instance, the hypervolume indicator values were computed using non-dominated solutions obtained from both DE and PSO algorithms; and the reference point for the indicator was calculated by adding ten to the maximum value of each objective across both results. Note that each data instance has the same reference point. In addition, the higher value of hypervolume indicator, the better of non-dominated solutions quality. Table 7 shows the comparisons of hypervolume indicators for all instances solved by DE and PSO algorithms, and also percent difference (PD) of hypervolume values between them. The formula for PD is given by equation 22.

$$PD = \frac{HV(DE)-HV(PSO)}{HV(PSO)} \times 100\% \tag{22}$$

From the results in Table 7, the performances of DE, PSO and LINGO are identical in the first three problem instances where the problems are small ($Nn = 2$, $Nm \leq 3$) which means both metaheuristics algorithms could find the solutions similar to the exact optimized solutions obtained from LINGO. When the problems get larger e.g. instance 4, 5 and 6 ($3 \leq Nn \leq 4$, $Nm \leq 3$), both algorithms almost reach the exact optimized solutions in all three objectives except the energy usage where LINGO needed more than 12 hours to achieve the exact optimized results for instance 4 and 5 while it could not find the solution of energy usage for instance 6 at all. Lastly, for instance 7 ($Nn = 4$, $Nm = 3$), both metaheuristics algorithms almost achieve the exact optimized solutions of makespan and even perform better than LINGO does in the solution of tardiness; however, LINGO could not obtain the exact optimized solution of energy usage.

For problem instances in which the $Nn \geq 4$ and $Nm \geq 3$, there are only performances results from DE and PSO (problems are simply too large for LINGO); and the total number of instances in this case is 30. Note that the largest instance is D25 where $Nj = 50$ and $Nm = 35$. From the results in Table 7, based on the hypervolume indicator values (HV) in all of the instances from D1 to D30, the performances of DE are clearly superior to those of PSO with the average PD of 45.22%. Table 8 presents the best optimization results for DE and PSO in each instance (D1 – D30) including 3 objectives: (1) makespan (2) tardiness and (3) energy usage. Note that each objective value for each instance is the optimal value chosen from a group of non-dominate solutions (pareto front). The results from Table 8 show that almost all the best makespan values obtained from DE (except in the instances D2 and D5) are better (lower) than those from PSO. Moreover, all the best tardiness values from DE in two instances (D2 and D3) are superior to those from PSO (the rest are all zeros). And lastly, most of the best energy usage values gained from DE are better than those from PSO. This indicate that DE is capable of finding greater non-dominated solutions or better pareto front than PSO. Therefore, DE is an alternate effective algorithm for solving this complex parallel machine problem i.e. PMSP-JSSDST.

5. Conclusion

This research utilized and compared two metaheuristics algorithms, namely DE and PSO algorithms, to solve the PMSP-SDST for three optimal objectives: makespan, tardiness and energy usage. In PMSP-SDST, job sizes were not required to be

Table 6. Experimental results of DE compared with those of PSO and LINGO for seven instances

Instance	*Nn*	*Nm*	LINGO			DE			PSO		
			MS	TD	EN	MS	TD	EN	MS	TD	EN
1	2	2	106	16	18.59	106	16	18.59	106	16	18.59
2	2	3	36	11	122.67	36	11	122.67	36	11	122.67
3	2	3	14	2	67.35	14	2	67.35	14	2	67.35
4	3	3	73	81	56.10*	73	81	67.35	73	81	67.35
5	3	2	88	18	106.75*	88	18	116.41	88	18	116.41
6	4	2	77	44	N/A	77	44	44.23	77	44	44.23
7	4	3	95*	164*	N/A	99	163	32.53	99	163	32.53

Note. *LINGO takes more than 12 hours to obtain the results, the reported value is lower bound value. *Nn* = Number of jobs, *Nm* = Number of machines, MS = Makespan, TD = Tardiness and EN = Energy Usage.

Table 7. Hypervolume indicator values (HV) and percent difference (PD) for all instances solved by DE and PSO algorithms

Instance	*Nn*	*Nm*	HV: DE	HV: PSO	PD (%)
D1	4	3	9122.07	8904.82	2.44
D2	7	4	2124880.60	1911632.91	11.16
D3	10	4	7324427.00	6069451.08	20.68
D4	10	5	4274798.72	3359147.28	27.26
D5	12	5	39099.59	34936.19	11.92
D6	20	10	9259830.10	6160254.80	50.32
D7	20	15	173382.40	113990.30	52.1
D8	25	10	3992856.98	3233869.86	23.47
D9	25	15	191475.50	140014.29	36.75
D10	25	20	175291.55	132414.36	32.38
D11	30	10	31549000.09	30146900.83	4.65
D12	30	15	550634.97	331342.08	66.18
D13	30	19	259763.74	139029.64	86.84
D14	30	25	200260.48	116080.81	72.52
D15	35	10	406045.03	310187.81	30.9
D16	35	15	476594.33	386708.57	23.24
D17	35	20	155061.35	61177.38	153.46
D18	35	25	291461.25	171926.38	69.53
D19	35	30	219609.99	136495.68	60.89
D20	40	10	487905.57	335144.72	45.58
D21	40	15	308834.73	179700.49	71.86
D22	40	19	5571727.61	4352713.57	28.01
D23	40	25	418672.61	311746.46	34.3
D24	40	30	392254.98	265819.21	47.56
D25	50	35	335296.37	186512.26	79.77
D26	50	30	366741.55	257288.49	42.54
D27	50	25	480517.83	308038.04	55.99
D28	50	20	1154778.33	921519.80	25.31
D29	50	15	891359.94	663957.78	34.25
D30	50	10	157507.58	101743.79	54.81
				Average	45.22

Table 8. The best value of each objective obtained from DE and PSO

Instance	*Nn*	*Nm*	DE			PSO		
			MS	TD	EN	MS	TD	EN
D1	4	3	111	0	232.28	112	0	232.28
D2	7	4	136	192	366.67	132	202	364.46
D3	10	4	152	310	408.99	160	319	417.23
D4	10	5	154	0	449.98	156	0	442.04
D5	12	5	74	0	220.37	74	0	219.36
D6	20	10	231	0	1368.75	241	0	1390.55
D7	20	15	144	0	1182.53	151	0	1196.46
D8	25	10	225	0	1443.96	236	0	1437.83
D9	25	15	165	0	1314.94	170	0	1343.37
D10	25	20	167	0	1835.07	166	0	1865.90
D11	30	10	294	0	1809.74	304	0	1777.12
D12	30	15	191	0	1571.79	201	0	1652.29

D13	30	19	174	0	1754.32	184	0	1811.45
D14	30	25	156	0	1983.32	167	0	2038.75
D15	35	10	230	0	1549.67	253	0	1578.39
D16	35	15	226	0	1838.57	238	0	1914.65
D17	35	20	215	0	2421.24	225	0	2427.16
D18	35	25	180	0	2461.45	190	0	2537.12
D19	35	30	163	0	2491.35	165	0	2558.65
D20	40	10	352	0	2370.89	368	0	2399.24
D21	40	15	214	0	1847.73	225	0	1919.14
D22	40	19	244	0	2650.46	253	0	2736.70
D23	40	25	200	0	2690.59	207	0	2736.57
D24	40	30	205	0	2974.02	207	0	3034.25
D25	50	35	200	0	3667.09	214	0	3686.99
D26	50	30	203	0	2979.60	209	0	3029.99
D27	50	25	201	0	2676.76	210	0	2737.31
D28	50	20	256	0	2849.22	268	0	2882.83
D29	50	15	262	0	2257.76	263	0	2299.48
D30	50	10	340	0	2249.82	349	0	2251.71

Note. *Nn* = Number of jobs, *Nm* = Number of machines, MS = Makespan, TD = Tardiness and EN = Energy Usage.

equal and jobs could be split into small lots such that they can be allocated to vacant machines. Essential parameters of each machine included setup times, sequence dependent setup times, energy usage and energy consumption at machine startup time. In particular, the startup energy usage was included in the machine parameters to dissuade the utilization of a machine because of high energy cost at the startup time, not to mention manpower and machine deterioration costs.

In order to apply DE and PSO to the problem, solution representation with encoding and decoding procedures were designed to obtain feasible solutions. Moreover, the parameter optimization procedure was implemented for both DE and PSO algorithms in order to obtain appropriate values of critical parameters. Both metaheuristics algorithms were verified for their performances by comparing their results with the commercial solver – LINGO – in the case of small problem instances, in which the results demonstrated that both algorithms, DE and PSO, had almost identical performances compared to those of the LINGO. In the case of large problem instances, using the hypervolume indicators computed from the pareto front to compare optimization performances of multi-objective problems, the results showed that DE clearly outperformed PSO in all the problem instances.

For real practice in the production line, the proposed algorithm can be integrated into the existing scheduling system to generate an efficient schedule. The findings of this study can help in developing visual tool to continuously monitor schedule status. In addition, it helps production planner to make decision corresponding to different objectives such that the organization's goals are met. The ongoing research have continued to investigate ways to improve the algorithm performance, for example, hybridization with other metaheuristics and adaptive search features. Further studies are recommended to apply the proposed algorithm for a wider range of scheduling or related problems and include other realistic constraints into the problem.

Acknowledgments

The authors would like to acknowledge Mae Fah Luang University and Chiang Mai University for a partial financial sponsorship and their facility support in this research.

Funding

This work was supported by the Mae Fah Luang University [grant agreement number: 621B01019].

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] T.C.E. Cheng, and C.C.S. Sin, "A state-of-the-art review of parallel-machine scheduling research," European Journal of Operational Research, vol. 47, no. 3, pp. 271–579, Aug. 1990, doi: 10.1016/0377-2217(90)90215-W.

[2] J. Behnamian, M. Zandieh, and S.M.T.F. Ghomi, "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm," Expert Systems with Applications, vol. 36, no. 6, pp. 9637–9644, Aug. 2009, doi:10.1016/j.eswa.2008.10.007.

[3] I. Saricicek, and C. Celik, "Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness, SA and VNS hybrid algorithm," Applied Mathematical Modelling, vol. 35, no. 8, pp. 4117–4126, Aug. 2011, doi: 10.1016/j.apm.2011.02.035.

[4] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," Theoretical Computer Science, vol. 425, pp. 75–103, Mar. 2012, doi: 10.1016/j.tcs.2011.03.012.

[5] LINGO - Optimization Modeling Software for Linear, Nonlinear, and Integer Programming. (2021). LINDO Systems. Available: https://www.lindo.com

[6] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," Theoretical Computer Science, vol. 425, no. 30, pp. 75–103, Mar. 2012, doi: 10.1016/j.tcs.2011.03.012.

[7] B.N. Srikar, and S. Ghosh, "A MILP model for the n-job, M-stage flowshop with sequence dependent set-up times," International Journal of Production Research, vol. 24, no. 6, pp. 1459-1474, Oct. 2007, doi: 10.1080/00207548608919815.

[8] F. Yalaoui, and C. Chu, "An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times," IIE Transactions, vol. 35, no. 2, pp. 183-190, Oct. 2010, doi: 10.1080/07408170304382.

[9] M.L. Pinedo, "Advanced Single Machine Models, " in Scheduling: Theory, Algorithms, and Systems, fifth edition. New York, USA: Springer, 2016, pp. 69-77.

[10] S.A. Kravchenko, and F. Werner, "Parallel machine problems with equal processing times: a survey," Journal of Scheduling, vol. 14, pp. 435-444, Mar. 2011, doi: 10.1007/s10951-011-0231-3.

[11] K. Sethanan, W. Wisittipanich, N. Wisittipanit, K. Nitisiri, and K. Moonsri, "Integrating scheduling with optimal sublot for parallel machine with job splitting and dependent setup times," Computer & Industrial Engineering, vol. 137, 106095, Nov. 2019, doi: 10.1016/j.cie.2019.106095.

[12] A. Kurt, and F.C. Cetinkaya, "Unrelated parallel machine scheduling under machine availability and eligibility constraints to minimize the makespan of non-resumable jobs," International Journal of Industrial Engineering and Management, vol. 15, no. 1, pp. 18-33, Mar. 2024, doi: 10.24867/IJIEM-2024-1-345.

[13] S.A. Torabi, N. Sahebjamnia, S.A. Mansouri, and M.A. Bajestani, "A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem," Applied Soft Computing Journal, vol. 13, no. 12, pp. 4750-4762, Dec. 2013, doi: 10.1016/j.asoc.2013.07.029.

[14] K.T. Fang, and B.M.T. Lin, "Parallel-machine scheduling to minimize tardiness penalty and power cost," Computer & Industrial Engineering, vol. 64, no. 1, pp. 224-234, Jan. 2013, doi: 10.1016/j.cie.2012.10.002.

[15] P. Liang, H.D. Yang, G.S. Liu, and J.H. Guo, "An ant optimization model for unrelated parallel machine scheduling with energy consumption and total tardiness," Mathematical Problems in Engineering, vol. 2015, 907034, Aug. 2015, doi: 10.1155/2015/907034.

[16] W. Wisittipanich, and V. Kachitvichyanukul, "Mutation strategies toward Pareto front for multi-objective differential evolution algorithm," International Journal of Operational Research, vol. 19, no. 3, pp. 315-337, Jun. 2014, doi: 10.1504/IJOR.2014.059507.

[17] K. Mihaly and G. Kulcsar, "A new many-objective hybrid method to solve scheduling problems," International Journal of Industrial Engineering and Management, vol. 14, no. 4, pp. 326-335, Dec. 2023, doi: 10.24867/IJIEM-2023-4-342.

[18] D. Li, J. Wang, R. Qiang, and R. Chiong, "A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility," International Journal of Production Research, vol. 59, no. 9, pp. 2722-2738, Mar. 2020, doi: 10.1080/00207543.2020.1740341.

[19] W.L. Wang, H.Y. Wang, Y.W. Zhao, L.P. Zhang, and X.L. Xu, "Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm,". Computers & Operations Research, vol. 40, no. 5, pp. 1196-1206, May 2013, doi: 10.1016/j.cor.2012.12.007.

[20] Q. Niu, T. Zhou, and L. Wang, "A hybrid particle swarm optimization for parallel machine total tardiness scheduling," International Journal of Advance Manufacturing Technology, vol. 49, no. 5, pp. 723-739, Jul. 2010, doi: 10.1007/s00170-009-2426-8.

[21] I. Alharkan, M. Saleh, M.A. Ghaleb, H. Kaid, A. Farhan, and A. Almarfadi, "Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server," Journal of King Saud University – Engineering Sciences, vol. 32, no. 5, pp. 330-338, Jul. 2020, doi: 10.1016/j.jksues.2019.03.006.

[22] J. Kennedy, and R. Eberhart, "Partical swarm optimization, " in Proc. of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol. 4, doi: 10.1109/ICNN.1995.488968.

[23] R. Storn, and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, Jan. 1997, doi: 10.1023/A:1008202821328.

[24] A.P. Guerreiro, C.M. Fonseca, and L. Paquete, "The hypervolume indicator: computational problems and algorithms," ACM Computing Surveys, vol. 54, no. 6, pp. 1-42, Jul. 2021, doi: 10.1145/3453474.

[25] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: theoretical foundations and practical implications," Theoretical Computer Science, vol. 425, pp. 75-103, Mar. 2012, doi: 10.1016/j.tcs.2011.03.012.

[26] Y. Yusoff, M.S. Ngadima, and A.M. Zain, "Overview of NSGA-II for Optimizing Machining Process Parameters," Procedia Engineering, vol. 15, pp. 3978-3983, Dec. 2011, doi: 10.1016/j.proeng.2011.08.745.

[27] G.V. Rossum, "Python tutorial, Technical Report CS-R9526," in Amsterdam: Centrum voor Wiskunde en Informatica (CWI), Amsterdam, Netherlands, 1995.

[28] PyCharm. (2020). JetBrains. Available: Available: http://jetbrains.com/pycharm

[29] F. Biscani, and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," Journal of Open Source Software, vol. 5, no. 53, pp. 2338-2350, Sep. 2020, doi: 10.21105/joss.02338.