Industrial Engineering
& Management Systems
Vol 11, No 1, Mar 2012, pp.114-122
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2012.11.1.114

Non-Identical Parallel Machine Scheduling with Sequence and Machine Dependent Setup Times Using Meta-Heuristic Algorithms

Cheol Min Joo
Department of Industrial and Management Engineering, Dongseo University
Byung Soo Kim*
Graduate School of Management of Technology (MOT), Pukyong National University

(Received: August 23, 2011 / Revised: January 27, 2012 / Accepted: February 9, 2012)

ABSTRACT

This paper considers a non-identical parallel machine scheduling problem with sequence and machine dependent setup times. The objective of this problem is to determine the allocation of jobs and the scheduling of each machine to minimize makespan. A mathematical model for optimal solution is derived. An in-depth analysis of the model shows that it is very complicated and difficult to obtain optimal solutions as the problem size becomes large. Therefore, two meta-heuristics, genetic algorithm (GA) and a new population-based evolutionary meta-heuristic called self-evolution algorithm (SEA), are proposed. The performances of the meta-heuristic algorithms are evaluated through compare with optimal solutions using randomly generated several examples.

Keywords: Non-Identical Parallel Machine Scheduling, Sequence and Machine Dependent Setup Time, Meta-Heuristic Algorithms

* Corresponding Author, E-mail: iekbs@pknu.ac.kr

1. INTRODUCTION

A technique for production scheduling problems is a methodology that optimizes the use of available resources by ordering a sequence of operations of jobs assigned to each resource. The scheduling problems are computationally very complex and therefore it is difficult to optimally solve in a reasonable time due to the combinatorial nature of its solution. As a result, finding a near-optimal solution in a reasonable time is important in real manufacturing area.

In a parallel machine scheduling problem, a set of the independent jobs are processed on a number of available identical parallel machines. Each machine processes only one job at a time, and each job can be processed on one of any machines with same processing time. But, in a non-identical parallel machine scheduling problem, the processing times of jobs depend upon the machine to which they are assigned to. Furthermore, the consideration of setup times between jobs is dependent upon not only the sequence of jobs to be processed on a machine but also the machine to which the jobs are assigned. In other words, the setup time between job i and j on machine k is different from the setup time between job j and i on the same machine k. In addition, the setup time between job i and j on machine s is different from the setup time between job i and j on machine p. This is the most common in the manufacturing industry, which is also the generalization of a parallel machine scheduling in practice.

Solving scheduling problems with sequence-dependent setup times is a very active research area (Allahverdi *et al*., 2008). The single machine scheduling problem with sequence-dependent setup times is known to

be NP-hard (Pinedo, 1995), and for the case of parallel machines, it is proved that the problem of minimizing the makespan with two identical machines is NP-hard (Garey and Johnson, 1997). The problem of minimizing the makespan on a scheduling problem of *m* parallel machines with sequence-dependent setup times is also NP-hard (Nait *et al*., 2003; Sveltana *et al*., 2001; Yalaoui and Chu, 2003). But the problem of this paper is more complex because it considers not only considering sequence-dependent setup times but also machine dependent setup times and machine dependent processing time.

For identical parallel machine problems, Monma and Potts (1989) considered the complex computing of scheduling parallel machine with sequence-dependent setup cost in batch processing machines. Tahar *et al*. (2006) presented a heuristic based on linear programming modeling to minimize maximum completion time for sequence-dependent setup time parallel machine scheduling. The performance of their proposed method was tested on problems with a lower bound. Rocha *et al*. (2007) proposed a variable neighborhood search algorithm and showed that their algorithm outperforms a greedy randomized adaptive search procedure algorithm. Several papers addresses the sequence-dependent setup times and non-zero release times for identical parallel machine scheduling was addressed. Ovicik and Uzsoy (1995) presented a time based decomposition approach to minimize the maximum lateness of the jobs. Nessah *et al*. (2005) presented a heuristic to minimize the total weighted completion time. Logendrana, *et al*. (2007) considered the minimization of the weighted tardiness of jobs in unrelated parallel machine scheduling with sequence-dependent setup times considering dynamic release of jobs and dynamic availability of machines. In Tahar *et al*. (2006) a linear programming approach was proposed with job splitting and sequence-dependent setup times. Pfund *et al*. (2008) presented a meta-heuristic method for the parallel machine scheduling problem with three objective functions. Gharehgozli *et al*. (2009) presented a new mixed-integer goal programming model to minimize the total weighted flow time and the total weighted tardiness simultaneously for a parallel machine scheduling problem with sequence-dependent setup times and release dates. Joo (2009) proposed ant colony optimization (ACO) for a parallel machine scheduling problem with sequence-dependent setup times and job release time.

Driessel and Monch (2010) addressed a parallel machine scheduling problem to minimize total weighted tardiness with sequence-dependent setup times, precedence constraints and ready time and proposed variable neighborhood search (VNS) heuristic to solve the problem.

For non-identical parallel machine scheduling problems, several studies have found. Agarwal *et al*. (2006) proposed several heuristics using a new neural network formulation to minimize the makespan. Hop and Nagaur (2004) considered the *n* printed circuit boards (PCBs) scheduling problem processed by *m* non-identical machines to minimize total makespan using a genetic algorithm. Li and Yang (2009) considered for the non-identical parallel machine scheduling to minimize total weighted completion time. Tavakkoli-Moghaddam *et al*. (2009) proposed a genetic algorithm to minimize the number of tardy jobs and total completion time for a unrelated parallel machine scheduling problem with sequence-dependent and machine-dependent setup times, ready times, and due-times. Gharehgozli *et al*. (2009) presented a new mixed-integer goal programming model to minimize the total weighted flow time and the total weighted tardiness simultaneously for a non-identical parallel machine scheduling problem with sequence-dependent and machine-dependent setup times, ready times, and due-times. Balin (2011) presented a genetic algorithm for the non-identical parallel machine scheduling to minimize makespan of the machines without having setup times, ready times, and due-times. In the non-identical parallel machine scheduling problem, if setup times are considered, they are sequence-dependent and machine-dependent in general case. In this paper, the authors propose a new "crossover operator" and a new "optimality criterion" in order to adapt the GA to non-identical parallel machine scheduling problem. Ko *et al*. (2010) proposed a dispatching rule that guarantees a predetermined minimum quality level for non-identical parallel machines with multiple product types. They only considered sequence-dependent setup times between product types. Vallada and Ruiz (2011) presented hybrid genetic algorithms to minimize makespan including a fast local search and a local search enhanced crossover operator based on the two dimensional representation of a chromosome for non-identical parallel machine scheduling problem.

Motivated by the literature discussed above, this paper considers the non-identical parallel machine scheduling problem with sequence and machine dependent setup times to minimize makespan. Though the problem can be formulated to a mathematical model, the model is not tractable as the problem size become large. Therefore, two meta-heuristic approaches are designed to solve the problem in a reasonable time. In section 2, a mathematical model is derived for finding the optimal solution. Two meta-heuristic algorithms, genetic algorithm (GA) and a new population-based evolutionary meta-heuristic called self-evolution algorithm (SEA), are proposed in section 3. In section 4, the performances of the meta-heuristics are evaluated through computational experiments. Finally, summary and further research areas are remarked in section 5.

2. MATHEMATICAL MODEL

In this section, we propose a mixed integer programming model for a non-identical parallel machine

scheduling problem with sequence and machine dependent setup times scheduling problem to minimize makespan. The following notations are used:

Parameters

p_{ik} : processing time of job i on machine k.
s_{ijk} : sequence and machine dependent setup time of job j processed after job i on machine k.
s_{Oik} : setup time of job i if job i is the first job in job sequence on machine k.
MS : set of machines
JS : set of jobs to be scheduled
O : dummy job index
M : big number

Continuous variables

x_i : starting time of job i
$c_{\max}$: makespan

Binary variables

$$y_{ik} = \begin{cases} 1, & \text{if job } j \text{ is assigned to machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijk} = \begin{cases} 1, & \text{if job } j \text{ is processed after job } i \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$z_{Oik} = \begin{cases} 1, & \text{if job } i \text{ is processed as the first job on machine } k \\ 0, & \text{otherwise} \end{cases}$$

The mathematical model is as given below:

$$Min \quad z = c_{\max} \tag{1}$$

s.t.

$$x_i + \sum_{k \in MS} \left(s_{oik} z_{oik} + \sum_{\substack{h \in JS \\ h \neq i}} s_{hik} z_{hik} + P_{ik} y_{ik} \right) \leq c_{\max} \quad \text{for} \quad \forall i \in JS \tag{2}$$

$$x_i + \sum_{k \in MS} \left(s_{oik} z_{oik} + \sum_{\substack{h \in JS \\ h \neq i}} s_{hik} z_{hik} + P_{ik} y_{ik} \right) \leq x_j + M\left(1 - \sum_{k \in MS} z_{ijk}\right), \quad \text{for} \quad \forall i, j \in JS, \ j \neq i, \tag{3}$$

$$\sum_{k \in MS} y_{ik} = 1, \quad \text{for} \quad \forall i \in JS, \tag{4}$$

$$\sum_{j \in JS} z_{ojk} = 1, \quad \text{for} \quad \forall k \in MS, \tag{5}$$

$$\sum_{\substack{j \in JS \\ j \neq i}} z_{jik} + z_{oik} = y_{ik}, \quad \text{for} \quad \forall i \in JS; \forall k \in MS, \tag{6}$$

$$\sum_{\substack{j \in JS \\ j \neq i}} z_{jik} = y_{ik}, \quad \text{for} \quad \forall i \in JS; \forall k \in MS, \tag{7}$$

$$x_i \geq 0, \quad \text{for} \quad \forall i \in JS,$$

$$y_{ik} = 0 \text{ or } 1, \quad \text{for} \quad \forall i \in JS; \forall k \in MS,$$

$$z_{ijk} = 0 \text{ or } 1, \quad \text{for} \quad \forall i, j \in JS; \forall k \in MS,$$

$$z_{oik} = 0 \text{ or } 1, \quad \text{for} \quad \forall i \in JS; \forall k \in MS,$$

In this model, a dummy job notation O is introduced to take care of the setup time for the job assigned to the first position in the sequence on each machine. Constraint (2) calculates the makespan. Constraint (3) ensures the precedence relation of jobs assigned in the same machine. Constraint (4) confirms that each job is processed in exactly one machine. Constraints (5)-(7) ensure that jobs assigned in the same machine must be appeared once in their sequence. Constraint (5) guarantees that the dummy job O is positioned at the beginning of the sequence before all the jobs on each machine. Constraint (6) depicts that if a job is assigned to a machine, then it will be immediately preceded by one job. Similarly Constraint (7) describes that if a job is assigned to a machine then it can be succeeded by at most one job. The job in the last position of the sequence on a machine will not have a succeeding job. Therefore, in total, model contains $2(JS \times MS) + JS^2 \times MS$ binary variables, $JS + 1$ continuous variables and $JS^2 + 2(JS \times MS) + 2JS + MS - 1$ constraints. This MIP model will be used later in the computational experiments.

3. META-HEURISTIC ALGORITHMS

The mixed integer programming model is not tractable for the problems over total 12 jobs because of the computation time (See Section 4). Thus, we focus on developing effective meta-heuristic approaches instead. In identical parallel machine scheduling problems with sequence dependent setup times, several authors have reported the effectiveness of the meta-heuristic approaches. Mendes *et al*. (2002) proposed TS (Tabu Search) in the parallel machine scheduling problem with sequence-dependent setup times. Behnamian *et al*. (2009) proposed hybrid algorithm including ACO (Ant Colony Optimization), SA (Simulated Annealing) and VNS (Variable Neighborhood Search). Joo (2009) proposed an improved (ACO) for a parallel machine scheduling problem with sequence-dependent setup times and job release time. Tavakkoli-Moghaddam *et al*. (2009) proposed a genetic algorithm with one dimensional representation with a special character for unrelated parallel machine scheduling problem. Balin (2011) presented a genetic algorithm for non-identical parallel machine scheduling using 0-1 encoding of two dimensional representation.

In this paper, we propose two meta-heuristics as solution approach for the non-identical parallel machine scheduling problem; conventional genetic algorithm (GA) and self-evolution algorithm (SEA) using one dimensional representation with special character. SEA is a new population-based evolutionary meta-heuristic. GA is one of the most powerful and broadly applicable meta-heuristic based on principles from evolution theory. GA

was first introduced and investigated by Holland (1975), and known as an effective and efficient algorithm for combinatorial optimization problems (Gen and Cheng, 2000). In general, the solution representation (chromosome) with a special character for separating machines has been used to apply GA in parallel machine scheduling problems (Cheng and Gen, 1997; Tavakkoli-Moghaddam *et al*., 2009). In GAs, two chromosomes are selected as parents and execute a sexual reproduction by the crossover operation. Therefore, the good characteristic of both chromosomes can be inherited to next generation. However, SEA executes a self-repro-duction by a single parent without the sexual reproduction by two parents. SEA explores broader solution space than GA by constantly maintaining the randomness during every generation and provides better effectiveness of solutions than GA by various neighborhood-search operations.

3.1 Representation of Solutions

In meta-heuristics, the solution performance is highly dependent upon the representation of a solution. For both meta-heuristic GA and SEA, it is necessary to describe the representation of a solution for the corresponding non-identical parallel machine schedule, and the representation is called *chromosome*. To represent machine assignments and the job sequences of each machine, the chromosome with a special character for separating machine is used in this paper. This representation has been popular in parallel machine scheduling problems since firstly introduced by Cheng and Gen (1997). The chromosome with a special character for separating machines has a single dimensional string array expressed by n digits from 1 to n and (m-1) machine separation indicator '*'s, where n is the number of jobs and m is the number of machines. The digits between '*' represent the sequence of jobs assigned to a same machine. Hence, the dispatching jobs to the corresponding machines and the sequencing of jobs in each machine are straightforwardly determined at the same time. Based on the representation of a chromosome with machine separation indicator '*' and its corresponding schedule are illustrated in Figure 1. The chromosome represents the assigning and sequencing of nine jobs to two machines. Job 2, 4, 8 and 3 are assigned to machine 1 and job 7, 1, 6, 9 and 5 are assigned to machine 2.

3.2 Genetic Algorithm (GA)

In the GA using chromosomes with special character, every chromosome is encoded into a structure that represents its properties, and the set of chromosomes forms a population. Initial population is generated randomly for the first generation. The chromosomes in the population are evaluated using the makespan of non-identical parallel machines as the measure of fitness. The chromosomes that have higher fitness value (lower objective function value) than the average fitness of current population make a potential parent pool. Parents are randomly selected only in the potential parent pool. Using three genetic operators such as a crossover operator, a mutation operator and a reproduction operator, the selected parents reproduce new chromosomes (i.e., children) to generate a population for the next generation. One-point crossover is used for both GA, and the crossover is illustrated in Figure 2. One point is randomly selected for dividing one parent. The set of genes on left side is inherited from the parent to the child, and the other genes are placed in order of their appearance in the other parent. For the mutation, two genes of a parent randomly selected are interchanged in GA as shown in

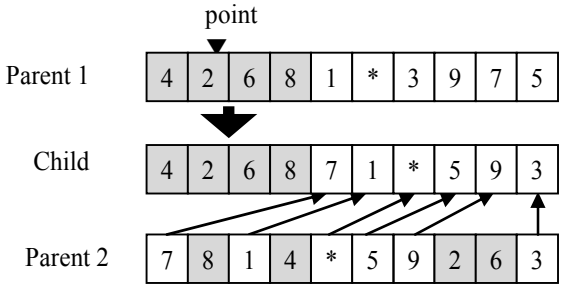


Figure 2. One-Point Crossover for GA.

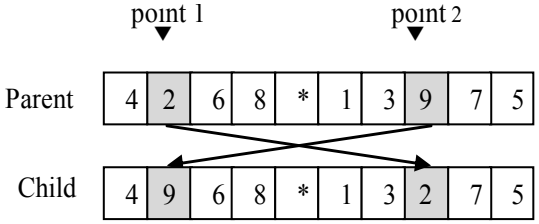


Figure 3. Swap Mutation for GA.

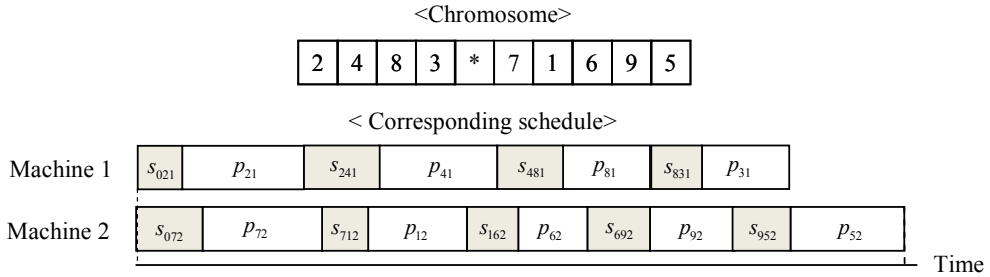


Figure 1. Chromosome and Corresponding Schedule.

Figure 3 (swap mutation). So a new generation is probabilistically formed according to the fitness values of chromosomes by genetic operators. Then the generation is evaluated and this process is repeated until a stopping criterion (maximum number of generations) is met.

3.3 Self-Evolution Algorithm (SEA)

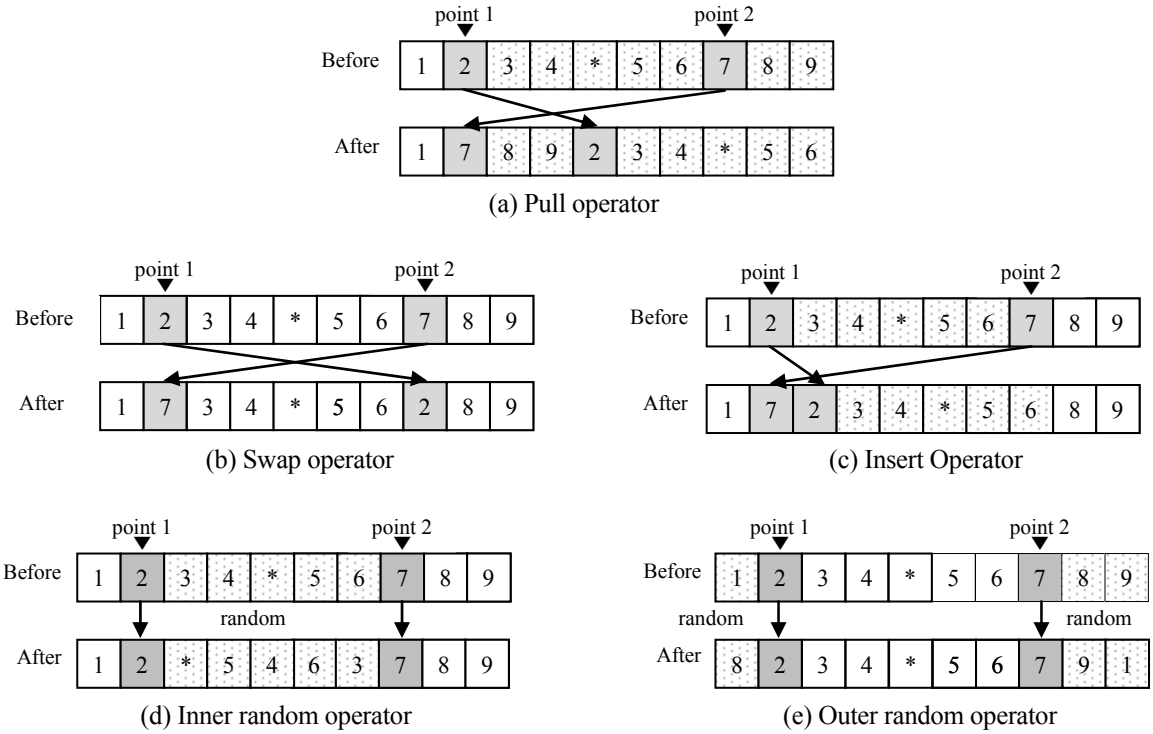
SEA is a new meta-heuristic algorithm which has a population (a set of solutions) based mechanism using the evolution of a solution by itself (self-evolution). Similar to GA, the set of chromosomes forms a population. The chromosome with special character proposed in Section 3.1 is also used for SEA. Initial population is generated randomly, and the chromosomes in the population are evaluated using the makespan of the non-identical parallel machines as the measure of fitness. A chromosome from the population is randomly selected and executes a self-reproduction using a randomly selected evolution operator to make a new chromosome. Then the new chromosome is evaluated and it replaces the original chromosome, if the fitness value of the new chromosome is better than that of the original chromosome. The algorithm continues until the number of self-reproductions becomes a predetermined stopping value.

We propose five evolution operators (pull operator, insert operator, swap operator, inner random operator and outer random operator) to make a new chromosome. For the operators, two points in the selected original chromosome are randomly selected. The pull operator is illustrated in Figure 4(a). The genes on right side of point 2 (including point 2) are pulled to the position of point 1, and the genes between point 1 and 2 (including point 1) are placed after. The two genes at the points are interchanged for swap operator as shown in Figure 4(b). Insert operator simply insert the gene at point 2 into the position of point 1 as shown in Figure 4(c). Inner random operator and outer random operator are illustrated in Figure 4(d) and Figure 4(e). The inner or outer genes of point 1 and 2 are randomly replaced for the operators.

SEA is running without providing any parameters for the algorithm, because all the selection processes in SEA, such as selection of chromosome from the population for self-evolution, selection of evolution operator, and selection of points for the operator are randomly executed.

4. COMPUTATIONAL RESULTS

To evaluate the performances of the meta-heuristic algorithms proposed in this paper, computational experiments were conducted using randomly generated test problems. Since the complexity of a problem highly depends on the number of jobs per machine, we fixed the number of machines as 2, 3, and 4 and generated two problem groups according to the average job size per machine. Total job size of each problem group is summarized in Table 1. The processing time and sequence and machine dependent setup time were randomly generated according to the range of [60, 180] and [10, 60], respectively, and the initial setup time was randomly



(a) Pull operator

(b) Swap operator

(c) Insert Operator

(d) Inner random operator

(e) Outer random operator

Figure 4. Operators for SEA.

generated by one value in the 70% range of the setup time.

Table 1. Total Job Size.

(a)Small sized problems

Jobs/MC	MC		
	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

(b) Large Sized Problems

Jobs/MC	MC		
	2	3	4
5	10	15	20
10	20	30	40
20	40	60	80
30	60	90	120

The small sized problems in the first group were generated for comparing the solutions obtained by meta-heuristic algorithms with optimal solutions. We used ILOG CPLEX 10.2 for finding the optimal solutions with the mathematical model presented in section 2. We imposed a 3600(sec.) time limit and simply terminated a particular run if the optimal solution had not been found and verified in that amount of time. The second group is to compare the performance of each meta-heuristic algorithm with large sized problems. GA was running with a population size of $2 \cdot n$ and a generation size of 1000, and fixed crossover and mutation rates of 0.8 and 0.2. The values of the crossover and mutation rates were predetermined by extensive preliminary experimentations. SEA was running with $2 \cdot n \times 1000$ iterations in order to be equally compared with GAs. All experiments solving each test problem using CPLEX and meta-heuristic algorithms were executed on a PC with 1.86 GHz Intel Core 2 processor and 2 GB RAM.

The test results of small sized problems are summarized in Table 2. Table 2 shows the optimal solution by CPLEX and mean and mean absolute deviation (*MAD*) of 10 replications, and relative percentage deviation (*RPD*) calculated with the expression (8) by GA and SEA for each test problem.

Table 2. Test Results of Small Sized Problems.

			CPLEX		GA			SEA		
Jobs/MC	MC	Jobs	*Opt.*	Time(sec.)	*RPD*(%)	*MAD*(%)	Time(sec.)	*RPD*(%)	*MAD*(%)	Time(sec.)
	2	4	238	1.547	0.00	0.00	0.044	0.00	0.00	0.058
	2	4	224	1.469	0.00	0.00	0.039	0.00	0.00	0.052
	2	4	241	1.297	0.00	0.00	0.039	0.00	0.00	0.044
	3	6	259	2.062	0.69	0.11	0.074	0.00	0.00	0.141
2	3	6	289	3.032	0.48	0.08	0.074	0.00	0.00	0.146
	3	6	207	2.281	0.00	0.00	0.074	0.00	0.00	0.144
	4	8	260	183.328	1.65	0.23	0.122	0.00	0.00	0.250
	4	8	219	88.406	14.70	1.09	0.120	0.37	0.07	0.258
	4	8	234	213.250	8.80	0.28	0.121	0.64	0.09	0.264
	2	6	392	2.547	0.46	0.06	0.069	0.00	0.00	0.130
	2	6	441	2.828	1.18	0.19	0.069	0.00	0.00	0.127
	2	6	354	3.344	0.25	0.05	0.069	0.00	0.00	0.132
3	3	9	391	727.859	2.69	0.31	0.132	0.38	0.03	0.323
	3	9	392	1211.641	0.97	0.12	0.133	0.00	0.00	0.318
	3	9	366	1277.422	2.35	0.10	0.132	0.68	0.07	0.318
	4	12	NA	+(3600)	-	0.42	0.225	-	0.09	0.554
	4	12	NA	+(3600)	-	0.57	0.223	-	0.15	0.565
	4	12	NA	+(3600)	-	0.27	0.231	-	0.16	0.566
	2	8	525	304.719	0.00	0.00	0.106	0.00	0.00	0.230
	2	8	544	173.610	1.40	0.17	0.105	0.00	0.00	0.228
	2	8	546	154.437	3.11	0.23	0.105	0.00	0.00	0.242
4	3	12	NA	+(3600)	-	0.14	0.211	-	0.11	0.552
	3	12	NA	+(3600)	-	0.19	0.214	-	0.13	0.545
	3	12	NA	+(3600)	-	0.12	0.216	-	0.07	0.550
	4	16	NA	+(3600)	-	0.47	0.370	-	0.14	0.940
	4	16	NA	+(3600)	-	0.63	0.386	-	0.24	0.968
	4	16	NA	+(3600)	-	0.63	0.370	-	0.17	0.930
Avg.					2.15	0.24	0.151	0.11	0.06	0.355

$$RPD\ (\%) = \frac{MH_{sol} - Best}{Best} \times 100, \quad (8)$$

where MH_{sol} is the solution obtained by meta-heuristic algorithms and *Best* is the best solution of all experiments for each test problem. *Best* can be optimal solution if the optimal solution is obtained. The optimal solutions for all the test problems up to total 12 jobs could not be obtained by CPLEX in a time limit. The *RPD*s and *MAD*s of SEA are much better than those of GA in all test problems. Both the *RPD*s and *MAD*s of SEA are nearly 0 in all test problems. This means that SEA is a very effective algorithm with low variation for the non-identical parallel machine scheduling problem.

The *RPD* and *MAD* of 10 replications by meta-heuristic algorithms of each large sized test problem are summarized in Table 3. Similar to the results of small sized problems, we can see that SEA is more effective with low variation than the conventional GA. Average *RPD* and *MAD* of SEA are 2.84 and 0.15, respectively. Meanwhile *RPD* and *MAD* of SEA are 11.59 and 0.21.

In order to validate the results, it is interesting to check if the observed differences in the *RPD* values of each meta-heuristic algorithm are statistically significant. Figure 5 shows the mean plots and Tukey HSD intervals at the 95% confidence level of all problems in Table 3. We can clearly see that there are statistically significant differences between the *RPD* values between SEA and GA because there is no overlap between the algorithms. The observed differences are more statistically signifi-

Table 3. Test Results of Large Sized Problems.

Jobs/MC	MC	Jobs	*Best*	GA			SEA		
				RPD(%)	*MAD*(%)	Time(sec.)	*RPD*(%)	*MAD*(%)	Time(sec.)
	2	10	529	2.95	0.13	0.117	0.40	0.06	0.431
	2	10	700	1.39	0.14	0.112	0.00	0.00	0.419
	2	10	665	2.08	0.16	0.112	0.00	0.00	0.424
	3	15	580	7.05	0.17	0.225	1.53	0.11	0.933
5	3	15	555	8.18	0.33	0.226	1.39	0.15	0.936
	3	15	590	3.92	0.11	0.225	1.97	0.09	0.932
	4	20	556	14.10	0.46	0.396	5.31	0.23	1.682
	4	20	592	7.25	0.37	0.396	3.14	0.15	1.656
	4	20	556	9.98	0.45	0.395	4.91	0.24	1.649
	2	20	1317	1.76	0.05	0.370	1.25	0.07	1.656
	2	20	1269	3.01	0.10	0.367	1.36	0.08	1.610
	2	20	1164	3.88	0.15	0.364	1.74	0.10	1.627
	3	30	1067	8.50	0.33	0.821	3.17	0.17	3.468
10	3	30	1019	8.34	0.18	0.813	4.73	0.23	3.468
	3	30	1098	5.66	0.12	0.822	4.61	0.17	3.511
	4	40	1039	18.07	0.40	1.503	5.18	0.20	5.989
	4	40	1147	15.48	0.33	1.503	6.40	0.29	5.985
	4	40	1137	12.44	0.22	1.506	2.27	0.15	5.985
	2	40	2292	2.60	0.08	1.440	2.13	0.10	5.948
	2	40	2522	2.57	0.08	1.442	1.90	0.11	5.940
	2	40	2222	4.73	0.08	1.428	2.52	0.10	5.968
	3	60	2190	13.37	0.21	3.332	3.17	0.13	13.046
20	3	60	2203	12.23	0.27	3.329	3.04	0.16	13.041
	3	60	2346	10.52	0.22	3.322	2.02	0.12	12.949
	4	80	2102	29.02	0.28	6.213	4.38	0.17	22.821
	4	80	2150	25.34	0.32	6.323	4.03	0.24	22.951
	4	80	2251	22.34	0.22	6.324	6.04	0.33	22.934
	2	60	3521	4.06	0.15	3.281	1.10	0.09	12.928
	2	60	3340	5.56	0.18	3.280	1.76	0.10	12.924
	2	60	3332	5.55	0.17	3.279	2.72	0.09	12.964
	3	80	3410	16.02	0.27	7.994	1.38	0.08	28.947
30	3	80	3416	17.02	0.16	7.959	2.38	0.13	28.895
	3	80	3197	20.65	0.18	8.002	3.88	0.16	28.918
	4	120	3266	30.23	0.14	15.780	3.49	0.26	51.535
	4	120	3259	30.19	0.19	15.769	2.97	0.18	51.110
	4	120	3294	31.18	0.19	15.804	4.08	0.23	51.251
Avg.				11.59	0.21	3.460	2.84	0.15	12.429

cant as the average number of jobs per machine and the number of machines increase, as shown in Figure 6. These results indicate that SEA consistently gives good performance for the non-identical parallel machine scheduling problem in any jobs size per machine and any machine size, but GA give worse performance as jobs size per machine and machine size become large.

The computation times of all test problems are summarized also in Table 2 and Table 3. The computation time of CPLEX significantly increases as the number of jobs per machine increases, and the optimal solution for the problem over total 12 jobs could not be obtained in a 3600(sec.) time limit by CPLEX. Meanwhile the computation time of GA is smaller than SEA, but the difference of the computation times is small enough to obtain solutions in a reasonable time. The difference is caused by the complexity of evolution operators for SEA.

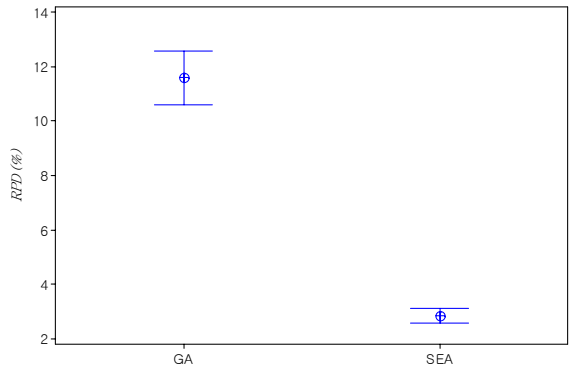


Figure 5. Mean Plots and Tukey HSD Intervals at the 95% Confidence Level of GA and SEA.

5. CONCLUSIONS

In this paper a non-identical parallel machine scheduling problem with sequence and machine dependent setup times is considered. The objective of this problem is to determine the allocation of jobs and the scheduling of each machine to minimize makespan. To address the problem, two different solution approaches are proposed. The first approach is based on a mixed integer programming model. Since the mathematical model is not tractable for the problems over total 12 jobs, we propose meta-heuristic algorithms (GA and SEA) to increase solution efficiency. SEA is a new meta-heuristic algorithm which has a population (a set of solutions) based self-evolution mechanism. Two problem groups are tested to verify the performance of proposed meta-heuristic algorithms. The test results indicate that SEA is very effective and efficient algorithm with low variation for the non-identical parallel machine scheduling problem.

Further study is required to assess the performance of SEA with other meta-heuristics (Simulated Annealing, Tabu-search and Ant-colony optimization, etc.) in the scheduling problems or other combinatorial problems.

REFERENCES

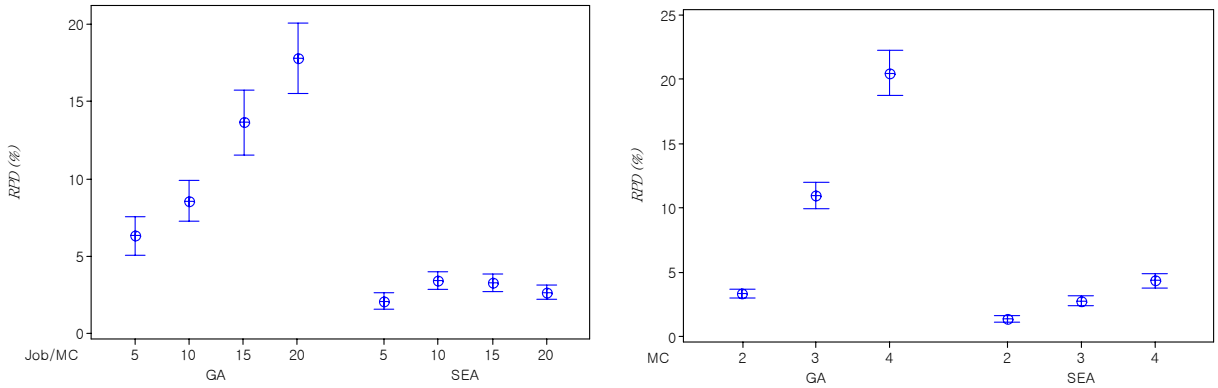
Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kolvalyov, M. Y. (2008), A survey of scheduling problems with setup times and costs, *European Journal of Operational Research*, **187**, 985-1032.

Agarwal, A., Colak, C., Jacob, V., and Pirkul, H., 2006. Heuristics and augmented neural networks for task scheduling with non-identical machines, *European Journal of Operational Research*, **175**(1), 296-317.

Balin, S. (2011), Non-identical parallel machine scheduling using genetic algorithm, *Expert Systems with Applications*, **38**, 6814-6821

Behnamian, J., Zandieh M., and Ghomi, F. (2009), Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm, *Expert Systems with Applications*, **36**, 9637-9644.

Cheng, R. and Gen, M. (1997), Parallel Machine Scheduling Problems Using Memetic Algorithms, *Com-*



(a) Jobs/MC Group (b) MC Group

Figure 6. Mean Plots and Tukey HSD Intervals at the 95% Confidence Level of GA and SEA.

puters and Industrial Engineering, **33**(3/4), 761-764.

Garey, M. and Johnson, D. (1997), *Computers and intractability: A guide to the theory of NP-completeness*, New York: W. H. Freeman.

Gen, M. and Cheng, R. (2000), *Genetic Algorithms and Engineering Optimization*, New York: Wiley.

Holland, J. H. (1975), *Adaptation in natural and artificial systems*, Ann Arbor, IL: University of Michigan Press.

Hop, N. V. and Nagarur, N. N. (2004), The scheduling problem of PCBs for multiple non-identical parallel machines, *European Journal of Operational Research*, **158**, 577-594.

Joo, C. M. (2009), An Improved Ant Colony System for Parallel-Machine Scheduling Problem with Job Release Times and Sequence-dependent Setup Times, *Journal of the Korean Institute of Industrial Engineers*, **35**(4), 218-225.

Mendes, A. S., Muller, F. M., Franca, P. M., and Moscato, P. (2002), Comparing metaheuristic approaches for parallel machine scheduling problems, *Production Planning and Control*, **13**, 143-154.

Nait, T. D., Chu, C., Yalaoui, F., and Amodeo, L. (2003), A new approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times based on linear programming, *In International conference on industrial engineering and production management* (IEPM'03), **3**, 266-274.

Pinedo, M. (1995), *Scheduling theory*, *algorithms*, *and systems*, Prentice-Hall.

Pfund, M., Fowler, J., Gadkari, A., and Chen, Y. (2008), Scheduling jobs on parallel machines with setup times and ready times, *Computers and Industrial Engineering*, **54**, 764-782.

Sveltana, A., Kravchenko, S., and Werner, F. (2001), A heuristic algorithm for minimizing mean flow time with unit setups, *Information Processing Letters*, **79**, 291-296.

Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., and Sassani, F. (2009), Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints, *Computers and Operations Research*, **36**, 3224-3230.

Tahar, D., Yalaoui, F., Chu, C., and Amodeo, L. (2006), A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times, *International Journal of Production Economics*, **99**, 63-73.

Vallada, E. and Ruiz, R. (2011), A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times, *European Journal of Operational Research*, **211**, 612-622.

Yalaoui, F. and Chu, C. (2003), An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times, *IIE Transactions*, **35**(2), 183-190.