Metaheuristics for Scheduling Unrelated Parallel Machines with Sequence-Dependent Setup Time and Machine Eligibility

Karn Moonsri, Kanchana Sethanan* and Chatnugrob Sangsawang

Research Unit on System Modeling for Industry, Department of Industrial Engineering Department, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

**Corresponding author. E-mail: skanch@kku.ac.th*

ABSTRACT

This research focuses on the parallel machines scheduling problem in the hard disk drive industry, with the objective of minimizing the makespan. In this problem, unrelated parallel machines with sequence-dependent setup time and machine eligibility are considered. This problem can be formulated as a deterministic $R_m|M_j,S_{jk}|C_{max}$ problem. To solve the problem, a mathematical model is developed for small-sized problems. For large-scale problems, algorithms based on a constructive heuristic (Unrelated Parallel Machines with Sequence-Dependent Setup Times and Machine eligibility (UPSDSTM)) and Differential Evolution (DE) are applied. The results demonstrate that the DE algorithm is very efficient. It yields higher relative improvement (RI) on the makespan.

Keywords: Heuristic algorithm, Makespan, Differential evolution algorithm

INTRODUCTION

Scheduling is crucial to production productivity. With highly competitive industries, efficient scheduling approaches are needed to maximize the efficiency of the production system to compete. Generally, production scheduling is extremely complicated, especially for unrelated parallel machines scheduling, and has various objectives, such as to determine the makespan, maximize the output, or minimize the total cost.

This study focuses on the hard-disk drive assembly process for the Head Gimbal Assembly (HGA), which assembles the head gimbal and suspension in order to join to other components. When the reader is made on an assembly line, the magnetic system is tested by a tester, which needs to be converted, if the previous product tested was different. Normally, converting the tester settings takes a long time to change the program, and this time is the sequence-dependent setup time (SDST). Machine eligibility is also a factor, because some test programs cannot be run on some machines, and the duration of machine setup is based on the job. This problem can be formulated as a deterministic $R_m|M_j,S_{jk}|C_{max}$ problem. Due to the diversity of product features, testing programs, and testers, scheduling

the testers to test the products is very challenging. Hence, inefficient production scheduling may cause long setup times on the tester machines. Consequently, significant capacity losses can occur due to the changeover time of a tester to enable it to work fully with the testing programs and products, which results in low production efficiency and late customer delivery.

Researchers have developed many different methods over the past few decades to solve the scheduling problem. Several methods, such as branch and bound algorithms, have been proposed to find an optimal solution. However, the two-identical-machines scheduling problem with minimization of the makespan is NP-hard (Garey and Johnson, 1979). Thus, the more complex scheduling problem with m unrelated parallel machines with SDST and machine eligibility with minimization of the makespan ($R_m|M_j,S_{jk}|C_{max}$) is also NP-Hard. Exact methods can solve only small-sized problems. Therefore, researchers have recently considered solving large-scale parallel machines scheduling problems using heuristics and metaheuristics algorithms, in order to obtain approximate solutions. Hence, this paper uses a well-known metaheuristic called the Differential Evolution algorithm (DE), an evolutionary algorithm, to solve the problem and improve the solution obtained in tractable time.

LITERATURE REVIEW

Researchers have been interested in the problem of production scheduling in a parallel machine (PMSP) for a long time. Dealing with real-life PMSP is a major challenge for both researchers and manufacturers. Different features of the scheduling and sequencing problems in PMSP depend on the requirements of the problems in practice, such as sequence dependent setup times (SDST), machine eligibility, and types of machines in the parallel production system.

Machine setup time, especially SDST, is an important factor for production scheduling in all production systems. Recently, various researchers have studied the features of SDST in PMSP. It is generally difficult to solve an PMSP with SDST (Behnamian, 2015). Various approaches have been proposed, for example, the constructive heuristic by Guinet (1993), the Tabu search algorithm by Franca et al. (1996), and simulated annealing by Longendran et al. (2007). Other approaches to solve the PMSP with SDST problem can be found in Lee and Pinedo (1997), Weng et al. (2001), and Kim et al. (2002).

Since a number of parallel machines scheduling problems have been proven to be NP-hard, many researchers have considered solving large-scale parallel machines scheduling problems using meta-heuristic algorithms in order to find approximate solutions. Among them, Arnaout et al. (2010) proposed a two-stage ant colony optimization (ACO) algorithm improved by a local search procedure, with the objective to minimize the makespan for the unrelated parallel machines scheduling problem with SDST and machine-dependence, when the ratio of the number of jobs to the number of machines is large. In 2009, Behnamian et al. proposed a hybrid meta-heuristic for the minimization of makespan in scheduling problems with parallel machines and sequence-dependent setup times. They

developed a hybrid of population-based evolutionary searching ability of ant colony optimization (ACO), simulated annealing (SA) for solution evolution, and a variable neighborhood search (VNS) heuristic and comparison for near-optimal solutions.

Recently, Wang et al. (2013) investigated parallel machine scheduling with job splitting. They applied the differential evolution metaheuristic as a solution approach, due to its effectiveness. Additionally, a new crossover method and a new mutation method are brought forward in the global search procedure for solving the job splitting constraint. Ruiz-Torres et al. (2013) developed algorithms and simulated annealing meta-heuristics for practical-sized problems of the unrelated parallel machines scheduling problem with deteriorating effect for minimizing the makespan. Cappadonna et al. (2013) developed a mixed integer linear programming (MILP) model for optimally solving the unrelated parallel machines scheduling problem with limited human resources, and developed a genetic algorithm (GA) for minimizing completion time. Edis et al. (2013) explained the components for parallel machines production scheduling in order to check and analyze related problems on parallel machines scheduling, to which resources were added, and estimate the results for further study by analyzing the strengths and weaknesses of studies in the literature, focusing on the machine environment, increase of resources, objective function, complication, solution, and importance of each literature report.

Given the successes of DE in solving various problems and its attractive features, this paper proposed this method for solving unrelated parallel machines with SDST and machine eligibility with the minimization of the makespan ($R_m|M_j,S_{jk}|C_{max}$). DE was first introduced by Storn and Price (1997) and is one of the most powerful techniques effectively applied for continuous optimization. DE has been successfully applied in several fields, such as production scheduling (see Wisittipanich and Kachitvichyanukul (2011), Chakaravarthy et al. (2013)), simple assembly line balancing (see Pitakaso and Sethanan (2015)), scheduling vehicle problems (see Erbao (2008), Erbao and Mingyong (2009), Lai and Cao (2010), Chen et al. (2010), Sun et al. (2012), Dechampai et al. (2015)), and manufacturing problems (see Nearchou (2006), Weaver et al. (2012). To the best of our knowledge, no metaheuristic algorithm has been applied to unrelated parallel machines with SDST, and machine eligibility with minimization of the makespan ($R_m|M_j,S_{jk}|C_{max}$). Therefore, this paper applies DE to solve this problem.

PROBLEM STATEMENT

This study focused on the scheduling problem for the testing process with *n* products in the hard disk drive industry. In this problem, 'n' jobs (j = 1, 2, 3…n) are produced on a single machine, while the workstation consists of 'm' unrelated machines (m = 1, 2, 3…M) in parallel. Each job can be processed by only one machine, and the processing times of jobs on different machines are not related (i.e., unrelated machines). In addition, the Machine eligibility (M_j) is considered, since not all machines are capable of processing job j. That means the machines (i.e., testers) may be used for testing using only certain programs from a total of

Pg programs. For example, test program 1 can only run on test machines 1 and 2, and test program 2 can only run on machine 3, so if any job needs to be tested with test program 2, only machine 3 can be used, as shown in Figure 1. When changing from one program to another, a setup time SDST for each machine is required for conversion between any two programs. The processing time used for testing each product depends on both the machine and the test programs.

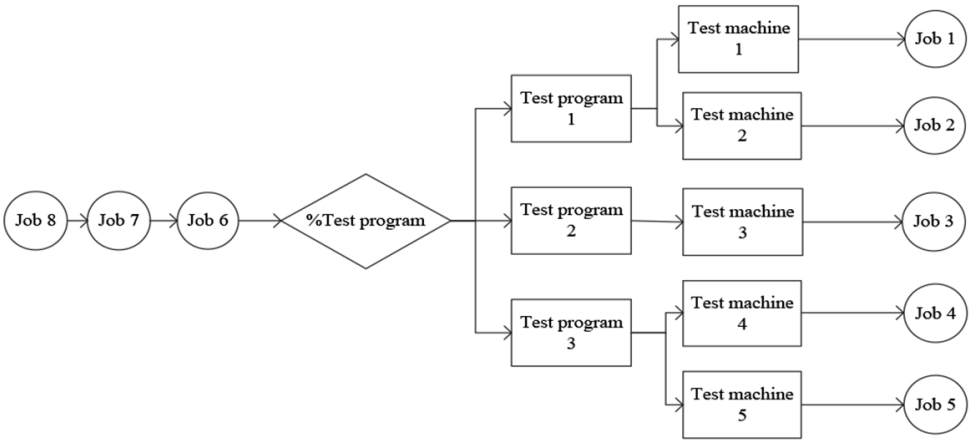

Figure 1. Characteristics of the problem.

MATHEMATICAL AND PROPOSED SOLUTION METHODS

A mixed integer programming (MIP) formulation was developed for solving the unrelated parallel machines with SDST and machine eligibility problem with minimization of the makespan ($R_m|M_j,S_{jk}|C_{max}$). To solve the problem, a mixed integer linear programming model was formulated. Parameters and decision variables used in formulating the model were defined. The model is presented with a brief explanation of each constraint.

Indexes

j,k	Product index j, k = 1, 2, 3, …, n
g,f	Test program index g, f = 1, 2, 3, …, Pg
i	Machine index i = 1, 2, 3, …, M

Parameters

n	The number of products
m	The number of machines
Pg	The number of test programs
$Pt_{(i,j,g)}$	The processing time on a machine i, for product j, and program test g
$Stp_{(i,j,k,)}$	Sequence dependent setup time for test program on machine i when processing job j after having processed job k
$M_{(j)}$	Set of machines performing job j

G A large enough number

Decision variables

C_{max} Makespan

$Ct_{(i,j,g)}$ Completion time for product j test program g on machine i

Binary variables

$X_{(i,j,g)}$ = 1, if product j and test program g assigned to machine i and 0 otherwise

$Y_{(i,j,k,g)}$ = 1, if product j test program g assigned to machine i before product k test program g to machine i and 0 otherwise

$U_{(0,k,i)}$ = 1, if product k assigned on machine i is first job and 0 otherwise

$W_{(0,j,i)}$ = 1, if product j on machine i is last job and 0 otherwise

Mathematical formulation

Objective function:

$$Minimize\ C_{max} \tag{1}$$

Subject to:

$$C_{\max} \geq Ct_{(i,j,g)} \quad ;\forall i,j,g \tag{2}$$

$$Ct_{(i,j,g)} \geq Pt_{(i,j,g)} * X_{(i,j,g)} \quad ;\forall j,g\ \exists i \in M_j \tag{3}$$

$$Ct_{(i,j,g)} - Ct_{(i,k,g)} + G*(1-Y_{(i,j,k,g)}) \geq Stp_{(i,j,k)} + Pt_{(i,j,g)} * X_{(i,j,g)} \quad ;\forall i,j,g \tag{4}$$

$$\sum_{i=1}^{m} X_{(i,j,g)} = 1 \quad ;\forall j,g \tag{5}$$

$$X_{(i,j,g)} - U_{(0,k,i)} - \sum_{j}^{n} Y_{(i,j,k,g)} = 0 \quad ;\forall i,k,g \tag{6}$$

$$X_{(i,j,g)} - W_{(0,k,i)} - \sum_{j}^{n} Y_{(i,j,k,g)} = 0 \quad ;\forall i,k,g \tag{7}$$

$$\sum_{k=1}^{n} U_{(0,k,i)} = 1 \quad ;\forall i \tag{8}$$

$$\sum_{j=1}^{n} W_{(0,j,i)} = 1 \quad ;\forall i \tag{9}$$

$$C_{\max}, Ct_{(i,j,g)}, Pt_{(i,j,g)} \geq 0 \tag{10}$$

$$X_{(i,j,g)}, Y_{(i,j,k,g)} \in \{0,1\} \tag{11}$$

The objective function (1) is to minimize the maximum completion time or makespan. Constraints (2) ensure that the makespan is equal to or greater than the completion time of the products. Constraints (3) ensure that the completion time of the products on every machine is equal to or greater than the processing time of the product. Constraints (4) are to control the completion times of the products at the machines. Basically, if a product j is assigned to a machine i after product k, its completion time $C_{(i,j,g)}$ must be greater than the completion time of i, $C_{(i,k,g)}$, plus the setup time between k and j and the processing time of k. If $Y_{(i,j,k,g)} = 0$, then the big constant G renders the constraint redundant. Constraints (5) ensure that every job is assigned to exactly one machine. Constraints (6) ensure that every job, except for the first job on a machine, must be immediately preceded by exactly one different job. Constraints (7) ensure that every job, except for the last job on a machine, must be immediately followed by exactly one product. Constraints (8) - (9) show that a machine can have exactly one first and one last product. Constraints (10) - (11) are the basic restrictions on the decision and binary variables.

Mathematical models are only suitable for solving small problems. Using such a model to find the optimal solution for a large-sized problem would be difficult or time-consuming, because a massive problem, such as a production sequence problem, is too complicated. The essential limitation is the technological limitation to finding the solution. The cause of the inefficiency in finding an optimal solution may lie in the limitation of variables.

Constructive heuristic

A heuristic is a process for finding solutions without rules, as well as testing the decision by trial and error, which is the guide for finding a solution. Finding solutions through heuristics is different from other methods, because heuristics do not take into account any possible solutions; instead, they choose the solution that is suitable for the method. This kind of process assists finding solutions from a large and complicated set of data. However, heuristics still have disadvantages; the solution from such a method is only a near-optimal solution. Thus, the present study aimed to develop an algorithm – unrelated parallel machines with sequence dependent setup times and machine eligibility (UPSDSTM) – to resolve the massive problem and gain a near-optimal solution. The key idea to developing the UPSDSTM algorithm is that scheduling job j with the shortest processing time to machine i results in minimizing the completion time of that job. Then, the load of machines is balanced. Finally, the next best rule is performed in order to minimize the dependent setup time of jobs on each machine. The details of UPSDSTM and the variables and symbols used to construct UPSDSTM are defined below:

j,k	Product index
g,f	Test program index
i	Machine index
N	Set number of products
MC	Set number of machines

ME	Set number of machines eligibility
$ST_{(i,j,k)}$	Set setup time of test program on machines i
$PT_{(i,j,g)}$	The processing time for product j test program g on machine i
$C_{(i,j,g)}$	Completion time for product j test program g on machines
C_{max}	Makespan MAX $\{C_{(i,j,g)}\}$

The UPSDSTM algorithm can be divided into seven stages. Details are given below:

Step 1: Consider overall machine needs by selecting the machine with the minimum test time, which is m'; $m' = MIN\{PT_{(i,j,g)}\}$, then follow step 2.

Step 2: Calculate the completion time of each job, $C_{(i,j,g)} = PT_{(i,j,g)} + ST_{(j,k)}$, and choose $MAX\{C_{(i,j,g)}\}$ for its makespan, then follow step 3.

Step 3: Update the data for sequence of set $MC_i = \{j,k \quad N\} \cap ME_j$; if job cannot be distributed into machines because of machine eligibility for choosing forward m', then follow step 4.

Step 4: Consider overall ready time on every machine and choose minimal ready time on machine set as $m'' = MIN\{C_{(i,j,g)}\}$, then follow step 5.

Step 5: Shift job from MC_i to $MAX\{C_{(i,j,g)}\}$ by choosing job for $MIN\{PT_{(i,j,g)}\}$ of m'', and calculate the completion time. If it decreases makespan, then back to step 3. If not, then follow step 6.

Step 6: Swap sequence of each machine by considering $MIN\{ST_{(j,k)}\}$ using the nearest neighbor algorithm, then follow step 7.

Step 7: Calculate the completion time of makespan as $C_{max} = MAX\{C_{(i,j,g)}\}$.

Metaheuristic development

For this case study, the performance of the constructive heuristic or UPSDSPM Algorithm may not be good enough to solve the problem. Because the heuristic algorithms we developed do not have flexibility or appropriate estimation of values, we find near optimal solutions. We use metaheuristics, which are independent of the problems, to find optimal solutions, and then apply them to complex and large problems. The meta-heuristic is developed to enhance the solution obtained in a tractable time. Therefore, a metaheuristic based on the original differential evolution algorithm (DE) is developed. The DE algorithm was first proposed by Storn and Price (1997), and it has proven to be an effective algorithm to solve both continuous and discrete optimization. It has four fundamental steps: generation of initial solution, mutation operation, recombination operation, and selection operation.

1. Begin
2. $t \leftarrow 0$;
3. Generate initialization vector $x_{i,G}$ real-valued encoding [0, 1)
4. While (Not termination); looping
5. Mutation
6. Crossover and Recombination
7. Evaluate for decoding rank order value (ROV) and randomize sequence into machines
8. Selection
9. $t \leftarrow t + 1$;
10. End while
11. End

Generate initial solution. Set the generation number G = 0, $P_G = \{x_{1,G},\ldots x_{NP,G}\}$ with $x_{1,G} = \{x^1_{i,G},\ldots x^D_{i,G}\}$, i=1..NP. Uniformly distribute in the range $[x_{\min}, x_{\max}]$, where $\{x^1_{\min},\ldots,x^D_{\min}\}$ and $x_{\max} = \{x^1_{\max},\ldots,x^D_{\max}\}$.

Mutation operation. Each of the N parameter vectors undergoes mutation and recombination. Mutation expands the search space. For a given parameter vector $X_{i,G}$, randomly select three vectors $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, such that the indices I, r_1,r_2,…r_3 are distinct. Add the weighted difference of two of the vectors to the third $V_{i,G+1} = X_{r1,G}+F(X_{r2,G} - X_{r3,G})$. The scaling factor F is a constant from [0, 2] $V_{i,G}$ and is called the mutant vector.

Recombination Operation. Recombination incorporates successful solutions from the previous generation. The trial vector $U_{i,G+1}$ is developed from the elements of the target vector, $X_{i,G}$, and the elements of the mutant vector, $V_{i,G+1}$. Elements of the mutant vector enter the trial vector with probability CR.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} & if\,(rand(j) \le CR)\,or\,j = rnbr(I) \\ X_{ji,G} & if\,(randb(j) \ge CR)\,or\,j \ne rnbr(I) \end{cases}$$

$rand_{ji}$ U[0,1], I_{rand} is a random integer from [1,2,…,D]. I_{rand} ensures that $V_{ji,G+1} \neq X_{r1,G}$

Selection Operation. The target vector $X_{i,G}$, is compared with the trial vector $V_{ji,G+1}$ and the one with the lowest function value is admitted to the next generation.

In this research, the parameters for the DE algorithm that are used are obtained from the study of Wang et al. (2012). The parameters and their values are as follows: Number of population (NP) = 25, Scaling factor or mutant factor = 2, CR = 0.8 and stops at 300 iterations. The algorithm was run with MATLAB Version 7.7.0.471(R2008b)

The decode method

Decoding sorted the Rank Order Value (ROV) of each vector in ascending order. The new vectors X_i were used to prioritize the product. For example, there are five products and two machines, as shown in Figure 2. The sequence of products is 3-2-1-5-4. The next step is the allocation of products into the machines. The product will be allocated randomly into the machines by following the sequence of products in the ROV steps. Figure 3 shows the results as M_1 = 3-2-4 and M_2 = 1-5

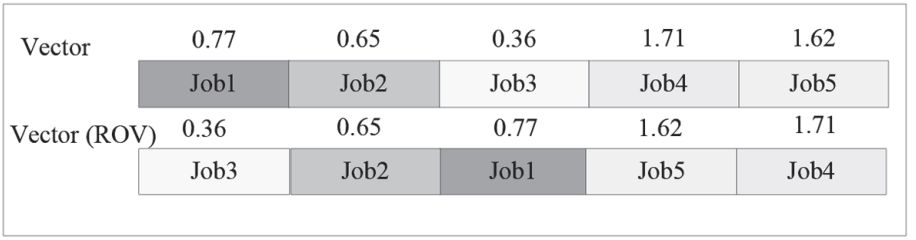

Figure 2. An example of decoding performed by sorting the rank order value (ROV).

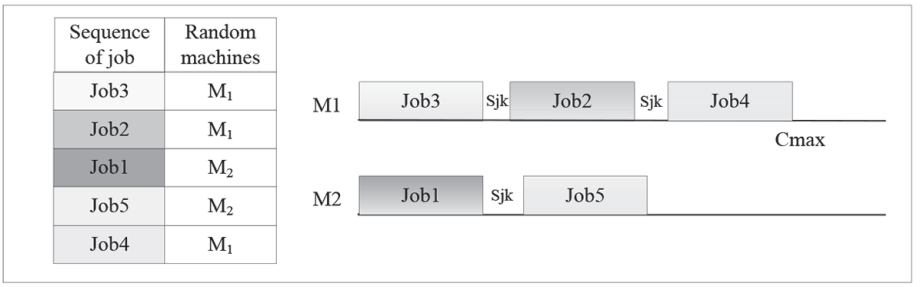

Figure 3. The results allocated randomly into the machines.

RESULTS

This section compares the findings from the UPSDSTM and DE algorithms with the solution from the mathematical model, which we created with five different problem sets: Set 1: 5 jobs, 2 test programs, and 2 machines; Set 2: 8 jobs, 2 test programs, and 2 machines; Set 3: 10 jobs, 2 test programs, and 2 machines; Set 4: 15 jobs, 3 test programs, and 2 machines; and Set 5: 20 jobs, 4 test programs,

and 2 machines. Five test problems were generated for each problem set. Computational results are shown in Tables 1-3 for small-sized problems and Tables 4-5 for large-sized problems. In Tables 1-3, the heuristic performance (HP) is calculated from equation (12), while Tables 4-5 show the relative improvement (RI) of the solutions obtained by the UPSDSTM algorithm with respect to those of the DE algorithm. The RI is determined using equation (13). The size of the test problems was generated based on the practical problem size. The optimal solution was determined using the CPLEX/MPL Modeling System for Windows licensed version, while a constructive heuristic and DE algorithm were developed through the software MATLAB Version 7.7.0.471(R2008b) on Intel® Pentium® 4CPU 2.80GHz PC (1 GB RAM) in order to find the solution.

$$HP = \left(\frac{Optimal}{Sol\_heu}\right)*100 \tag{12}$$

Where HP = the heuristic performance (%)
Optimal = the optimal solution obtained from the mathematical model
Sol_heu = the solution obtained from the algorithms (UPSDSTM and DE algorithms)

$$RI = \left(\frac{Sol\_UPSDSTM - Sol\_DE}{Sol\_UPSDSTM}\right)*100 \tag{13}$$

Where RI = the relative improvement (%) between sol_UPSDSTM and sol_DE
Sol_UPSDSTM = the solution obtained from the UPSDSTM algorithm
Sol_DE = the solution obtained from the DE algorithm

Table 1. Comparison of heuristics algorithm and optimal solution (Number of jobs, number of test programs, number of machines).

Product size (5,2,2)	Optimal solution	UPSDSTM		Original DE	
		Solution	HP (%)	Best	HP (%)
1	627	627	100	627	100
2	764	787	97.08	764	100
3	643	643	100	643	100
4	701	701	100	701	100
5	687	687	100	687	100
Average			99.42		100

Table 2. Comparison of heuristics algorithm and optimal solution (number of jobs, number of test programs, number of machines).

Product size (8,2,2)	Optimal solution	UPSDSTM		Original DE	
		Solution	HP (%)	Best	HP (%)
1	845	922	91.65	884	95.59
2	987	1048	94.18	1014	97.34
3	1467	1612	91.00	1491	98.39
4	856	856	100	856	100
5	761	761	100	761	100
Average			95.37		98.26

Table 3. Comparison of heuristics algorithm and optimal solution (number of jobs, number of test programs, number of machines).

Product size (10,2,2)	Final feasible solution	UPSDSTM		Original DE	
		Solution	HP(%)	Best	HP (%)
1	759	824	92.11	789	96.20
2	827	879	94.08	845	97.87
3	1121	1176	95.32	1156	96.97
4	1163	1249	93.11	1175	98.98
5	990	1087	91.08	1024	96.68
Average			93.14		97.34

Note: *Final feasible solution = the last answer from the software before the program showed not enough memory.

Table 4. Comparison between the UPSDSTM algorithm and DE algorithm for 15 jobs, 3 test programs, 2 m/c.

Product size (15,3,2)	UPSDSTM	Original DE	RI(%)
	Solution	Best	
1	1241	1178	5.08
2	1342	1241	7.53
3	1015	947	6.70
4	1548	1354	12.53
5	1351	1289	4.59
Average	1320.6	1240.6	7.28

Table 5. Comparison between the UPSDSTM algorithm and DE algorithm for 20 jobs, 4 test programs, 2 m/c.

Product size (20,4,2)	UPSDSTM	Original DE	RI(%)
	Solution	Best	
1	1452	1354	6.75
2	1374	1247	9.24
3	1017	954	6.19
4	1249	1154	7.61
5	1478	1357	8.19
Average	1314	1213.2	7.60

DISCUSSION

Production scheduling is important for manufacturing. An efficient scheduling approach is needed to maximize production efficiency. Generally, the type of production scheduling varies, depending on restrictions of the production environment and different objective functions. One of the complexities of parallel machine scheduling is the sequence-dependent setup times for each machine. However, many researchers have considered the problem, including Lee and Pinedo (1997), Maroto (2003), Tahar et al. (2006), Logendran et al. (2007), Behnamian et al. (2009), Ying and Cheng (2010), and Ruiz-Torres et al. (2013). To find the optimal solution of problems involving parallel machines with sequence-dependent setup times, corresponding to Cappadonna et al. (2013) and Tahar et al. (2006), a mixed integer linear programming (MILP) model was developed; this approach applies to small-sized problems, as well. Cappadonna et al. (2013) studied the unrelated parallel machines scheduling problem with limited human resources, with an objective to minimize the makespan. A Mixed Integer Linear Programming (MILP) model for optimally solving the problem was formulated first. Then, a proper genetic algorithm (GA) was presented aiming to cope with larger-sized problems. Average performance of GA on small-sized test cases showed an Average Relative Percentage Deviation of 1.50 and an Average CPU time of 2.87. The GA for large-sized test cases showed an average makespan of 241.7 and average CPU time of 18.8. The computational results of Tahar et al. (2006) showed a general average performance of 4.7% using linear programming for more than 6,000 instances.

In the scheduling unrelated parallel machine with sequence-dependent setup time problem, Behnamian et al. (2009) developed two algorithms hybridized between simulated annealing (SA) and VNS algorithms. Their study revealed that the proposed algorithms have advantages in terms of generality and quality for large instances. In addition, the "Differential Evolution (DE)" method was proposed to compare its solutions with those of PSO and GA. For example, Wang et al. (2013) proposed a DE algorithm to solve the parallel machine with sequence-dependent setup time problem with the objective to minimize the makespan. The results showed that the proposed hybrid-DE (HDE) converged to the global search procedure faster than DE.

In this research, the problem was more complex than those in the literature. The machine eligibility was also considered. To solve the Unrelated Parallel Machines with Sequence Dependent Setup Times and Machine eligibility (UPSDSTM)) problem, we developed a DE algorithm. Based on our results, the average performance ranged between 97.1-100% for the UPSDSTM algorithm and was 100% for the DE algorithm (see Table 1). The average performance of Set 2 was lower than that of Set 1, and ranged between 91.0-100% for the UPSDSTM algorithm and 95.6-100% for the DE algorithm (see Table 2). For Set 3, the average performance was lower than that of Set 1 and Set 2, and ranged between 91.1-95.3% for the UPSDSTM algorithm and 96.2-99.0% for the DE algorithm (see Table 3).

To solve larger-sized problems, the DE algorithm provided better makespan values than the UPSDSTM algorithm in the individual test runs, as shown in

Tables 4 and 5. The relative improvement (RI) obtained was 7.3% with (15, 3, 2) configuration, and a slightly larger RI of 7.6% for the (20, 4, 2) configuration. This can be expected as the quality obtained when applying the DE algorithm to problems with larger numbers of jobs or different numbers of test programs may suffer, thus leaving more room for the DE to improve the solutions. The advantage of DE is that it finds the solution quickly and the optimal solution is acceptable, due to the floating-point numbers used in the solution. In addition, DE is employed as a solution approach due to its distinctive features, and a new crossover method and mutation method are brought forward in the global search procedure. However, there are possibilities to modify the DE algorithm and use other meta-heuristics or hybrid methods to improve the solutions.

CONCLUSION

This study of unrelated parallel machines scheduling with sequence dependent set up times and machine eligibility to minimize completion time compared the efficiency between the UPSDSTM algorithm and the DE algorithm based on the optimal solution and demonstrated that for small-sized problems both of the heuristics can give optimal solutions. When problem size was expanded, the mathematical model took more than two hours to find the solutions, which were only near-optimal solutions, whereas the two heuristics took less than a minute to obtain the near-optimal solutions. Aside from this, with large-sized problems, the mathematical model could not offer any solutions or took an inordinate amount of time. As a result, the UPSDSTM algorithm and the DE algorithm were compared. The DE algorithm improved the solution obtained from the UPSDSTM by 7.3% and 7.6% for problem Sets 4 and 5, respectively. Therefore, it can be assumed that a random search can provide better solutions than a local search and, that if the flexibility of the UPSDSTM algorithm is improved, it will provide better solutions. This UPSDSTM algorithm and DE algorithm can be simply applied for production scheduling without complications. It can shorten production planning time, as well as reduce production costs. However, even though the DE algorithm has demonstrated an outstanding ability to solve the problem at hand, there are possibilities to modify the DE algorithm and use other meta-heuristics or hybrid methods to improve the solutions.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. In addition, this research was supported by the Research Unit on System Modeling for Industry (Grant no. SMI.KKU 2/2558), Khon Kaen University, Thailand.

REFERENCES

Arnaout, J.P., G. Rabadi, and R. Musa. 2010. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. Journal of Intelligent Manufacturing 21(6): 693-701. doi:10.1007/s10845-009-0246-1

Behnamian, J. 2015. Graph colouring-based algorithm to parallel jobs scheduling on parallel factories. International Journal of Computer Integrated Manufacturing. 1-14. doi:10.1080/0951192X.2015.1099074

Behnamian, J., M. Zandieh, and S.F. Ghomi. 2009. Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. Expert Systems with Applications 36(6): 9637-9644. doi: 10.1016/j.eswa.2008.10.007

Cao, E., and M. Lai. 2010. The open vehicle routing problem with fuzzy demands. Expert Systems with Applications 37(3): 2405-2411. doi:10.1016/j.eswa. 2009.07.021

Cappadonna, F.A., A. Costa, and S. Fichera. 2013. Makespan Minimization of Unrelated Parallel Machines with Limited Human Resources. Procedia CIRP 12: 450-455. doi:10.1016/j.procir.2013.09.077

Chakravarthy, S.R., and H.D. Karatza. 2013. Two-server parallel system with pure space sharing and Markovian arrivals. Computers & Operations Research 40(1): 510-519. doi:10.1016/j.cor.2012.08.002

Chen, P., H.K. Huang, and X.Y. Dong. 2010. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Expert Systems with Applications 37(2): 1620-1627. doi:10.1016/j.eswa.2009.06. 047

Dechampai, D., L. Tanwanichkul, K. Sethanan, and R. Pitakaso. 2015. A differential evolution algorithm for the capacitated VRP with flexibility of mixing pickup and delivery services and the maximum duration of a route in poultry industry. Journal of Intelligent Manufacturing 1-20. doi:10.1007/s10845-015-1055-3

Edis, E.B., C. Oguz, and I. Ozkarahan. 2013. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. European Journal of Operational Research 230(3): 449-463. doi:10.1016/j.ejor.2013.02.042

Erbao, C., and L. Mingyong. 2009. A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands. Journal of computational and applied mathematics 231(1): 302-310. doi:10.1016/j.cam.2009. 02.015

Erbao, C., L. Mingyong, and N. Kai. 2008. A differential evolution & genetic algorithm for vehicle routing problem with simultaneous delivery and pick-up and time windows. In Proceedings of the international federation of automatic control 17th world congress (pp. 6-11). doi:10.3182/20080706-5-KR-1001.0778

Franca, P.M., M. Gendreau, G. Laporte, and F.M. Müller. 1996. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. International Journal of Production Economics 43(2): 79-89. doi:10.1016/0925-5273(96)00031-X

Garey, M.R., and D.S. Johnson. 1979. Computers and intractability: a guide to the theory of NP-completeness. 1979. San Francisco. LA: Freeman.

Guinet, A. 1993. Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. The International Journal of Production Research 31(7): 1579-1594. doi:10.1080/00207549308956810

Kim, D.W., K.H. Kim, W. Jang, and F.F. Chen. 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robotics and Computer-Integrated Manufacturing 18(3): 223-231. doi:10.1016/ S0736-5845(02) 00013-3

Lee, Y.H., and M. Pinedo. 1997. Scheduling jobs on parallel machines with sequence-dependent setup times. European Journal of Operational Research 100(3): 464-474. doi:10.1016/S0377-2217(95)00376-2

Logendran, R., B. McDonell, and B. Smucker. 2007. Scheduling unrelated parallel machines with sequence-dependent setups. Computers & Operations Research 34(11): 3420-3438. doi:10.1016/j.cor.2006.02.006

Moonsri, K., and K. Sethanan. 2015. Makespan Minimization for Scheduling Unrelated Parallel Machine with Sequence-Dependent Setup Time. In Toward Sustainable Operations of Supply Chain and Logistics Systems (pp. 253-264). Springer International Publishing. doi:10.1007/978-3-319-19006-8_17

Nearchou, A.C. 2006. Meta-heuristics from nature for the loop layout design problem. International Journal of Production Economics 101(2): 312-328. doi:10.1016/j.ijpe.2005.02.001

Pinedo, M.L. 2012. Scheduling: theory, algorithms, and systems. Springer Science & Business Media. doi:10.1007/978-1-4614-2361-4

Pitakaso, R., and K. Sethanan. 2015. Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types. Engineering Optimization, (ahead-of-print), 1-19.

Price, K., and R Storn. 1997. Differential evolution: a simple evolution strategy for fast optimization. Dr. Dobb's Journal 22:18-24.

Ruiz, R., and C. Maroto. 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169(3): 781-800. doi:10.1016/j.ejor.2004.06.038

Ruiz-Torres, A.J., G. Paletta, and E. Pérez. 2013. Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. Computers & Operations Research 40(8): 2051-2061. doi:10.1016/j.cor.2013.02.018

Sangsawang , C. 2012. Scheduling unrelated parallel machines under machine eligibility constraints in Thailand's hard disk drive industry. Master of Engineering thesis in Industrial Engineering, Graduate School, Khon Kaen University Thailand

Sun, B., Q. Liao, P. Xie, G. Zhou, Q. Shi, and H. Ge. 2012. A cost-benefit analysis model of vehicle-to-grid for peak shaving. Power System Technology 10: 008.

Tahar, D.N., F. Yalaoui, C. Chu, and L. Amodeo. 2006. A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. International Journal of Production Economics 99(1): 63-73. doi:10.1016/j.ijpe.2004.12.007

Thaweerungsrisup, O. 2010. On the minimization of total makespan with job splitting policy in parallel machine. Master of Engineering thesis in Industrial Engineering, Graduate School, Khon Kaen University Thailand, ISBN 974-626-404-4

Wang, W.L., H.Y. Wang, Y.W. Zhao, L.P. Zhang, and X.L. Xu. 2013. Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm. Computers & Operations Research 40(5): 1196-1206. doi:10.1016/j.cor.2012.12.007

Weaver, V.M., M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. 2012. Measuring energy and power with PAPI. In Parallel Processing Workshops. In 2012 41st International Conference on Sep. pp: 262-268.

Weng, M., J. Lu, and H. Ren. 2001. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. International Journal of Production Economics 70: 215-226. doi:10.1016/S0925-5273(00)00066-9

Wisittipanich, W., and V. Kachitvichyanukul. 2011. Differential evolution algorithm for job shop scheduling problem. Industrial Engineeering & Management Systems 10(3): 203-208.

Ying, K.C., and H.M. Cheng. 2010. Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. Expert Systems with Applications 37(4): 2848-2852. doi:10.1016/j.eswa.2009.09.006

Ying, K.C., Z.J. Lee, and S.W Lin. 2012. Makespan minimization for scheduling unrelated parallel machines with setup times. Journal of Intelligent Manufacturing 23(5): 1795-1803. doi:10.1007/s10845-010-0483-3