Particle Swarm Optimization to Solve Unrelated Parallel Machine Scheduling Problems

F Pulansari and M D R Triyono

1Department of Industrial Engineering, Faculty of Engineering, Universitas Pembangunan Nasional “Veteran” Jawa Timur, Surabaya, 60294, Indonesia
*corresponding author: pulansari@gmail.com

Abstract. The problem of unrelated parallel machines scheduling is very important in this industry. Scheduling is useful to save company resources, one of which is in terms of time. With minimization of completion time, companies can fulfill it quickly and precisely. Focuses on unrelated parallel machine scheduling problems that depend on sequences aimed at minimizing total turnaround time by considering setup time. This paper presents how unrelated parallel machine scheduling using the particle swarm optimization algorithm approach. The experimental results obtained indicate the optimum value.

Key word: Particle Swarm, Optimization, Scheduling, Parallel Machine

1. Introduction

In a real industrial environment, unplanned events and unexpected incidents can occur at any time [1]. The arrival of random jobs that often occur in manufacturing practices can create the need for dynamic scheduling [2]. Unexpected scheduling allows unexpected interruptions to be considered. According to Zhao [3], scheduling problems have an important role in the manufacturing industry. Parallel machine scheduling (PMS) is an issue identified with distributing a collection of occupations to various machines to satisfy client demands. Torabi argues [4] that the study of PMS can generally be categorized into three classes: uniform, identical, and unrelated parallel scheduling machine problems. From these classifications, the issue of scheduling unrelated parallel machines is an oversimplification of two different classifications, where various machines carry out a similar task although own distinctive capacities or capabilities. Nevertheless, all things considered, the scheduling of unrelated parallel machines is a trial for analysts and experts, because of the problem of NP-hard to minimalist makespan for various employments on unrelated parallel machines [5]. Some research also presents the unrelated parallel machine scheduling (UPMS) problem that involves makespan considerations [4], [6], [7]. This study emphases on the scheduling parallel machines problem that are not related by considering the inherent uncertainty in processing time and setup time.

Inspired by a genuine contextual analysis at PT.X plastic factory in Surabaya, this research is concerned with various employments scheduling on a parallel machine system that is not related to constraints of secondary resource, in which each activity must be handled if the machine and other secondary resources (for example, power work, equipment, and so forth.) accessible. Every task owns a deadline as well as necessitates one operation with a non-zero ready time. In addition, at the point when various occupations are gotten from clients competing for a similar resource (for example, a set of unrelated parallel machines), besides agreeing a particular deadline, it is usually to determine the level of importance or weight on the basis of the type and work. Relationships exist between customers and producers. Besides, the installation period needed for an occupation on a machine depends on the similarity level or degree of inequality that are present among this occupation and the immediate
previous work. Thus, the problem we consider here is scheduling unrelated parallel machines that depend on sequences that aim to minimize the total turnaround time or so-called makespan by considering setup time. Chang et al., [8] has proved that the problem of minimization of a single machine-weighted delay related to makespan, a total burdened with the release of static work and the availability of static machines and the weight of all work alike, is very NP-hard. Therefore, the problem that will be researched in this writing is likewise very NP-hard. As classified by Amiri and Khammohammadi [9], the methods that would be proposed to resolve the issue into two categories as a classical and intelligent algorithm (artificial intelligence). Thus, Torabi [4] argues that it is not possible that a polynomial time algorithm is able to be built up that can decide the ideal answer for UPMS problems to such a related issue in its practice. Therefore, several experts applied a meta-heuristic method to such problems (see [5], [10]–[13]). Therefore, we suggest a Particle Swarm Optimization (PSO) algorithm in resolving related problems. Singh [14] argues that PSO is an effective algorithm that delivers quality solutions in sensible computing period and needs a few parameters to be set up compared to other evolutionary meta heuristic. On the other side of Lin [5] says that computing results using PSO outperforms the existing meta heuristic and is very accurate.

2. Problem formulation parallel machine scheduling
Scheduling could be deciphered as apportioning various resources to play out various duties or activities inside a specific time frame and is a procedure of decision-making that is critical in the industry and services of manufacture that allocate existing resources in order to optimize the objectives and objectives of the company [15]. According to Pinedo [16], UPMS is scheduling N jobs on parallel machine unrelated extension of parallel non-identical. There is a parallel machine m, where the machine i to process a task j, then the machine speed is Vj.

In the case of PMS, a collection of independent work must be planned in the machine with no preemption. The order-dependent setup time is also considered in this paper. This relates to arrangement period at the time the machine changes production starting with one task then onto the next. As for several symbols, they are defined, namely: n number of jobs to be processed; m number of machines; i jobs’ index; k machines’ index; Cmax the jobs’ maximum completion time, e.g., makespan; Ci the job completion time i; pij the job processing time i on machine k; STijk the setup times of processing job j right after job i on machine k; Xijk a binary variable that is equal to 1 if job j is right after job i on machine k, 0 otherwise; X0ijk a binary variable that is equal to 1 if job j is the first one on machine k, 0 otherwise; V a very large constant. The assumption used is that each machine could only processes one job at a time, and each job could only be done once. In the matter of the formulation the researcher refers to (see[17], [18]).

\[
\text{Minimize } \sum_{i=1}^{n} C_i \\
\sum_{i=0}^{n} \sum_{j=1}^{m} X_{ijk} = 1, \quad \forall j = 1, 2, ..., n, \quad \forall k = 1, 2, ..., m, \quad (1)
\]

\[
\sum_{i=0}^{n} X_{ijk} - \sum_{j=0}^{m} X_{hjk} = 0, \quad \forall h = 1, 2, ..., n; \quad \forall k = 1, 2, ..., m, \quad (2)
\]

\[
C_j \geq C_i + \sum_{k=1}^{m} X_{ijk} (S_{ijk} + P_{ijk}) + V (\sum_{k=1}^{m} X_{ijk} - 1), \quad \forall i = 0, 1, ..., n; \quad \forall j = 1, 2, ..., n, \quad (3)
\]

\[
\sum_{j=0}^{m} X_{0jk} = 0, \quad \forall k = 1, 2, ..., m, \quad (4)
\]

\[
X_{ijk} \in \{0, 1\}, \quad \forall i, j, k = 0, 1, ..., n; \quad \forall k = 1, 2, ..., m, \quad (5)
\]

\[
C_0 = 0, \quad (6)
\]

\[
C_j \geq 0, \quad \forall j = 1, ..., n. \quad (7)
\]

Equations (1) are objective functions. Equation (2) guarantees that each job must be processed just a single time. Equation (3) suggests that each work has a predecessor and successor. Note that, Ji job is utilized to introduce the first job on the machine. Equation (4) ensures that the job processing must start after the completion of its predecessor. Equation (5) ensures that each machine has only one job during
the process. Equation (6) characterizes binary variables. Equation (7) suggests that the completion time for a job is zero. Equation (8) guarantees that the completion time of regular work is not negative.

3. Particle swarm optimization-based scheduling

The PSO began with a population consisting of a number of particles (stating proposed solutions) randomly generated. Next is an update of the position and speed of each particle in an iterative manner to produce a new, better solution. Torabi [4] also said that in PSO, each particle defines its speed in the basis of its own past experience. That experience comes from particles as individuals and members of the whole population. The speed of each individual is renewed by reference to the best individual discovered so far by the horde (gbest), and the best individual discovered earlier by that individual (pbest). The PSO algorithm used refers to [19], [20]:

\[ v_i(t) = v_i(t-1) + \varphi C_1 (p_i - x_i(t-1)) + \varphi C_2 (g - x_i(t-1)) \]  
\[ x_i(t) = x_i(t-1) + v_i(t) \]  

The PSO equation above can be explained as follows: \( v \) is the velocity of particle; \( x \) is the particle position; \( \varphi \) is a random vector distributed with a value limit \([0 - 1]\); \( C_1 \) is a constant factor of cognitive learning; \( C_2 \) is a constant factor of social learning; \( p \) is the best position of particle; \( g \) is the global best position; \( i \) is the index of particle; and \( t \) is an index of iteration.

4. Method

The procedure in this study begins by identifying problems, determining the type of work, determining the problems of the scheduling system and learning the concepts of scheduling, analyzing system requirements, applying the PSO algorithm as a system test. Determination of the relevant PSO algorithm refers to [4], [19]. Retrieval data is done by the method of observation, interviews, and literature study. Sampling of data based on the characteristics determined in accordance with the research problem. In this case, the data is taken is data from PT. X in Surabaya in January 2020 with seven kinds of products namely 1000cc bottle caps, pet fertilizer caps, jerry can cap 5l, clock frames, clock bodies, basketball seashells, 500cc m caps. The data needed were in the form of processing time data and setup time data.

In the plastic production process PT. X each job to be scheduled has a different processing time on each machine. The setup time varies from one job to another. Below is a table of setup and processing times for each job to be scheduled:

| No. | Product name | Machine processing time (hours) |
|-----|--------------|---------------------------------|
|     |              | M1  | M2  | M3  |
| 1.  | Bottle caps 1000cc | 60  | 44  | 41  |
| 2.  | Pet fertilizer caps | 72  | 53  | 51  |
| 3.  | Jerry can caps 5 l | 69  | 66  | 48  |
| 4.  | Clock frame     | 166 | 166 | 161 |
| 5.  | Clock body      | 166 | 166 | 161 |
| 6.  | Seashell basket | 103 | 103 | 103 |
| 7.  | Caps M 500cc   | 71  | 53  | 50  |
| 8.  | Bottle caps 1000cc | 89  | 65  | 61  |
| 9.  | Pet fertilizer caps | 87  | 63  | 61  |
| 10. | Clock body     | 146 | 146 | 142 |
| 11. | Clock frame    | 146 | 146 | 142 |
| 12. | Caps M 500cc   | 82  | 62  | 58  |
| 13. | Jerry can caps 5 l | 83  | 80  | 58  |
| 14. | Seashell basket | 146 | 146 | 146 |
| 15. | Pet fertilizer caps | 69  | 50  | 48  |

| Table 2. Setup time |
|---------------------|
| From/Job (hours)    |
| To  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
|     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |
Scheduling using PSO algorithm is calculated using the help of MATLAB software where the test results are obtained from the running coding results that have been made.

The stages of the PSO method are as follows:

1. Determination of initial parameters.
   In this research, the followings are the initial parameters used: Iterations (i) of 100; Particles (i) of 30; Factor of cognitive learning (C_1) of 1; factor of social learning (C_2) of 2; and Velocity (v) of 1.

2. Initialization
   In this study the initial particle positions or job sequences are made in sequence and placed on the first machine. After the job is scheduled as in the order above then the makespan value is calculated. The makespan value obtained is then used as the initial best solution and the job sequence is used as the p_{best} value and the initial g_{best} value.

3. Solution Search
   The search for a solution begins with determining the particles’s best position as well as the best global position. The position was then used to calculate velocity. Here is an example of how velocity is calculated:
   
   
   \[ v_i(t) = v_i(t-1) + \varphi \cdot C_1 \cdot \left( p_i - x_i(t-1) \right) + \varphi \cdot C_2 \cdot \left( g - x_i(t-1) \right) \]
   
   \[ v_i = 1 + 0.6352 \cdot 2(1-1) + 0.8367 \cdot 2(1-1) = 1.2 \]
   
   \[ v_i = 1 \]

   after velocity is calculated then the position of the particles is renewed
   
   \[ x_i = x_i(t-1) + v_i(t) \cdot x_i = 1 + 1 = 2 \]

   Each rounding position value is rounded. All position values less than one are rounded up to one and position values that are more than the number of machines will be rounded according to the number of machines. This is done until all work is completed scheduled by each particle.

4. Determination of the best global
   In the previous stage it has been explained that the particle schedules all jobs until a new scheduling sequence is obtained by each particle. The results of the new solution are contrasted with the old value, if the new value is smaller than the solution is updated. This continues until all the termination criteria are met.

5. Check Criteria
   The last stage in the process of scheduling employing the PSO algorithm is checking criteria.

5. Result and discussion
   At this stage a graph is presented to show the results of the PSO algorithm using the MATLAB program and for more details can be seen in the figure below:
Based on the picture above, the best makespan value obtained from the fitness value in each iteration is indicated by a blue line where from iteration 1 to iteration 2 is constant and the graph goes down to iteration 3 and is constant until iteration 5 then falls again during iteration 6 then on iteration 7 goes down to 8 then on iteration 9 goes down again and on iteration 10 goes down to iteration 17 then on iteration 18 and so on is constant, it indicates that from iteration 1 to iteration 17 it still changes but after iteration 18 to iteration 100 values makespan remains constant or unchanged. From the picture above it can also be seen that the determination of the criteria for when to stop to get the best makespan value is at the time of iteration 18 with a makespan value of approximately 472 hours.

Based on the results of running that can be seen on the first engine in the sequence J4-J10-J11, the second engine in the order J1-J5-J6-J14, the third engine in the order of J2-J3-J8-J9-J12-J13-J15. The total value of completion of the first engine is 469 hours, the value of completion of the second engine is 472 hours, and the value of completion of the third engine is 470 hours. This indicates that UPMS problems can be solved by a meta-heuristic approach using PSO algorithm with an optimal makespan value of 472 hours for fifteen jobs on three machines scheduled at PT X.

6. Conclusion
PSO algorithm succeeded in solving UPSMP with an optimal makespan value of 472 hours, where the first machine did the work in the J4-J10-J11 work order, the second machine in the J1-J5-J6-J14 work order, and the third machine in the J2-J3- work sequence J7-J8-J9-J12-J13-J15. That way UPSMP with the PSO algorithm is able to assist companies in completing orders fully on each machine. Equitable distribution of work done on each machine is also able to minimize turnaround time, this increases savings in resources such as energy, labor, to production costs. From this the PSO algorithm are able to be used to resolve similar scheduling issues. This PSO algorithm can also be compared with other meta-heuristic methods, such as genetic algorithms, simulated annealing, ant colony optimization and so on. The author hopes this algorithm can be developed further so that it has a new mechanism in generating solutions.

7. References
[1] Nouiri M Bekrar A Jemai A Trentesaux D Ammari A C and Niar S, Oct. 2017 Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns Computers and Industrial Engineering 112 p. 595–606.
[2] Wang Z, Zhang J, and Yang S, Dec. 2019 An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm and Evolutionary Computation* 51.

[3] Zhao F, Qin S, Yang G, Ma W, Zhang C, and Song H, Jul. 2019 A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. *Expert Systems with Applications* 126 p. 41–53.

[4] Torabi S A, Sahebjamnia N, Mansouri S A, and Aramon Bajestani M, 2013 A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing Journal* 13 p. 4750–4762.

[5] Lin Y K, 2013 Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates. *Mathematical Problems in Engineering*.

[6] Kashan A H and Karimi B, Feb. 2009 A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers and Industrial Engineering* 56, 1 p. 216–223.

[7] Kamath S, Dec. 2011 Unrelated parallel machine scheduling—perspectives and progress. *OPSEARCH* 48, 4 p. 318–334.

[8] Chang P C, Chen S S, and Fan C Y, Jan. 2008 Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing Journal* 8, 1 p. 767–777.

[9] Amir M and Khanmohammadi S, Feb. 2013 A primary unit commitment approach with a modification process. *Applied Soft Computing Journal* 13, 2 p. 1007–1015.

[10] Caniylmaz E, Benli B, and Ilkay M S, Apr. 2015 An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date. *International Journal of Advanced Manufacturing Technology* 67, 9–12 p. 2105–2115.

[11] Lin C W, Lin Y K, and Hsieh H T, Jul. 2013 Ant colony optimization for unrelated parallel machine scheduling. *International Journal of Advanced Manufacturing Technology* 67, 1–4 p. 35–45.

[12] Amaral G, Costa L, Rocha A M A C, Varela L, and Madureira A, 2020 Application of the Simulated Annealing Algorithm to Minimize the makespan on the Unrelated Parallel Machine Scheduling Problem with Setup Times in *Advances in Intelligent Systems and Computing* 923 p. 398–407.

[13] Lee J H, Yu J M, and Lee D H, Dec. 2013 A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: Minimizing total tardiness. *International Journal of Advanced Manufacturing Technology* 69, 9–12 p. 2081–2089.

[14] Singh M R and Mahapatra S S, Mar. 2016 A quantum behaved particle swarm optimization for flexible job shop scheduling. *Computers and Industrial Engineering* 93 p. 36–44.

[15] Baker K R and Trietsch D, 2009 *Principles of Sequencing and Scheduling* Hoboken, NJ, USA: John Wiley and Sons.

[16] Pinedo M L, 2016 *Scheduling Theory, Algorithms, and Systems (5th ed.)*.

[17] Guinet A, 1993. Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research* 31, 7 p. 1579–1594.

[18] Wang L, Wang S, and Zheng X, 2016. A Hybrid Estimation of Distribution Algorithm for Unrelated Parallel Machine Scheduling with Sequence-Dependent Setup Times.

[19] Kennedy J and Eberhart R, 1995 Particle swarm optimization in *Proceedings of ICNN’95 - International Conference on Neural Networks* 4 p. 1942–1948 vol.4.

[20] Shi Y and Eberhart R, 1998 Modified particle swarm optimizer in *Proceedings of the IEEE Conference on Evolutionary Computation*, *IEEE* p. 69–73.