International Journal of Operations Research

International Journal of Operations Research Vol. 4, No. 1, 42–49 (2007)

Parallel Machine Scheduling with Load Balancing and Sequence Dependent Setups

Mehmet B. Yildirim[1,*], Ekrem Duman[2], Krishnan Krishna[1], and Karthikeyan Senniappan[1]

[1]Department of Industrial and Manufacturing Engineering, Wichita State University, Wichita, KS 67260-0035, USA

[2]Industrial Engineering Department, Dogus University, Istanbul 34722, Turkey

Received August 2006; *Revised October 2006*; *Accepted November 2006*

Abstract—In this paper, we study the problem of minimizing total completion time with load balancing and sequence dependent setups in a non-identical parallel machine environment. A mathematical model has been presented for the objective of minimizing total completion time with workload balancing constraint. Since this problem is an NP-Hard problem, some simple heuristics and a genetic algorithm are developed for efficient scheduling of resources. The heuristics and genetic algorithm are tested on random data.

Keywords—Load balancing, Sequence dependent setups, Parallel machine scheduling, Scheduling theory, Genetic algorithms

1. INTRODUCTION

In this paper, we consider a parallel machine environment with sequence dependent setups where the objective is minimizing total completion and setup time while satisfying load balance constraints. In this parallel machine environment, by distributing the available workload among the available machines as equally as possible, bottlenecks can be eliminated, throughput can be maximized, work in process, finished goods inventory and operating costs can be lowered (Rajakumar et al., 2004).

The motivation of this paper comes from both service and manufacturing industries: for example, in an intensive care unit, nurses should have a balanced workload in order to provide a quality care and possibly allocate additional workload. In this case setups might be related to hygienic requirements for different patients. If a nurse visits her patients in a particular order, then there can be significant savings in setups. As an additional example, consider two workers in a machine shop. The first worker can operate a drill and the more experienced worker can operate a CNC machine in addition to the drill. To accommodate new orders efficiently, the goal of the production manager is to assign jobs to each worker in such a way that their workloads are similar. If an incoming order requires drilling, the order can be assigned to any of the workers. However, if it is a job which needs to be processed on the CNC machine, then the second worker has to be assigned. Furthermore, the order of jobs on the CNC machine and drill might affect the amount of setup required to complete the incoming orders.

Another example is a rolled aluminum sheet metal manufacturing process whereby ingots or slabs of pure aluminum are melted in a furnace. Then, additive elements such as magnesium, vanadium, etc., which determine the alloy of the aluminum, are added to the furnace in specific amounts. The melted metal in the furnace flows through a shaper and a rolling mill. The entire process takes place on casting lines. In this type of process, continuous production is required to avoid reheating a furnace after it cools down, which is long and costly. After the casting operation, to obtain the desired width and thickness, the metal is fed in coil form through a series of cold rolling mills, which successively reduce the metal thickness and recoil it after each rolling pass. Type of additives, and width and thickness of the rolled aluminum sheets determine the characteristics of an order (i.e., job). Each casting line is capable of processing aluminum sheets up to a certain width. The lines that can process wider sheets can also process the narrower sheets. Thus, each line have a certain capability. The objective in casting line scheduling is to minimize the total sequence dependent setup incurred between orders and balance the workload between parallel casting lines to accommodate potential orders.

Scheduling jobs with sequence dependent setups is a well studied problem. In their survey paper, Allahverdi et al. (1999) present the state of the art for machine scheduling problems with setups. Kurz and Askin (2001) note that single machine problem where the objective is minimization of the maximum completion time is NP-complete when there are sequence dependent setups. Thus, in a parallel machine environment with identical machines, minimizing the maximum completion time is NP-Complete as well (Kurz and Askin, 2001). Furthermore, Chen and Powell (2003) observe that solving non-identical parallel machines for any objective is more

* Corresponding author's email: Bayram.yildirim@wichita.edu

complex than the identical parallel machines and sequence dependent setup time will further complicate the problem.

In parallel machine scheduling problems, the machines are either identical (related) or non-identical (unrelated). Some of the problems considered in the literature for identical machines are minimize maximum completion time (C_{max}) objective with setups (Kurz and Askin, 2001); makespan, total weighted completion time, and total weighted tardiness objectives with setups (Fowler et al., 2003); total weighted tardiness objective with setups (Lee and Pinedo, 1997); workload balancing objective without setups (Rajakumar et al., 2004). For the non-identical machine case, weighted completion time objective without setups (Arnaout et al., 2005); batch-scheduling, no-setups (Chen, 2005); C_{max} objective (Glass et al., 1994 and Rabadi et al., 2006); flow time and tardiness objectives without setups (Randhawa and Kuo, 1997); CNC scheduling with setups (Turkcan et al., 2003); and makespan and flow time objectives without setups (Yu et al., 2002) are some of the problems considered in the literature.

Due to the nature of complexity involved in terms of formulation and computational time several methods are proposed to solve parallel machine problems with sequence dependent setups: dynamic programming (Gascon, 1998), branch and bound (Dietrich and Escudero, 1989) and heuristics (Pinedo, 1995). Rolling horizon heuristics are proposed by Uzsoy and Ovacik (1995) who decompose the scheduling problem into sub-problems and then optimally solve these problems by branch and bound algorithm. The genetic algorithms which were first developed by Holland (1975) have also been applied in scheduling parallel machines with setups: for example, Van Hop and Nagarur (2004) apply genetic algorithms to group similar boards, balance machines, and reduce setups in printed circuit board production. Kurz and Askin (2001), Fowler et al. (2003), Arnaout et al. (2005) and Rabadi et al. (2006) use genetic algorithms to solve parallel machine problem with sequence dependent setup times. Tamaki et al. (1993) propose a genetic algorithm to solve unrelated parallel machine scheduling problems with resource constraints. Chen (2006) uses heuristics and simulated annealing for unrelated parallel machines with mean tardiness objective and secondary resource constraints and setups.

The goal of workload balancing is to distribute the jobs/tasks to resources in such a way that the relative imbalance is minimized, i.e., utilization of resources is approximately equal. Rajakumar et al. (2004) test the effectiveness of random, shortest processing time and longest processing time dispatching rules on workload balancing in a parallel machine setting without any setups. They show that the longest processing time rule shows better performance for the higher number of jobs and/or machines. Becker et al. (2006) provide a survey of balancing on assembly lines. Another application is in workload balancing in printed circuit board assembly (Hillier and Brandeau, 2001).

In this paper, our goal is to formulate a mathematical model to solve the load balancing problem in parallel machine environment with sequence dependent setups and minimum sum of total completion time objective. In our setting, the machines (resources) are non-identical and at the same time they are related. As in the case of aluminum casting lines, some resources can process all of the jobs while others are less capable. The contributions of this paper can be summarized as follows: 1) We present a mathematical model to formulate the unrelated parallel scheduling problem with load balancing constraint and sequence dependent setups when the resources are non-identical and at the same time related; 2) The objective function is minimizing the sum of total completion time on all parallel lines. 3) Five heuristics are proposed and their performance is compared with a genetic algorithm.

The organization of this paper is as follows: In the next section, we present the notation utilized in this paper and a mixed integer mathematical model which minimizes the sum of the maximum completion time of jobs on each machine while satisfying the load balance constraint. Section 3 presents heuristics to solve this problem. Section 4 describes a genetic algorithm to determine the optimal machine assignment and job sequence to minimize sum of completion time on machines while satisfying the load balancing constraints. The experimental setup is followed by computational experimentation.

2. PROBLEM DESCRIPTION

Let K be the set of the parallel machines and $|K|$ denote the cardinality of set K, i.e., the number of machines. Similarly, J is the set of jobs. K_j is the set of machines on which job j can be processed and J_k is the set of jobs that can be processed on machine k. Let p_{ik} be the processing time of job i on machine k. If job i precedes job j on machine k, then there is a sequence dependent setup time of S_{ijk} between jobs i and j. The setups follow the triangle inequality. Let α be the maximum allowable level of imbalance on the production line where $0 \le \alpha \le 1$. Without loss of generality, it is assumed that if $k < k'$, then $J_k \subseteq J_{k'}$, i.e., jobs which can be processed on lower indexed machines can also be processed in higher indexed machines. Although the machines are non-identical, they are related. It can be assumed that they have different capabilities. We define this case as the semi-related machines case.

Jobs are available at time zero, and each job has a processing time and there is sequence dependent setup time between consecutive jobs. No preemption is allowed between jobs. Each job is processed on one machine and only once. The objective considered is to minimize the sum of the total completion times while balancing the load on all the non-identical parallel machines. The total completion time on machine k is the sum of processing time and sequence dependent setup time.

Below, is a mathematical program for Scheduling Parallel Machines with Load Balancing and Sequence Dependent Setups (SPM-LBSDS). The goal of SPM-LBSDS is to schedule jobs on semi-related parallel machines to minimize sum of completion times on all machines with load balancing constraints and sequence dependent setups.

The objective function minimizes the total completion time, C_{total}, (i.e., sum of processing time and setup time) needed to complete all available jobs on all of the processing lines

$$\min\ C_{\text{total}} \tag{1}$$

where

$$C_{\text{total}} = \sum_{k \in K} C_k \tag{2}$$

and C_k is the total processing and setup time on line k. Mathematically,

$$C_k = \sum_{i \in J} y_{ik} p_{ik} + \sum_{i \in J_k} \sum_{j \in J_k} x_{ijk} s_{ijk} \quad \forall k \in K \tag{3}$$

where

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

and

$$x_{ijk} = \begin{cases} 1 & \text{if job } i \text{ is the immediate predecessor of job } j \\ & \text{on machine } k \\ 0 & \text{otherwise} \end{cases}$$

Eq. (4) and (5) are workload balancing constraints for each machine. A perfect balancing is not usually possible. Hence, a certain percentage of tolerance α above and below the average workload are permitted.

$$C_k \le \frac{1}{|K|} C_{\text{total}} (1+\alpha) \quad \forall k \in K \tag{4}$$

$$C_k \ge \frac{1}{|K|} C_{\text{total}} (1-\alpha) \quad \forall k \in K \tag{5}$$

Constraint 6 ensures that each job is assigned to a processing line.

$$\sum_{k \in K_i} y_{ik} = 1 \quad i \in J \tag{6}$$

Constraint 7 guarantees that a job cannot precede another job on machine k unless it has been assigned to machine k.

$$x_{ijk} \le y_{ik} \quad i \in J,\ k \in K_i,\ j \in J_k \tag{7}$$

Constraint 8/constraint 9 ensure that a job must be before/after another job on a production line.

$$\sum_{i \in J_k} x_{ijk} \le y_{ik} \quad k \in K,\ j \in J_k \tag{8}$$

$$\sum_{j \in J_k} x_{ijk} \le y_{ik} \quad k \in K,\ i \in J_k \tag{9}$$

Constraint 10 represents sub-tour elimination constraints which ensures that a job cannot be the immediate predecessor or successor of two or more different jobs at the same time.

$$\sum_{i \in J_k'} \sum_{j \in J_k'} x_{ijk} \le |J_k'| - 1 \quad J_k' \subseteq J_k \tag{10}$$

Finding the solution for $1|S_{ij}|C_{\max}$ is NP-Complete. Hence, $P_m || C_{\max}$ and $R_m|S_{ijk}|C_{\max}$ problems are NP-complete (Kurz and Askin, 2001) where m is the number of machines. When $|K| = 1$, SPM-LBSDS reduces to $1|S_{ij}|C_{\max}$. As a result, SPM-LBSDS is NP-complete as well (Rabadi et al., 2006).

For this problem, the search space for determining the optimal solution is quite large. As a result, we do not expect the traditional optimization methods to perform well for larger sized problem instances. For example, for a problem with 20 jobs and five machines, after three "days" of running GAMS/CPLEX 9.0, the optimal solution was still not found on a Pentium IV 1.8Ghz computer with 1Gb of RAM. This motivated us for developing heuristics and a GA to find good solutions in reasonable amount of time (e.g., less than a minute of CPU time).

3. HEURISTICS TO SOLVE SEMI-RELATED PARALLEL MACHINES PROBLEM

The following is outline of the procedure for load balancing on semi-related machines when there are sequence dependent setups:

Step 1. INPUT the problem data
Step 2. SAVE the assignment order of the jobs in a LIST
Step 3. SELECT a job from the top of the LIST
Step 4. ASSIGN the job to a machine with respect to an ASSIGNMENT RULE
Step 5. REMOVE the job from the LIST
Step 6. If the LIST = ϕ, then STOP. Check the feasibility. Otherwise, go to *Step* 3.

Using this procedure, based on how the LIST is formed (i.e., how the assignment order is determined) and which ASSIGNMENT RULE is utilized, several dispatching rules/heuristics are proposed. The assignment orders/LISTs considered here are as follows:

Random (RN): The order of jobs is created randomly.

Longest Processing Time (LPT): Jobs are ordered in non-increasing processing time.

Longest Processing Time with Most Restricted Assignment First (LPT-MRAF): Jobs are first grouped with respect to the number of machines, $|K_j|$, which they can be processed in a non-decreasing order (i.e., jobs that

can be processed on one machine, then on two machines, ..., and finally on $|K|$ machines). In each group, the job order is obtained using the longest processing time.

In the fourth step, using the assignment rule, jobs selected from the list are assigned to machines to be processed. The assignment rules considered here are as follows:

Setup Avoidance (SA): Whenever jobs and machines are available, the SA rule searches for the machine/job combination that causes the least setup time. Fowler et al. (2003) note that the SA rule is a commonly used rule by scheduling practitioners for problems with sequence dependent setups when the objective is to minimize the makespan.

Cumulative Processing Time (CPT): CPT assigns the job to the machine with the least cumulative workload.

Hybrid Cumulative Processing Time and Setup Avoidance (CPT-SA): At any iteration, if the imbalance is within α then SA rule is used. Otherwise, CPT is applied. In case of a tie, jobs are assigned to the lower index machine. Note that the imbalance for machine k is calculated for any heuristic as

$$\alpha_k = \left|1 - \frac{C_k}{\overline{C}}\right| \quad \text{where} \quad \overline{C} = \frac{1}{|K|} C_{\text{total}} .$$

This definition of imbalance is different than the one proposed by Rajakumar et al. (2004) who used the maximum completion time instead of the average completion time in their calculations.

Table 1. Heuristics to minimize load balance in semi-related machines

Heuristic	Assignment Order	Assignment Rule
LPT-SA	LPT	SA
RN-CPT	RN	CPT
LPT-CPT	LPT	CPT
LPT-MRAF-CPT	LPT-MRAF	CPT
LPT-MRAF-CPT-SA	LPT-MRAF	CPT-SA

Based on the above assignment "orders" and "rules", the following heuristics are proposed: LPT-SA, RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-CPT-SA. As can be seen in Table 1, for each heuristic, the first part is the assignment order and the last part is the assignment rule. For example, LPT-SA is the heuristic, which orders the job in the longest processing time order and then assign to the lines using the setup avoidance rule. Similarly, in LPT-MRAF-CPT-SA, jobs are sorted with respect to the Longest Processing Time with Most Restricted Assignment First and then the Hybrid Cumulative Processing Time and Setup Avoidance rule is utilized to assign the jobs to machines.

4. A GENETIC ALGORITHM TO SOLVE SPM-LBSDS ON SEMI-RELATED PARALLEL MACHINES

According to Fowler et al. (2003), Genetic Algorithms (GAs) have widely been applied to parallel machine problems mainly for two reasons. (i) In each generation, the GA is capable of producing feasible solutions whereas in mathematical modeling, the possibility of getting the feasible solution largely depends on complexity and nature of the problem. (ii) The GA maintains a set of feasible solutions. According to Fowler et al. (2003), "The knowledge of a family of good solutions is far more important than obtaining an isolated optimum." The problem considered in this paper is NP-complete as explained in section 2. Thus, mixing GA with problem oriented heuristics might provide better results than using the heuristics by itself.

In the proposed GA, after the chromosome (the assignment order) is generated, CPT is used to assign jobs to machines. The fitness values of the solutions are then used to determine the next generation. The process is repeated for a fixed number of generations (iterations). The genetic algorithm can be summarized as

- Generate an initial population.
- Perform crossover and mutation operations to generate offsprings.
- Evaluate the fitness value of generated offsprings and give only those offspring that have better fitness values a chance to survive in the next generation.
- Repeat *Steps* 2 and 3 until the GA is run for the predetermined number of generations.
- Select the best chromosome.

Below, we explain components of the proposed GA in details.

Representation (Coding): A good representation scheme is necessary to describe the problem-specific characteristics in detail. The representation method plays a major role in the subsequent steps of GA. Genes can be represented in several forms such as binary, real integer number or a combination of characters (Tiwari and Vidyarthi, 2000). The sequence-oriented natural number representation scheme is used to solve the SPM-LBSDS problem. In this scheme, the length of the chromosome is equal to the total number of jobs to be scheduled on all the machines. For example, if there are six jobs to be scheduled, then the chromosome may be represented as [5 1 4 3 2 6], where the number in the brackets represents the jobs. Similar to LPT-MRAF, jobs are first grouped with respect to the number of machines, $|K_j|$, where they can be processed, in a non-decreasing order (i.e., first group has the jobs that can be processed on one machine, then the second group has the jobs that can be processed on two machines, and so on). In each job group, the job order is obtained using the order in the chromosome.

Initialization: Fowler et al. (2003) report that assigning jobs based on some pre-determined rules may provide better solutions and reduce computational time than generating random solutions. Hence, an initial set of solutions is generated using the RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-CPT-SA heuristics presented in Section 3.

Evaluation of Fitness Function: The fitness function f_p for a chromosome p to penalize the objective function severely can be calculated as a function of the objective function value as

$$f_p(C_{\text{total}}) = C_{\text{total}}^{\gamma-1} \exp^{-C_{\text{total}}}$$

where $\gamma = 0.5$.

Cross-over: Once parents are selected, the cross-over operation is applied to generate a new solution. Cross-over is the process by which two parent strings unite together to form new offspring strings. The cross-over operation will occur on a pair of strings only with a probability of the "cross-over probability" (0.95 in our GA). The type of cross-over employed in our GA implementation is single point cross-over: Two parent strings are selected randomly from the population. A random number between 0 to $l - 1$ is generated (l is the length of the chromosome) to determine the cross-over point. When crossover is done, the genes before the crossover point in chromosome 1 are the first part of the child chromosome. The second part of the child chromosome is generated by checking the genes from the second chromosome one by one and adding the genes which are not in the child chromosome yet. For example if a crossover operation is applied to chromosomes [1 5 2 6 3 4] and [1 6 2 5 3 4]. Assuming the cross-over point is after gene 3, the resulting child chromosome will be [1 5 2 6 3 4].

Mutation: Mutation creates a new chromosome by altering the locus of the genes. As in the cross-over operation, mutation occurs only if the random number generated is less than or equal to the mutation probability. The mutation probability is set to 0.05. Type of mutation operator in this paper is the reciprocal exchange in which two positions on a chromosome are randomly selected and the genes in those positions are then exchanged to form a new chromosome. For example: In a chromosome string [1 5 2 6 3 4] for a six job problem, two positions are randomly selected (e.g., positions 2 and 4). The genes in these positions are exchanged to form the new chromosome string [1 6 2 5 3 4].

Selection: The selection model should reflect the nature's survival of the fittest. Normally, in a genetic algorithm, chromosomes with a better fitness value will receive more chances to survive in the next generations. In this paper, the roulette wheel system is used to select the parents for crossover and mutation. The process of selection of the parents for generating the next population is described below:

Step 0. Determine the fitness function value for different chromosomes in an initial population.
Step 1. Arrange the chromosomes in descending order of the fitness value.
Step 2. Calculate the probability function $Pr(f_p)$ for individual chromosome p as a function of the fitness function values as

$$Pr(f_p) = \frac{f_p}{\sum_{p'} f_{p'}}$$

Step 3. Utilize the roulette wheel method to select candidates for cross over and mutation using the probability values found in *Step* 2.

Reproduction: We utilize the elitist selection scheme to determine the best chromosomes from the current and new population to form the next generation. All the candidate solutions have to be arranged in the descending order with respect to their fitness function value.

Note that the crossover and mutation operations do not necessarily produce a feasible result all of the time.

5. COMPUTATIONAL EXPERIMENTATION

This section discusses how to generate different scenarios to test the effectiveness of the proposed heuristics in two-machine and five-machine environments. The heuristics and GA are programmed using Visual C++ 6.0, and the statistics are collected using Microsoft Excel. Below is a description of the experimental design, which is followed by results and discussions.

5.1 Experimental design

The experimental setup considers the following factors: number of machines (two-machines (2M) and five-machines (5M)); number of jobs (20, 40, and 60); processing time-setup time ratio, ρ, (low ($\rho = 0.1$), medium ($\rho = 1$) and high ($\rho = 10$)); and categories of jobs which can be processed on semi-related machines (two levels for the 2M case and seven levels for the 5M case). As a result, the total number of scenarios are 18 for the 2M case and 63 for the 5M case.

In two machine setting, there are two categories of jobs. The first category can be processed on both machines while the second category can only be processed on machine two (i.e., the more capable machine). On the other hand, in the five machines setting, machines three, four, and five are the most capable machines and identical. Machine two can process all of the jobs that machine one can process and is less capable than machines three to five. As a result, there are three categories of jobs. The first category can be processed on machines one to five; similarly, the second and third categories of jobs can be processed on machines two to five and three to five, respectively. The proportion of total processing and setup time allocated to each of the categories is examined on

seven levels as shown in Table 2.

Table 2. Experimental setup for types of job categories

	Scenarios									
	2M			5M						
Category	1	2	3	1	2	3	4	5	6	7
1	45	50	55	50	50	50	25	40	25	40
2	55	50	45	10	25	40	50	50	25	10
3				40	25	10	25	10	50	50

Experiments were performed on several values of imbalance. When $\alpha = 0.05$ or $\alpha = 0.10$, the proportion of infeasible solutions by the proposed heuristics was significantly high. When $\alpha > 0.20$, the problem behaved similarly to unrelated parallel machine scheduling problem with completion time objective. As a result, $\alpha = 0.15$ is chosen for experimentation. Note that finding a partition of jobs that satisfy the load balancing constraint is similar to the set partitioning problem, which is NP-hard.

The setup time matrix is asymmetric (i.e., s_{ijk} may not be equal to s_{jik}). The job data, processing time and setup time are generated using a method similar to Fowler et al. (2003).

Finally, selection of good parameters can play a vital role in GA experiments. We performed several experiments to determine the sensitivity of the problem to GA parameters: The size of the population was tested for 5, 10, 15, 20 and 25. The population size of 10 worked the best. When we tested the number of generations for 500, 1000, 2000 and 3000, in 2000 generations, a quality solution is obtained in reasonable amount of CPU time (in less than 10 CPU seconds). The cross-over and mutation probability are 0.95 and 0.05, respectively. In the genetic algorithm, one hundred initial solutions (five solutions from the proposed heuristics and 95 random solutions) are generated. Hence, out of the one hundred initial solutions, the ten best solutions based on the fitness value are selected. At any iteration, after the new population is generated, the ten best solutions from both the new and current set of solutions is used to determine the most elite solution. While selecting the ten best solutions, repetition of solutions is avoided. This helps to maintain the diversity in the population, and to prevent the solution from being stuck at local optimum.

5.2 Results and discussion

Table 3. Performance of heuristics (C_{total} and total setup) in the 2M and 5M environments

	C_{total}		total setup	
Heuristic	2M	5M	2M	5M
RN-CPT	1989.0	1976.7	118.2	110.3
LPT-CPT	2239.0	2029.5	159.2	145.1
LPT-MRAF-CPT	2002.4	1985.1	131.7	118.8
LPT-MRAF-CPT-SA	1969.0	1956.6	108.6	94.8
GA	1937.9	1918.2	67.2	75.2

Each scenario is run for five different data sets on a Pentium IV 1.6 Ghz machine with 512MB of RAM. Furthermore, for each data set, the genetic algorithm is run five times to obtain different solutions. Although, five different heuristics (i.e., LPT-SA, RN-CPT, LPT-CPT, LPT-MRAF-CPT, and LPT-MRAF-CPT-SA) are tested, the SA assignment rule did not produce feasible results for 96% of the cases. Note that the SA assignment rule is widely used in practice (Fowler et al., 2003) for parallel machine scheduling with setups for minimum completion time objective. However, it is observed that this rule can not be used to solve problems with load balancing in parallel machine scheduling. LPT-MRAF-CPT and then RN-CPT give feasible solutions in all of the cases that we have experimented. Recall that the LPT-MRAF first prioritizes the most restricted jobs and RN-CPT generates random orders until a feasible solution is determined. On average, RN-CPT determines a feasible assignment in 0.5 seconds in the 2M case and 1.5 seconds in the 5M case compared to 5.1 and 5.4 seconds for GA to determine its final solution in the 2M and 5M cases, respectively. Other heuristics do not consume any significant CPU time. LPT-CPT and LPT-MRAF-CPT-SA produce feasible solutions (i.e., satisfying the load balancing constraint) for 75% and 85% of the cases. As can be seen in Table 3, the GA outperforms the proposed heuristics. The performance of heuristics can be ranked as LPT-MRAF-CPT-SA, RN-CPT, LPT-MRAF-CPT and LPT-CPT.

Table 4. Performance of heuristics to solve the SPM-LBSDS problem

	$\overline{\sigma}$		$\overline{\alpha}$		$\overline{best}$	
Heuristic	2M	5M	2M	5M	2M	5M
RN-CPT	3.12	2.15	7.70	3.61	59.2	50.7
LPT-CPT	4.21	4.14	8.26	1.98	0.0	0.0
LPT-MRAF-CPT	3.22	3.04	3.98	2.17	7.4	7.9
LPT-MRAF-CPT-SA	2.76	2.20	8.25	4.19	55.5	47.6

Statistics are collected on the number of times that a heuristic performs the best: in the 2M case, RN-CPT gives the best solution in 59.2% of runs while LPT-MRAF-CPT-SA gives the best solution in 55.5% of the runs as shown in Table 4, the $\overline{best}$ columns). In Table 4, $\overline{\sigma}$ denotes the average deviation of a heuristic solution from the GA solution. In case of 2M, LPT-MRAF-CPT-SA obtains the lowest $\overline{\sigma}$ and LPT-MRAF-CPT obtains the lowest imbalance ($\overline{\alpha}$). Even though the heuristic LPT-MRAF-CPT-SA gives the best average deviation, it fails to give feasible solutions in 15% of the cases. As a result, RN-CPT provides the second best average deviation and the maximum number of best solutions. Note that similar observations hold for the 5M case as well. Note that Rajakumar et al. (2004) report that the LPT-CPT gives best solutions with respect to load balancing in identical parallel machine environment without setups. However, the LPT-CPT performs poorly in case of semi-related machines with sequence dependent setups.

On the average, excluding the GA, Table 5 shows that the LPT-MRAF-CPT-SA heuristic gives the best average objective function value. This might be due to the fact that the assignment order is determined by considering the restrictions in job processing and sequence dependent setups at the same time. The second best performance is given by the RN-CPT heuristic. As the number of jobs increases from $|J| = 20$ to $|J| = 40$, the objective function value approximately doubles. However, this is not the case, when $|J|$ is increased from 40 to 60. In this case, C_{total} increases by 50%.

Table 5. Performance of the heuristics with respect to the number of jobs

	2M			5M		
Heuristic/n	20	40	60	20	40	60
RN-CPT	982.7	2015.0	2969.3	973.8	1972.0	2984.2
LPT-CPT	1357.3	2168.8	3191.0	1085.6	1984.7	3018.3
LPT-MRAF-CPT	992.0	2031.3	2984.0	983.2	1982.5	2989.8
LPT-MRAF-CPT-SA	965.6	1991.4	2949.9	958.6	1959.9	2951.4
GA	935.6	1973.1	2904.1	944.3	1938.0	2872.2

Table 6. Effect of processing time setup time ratio on the objective function

	2M			5M		
Heuristic/ρ	0.1	1.0	10	0.1	1.0	10
RN-CPT	919.3	2143.0	2904.7	905.5	2117.2	2907.3
LPT-CPT	1292.3	2226.9	3197.9	1016.7	2138.7	2933.2
LPT-MRAF-CPT	935.1	2157.7	2914.7	921.2	2124.5	2909.8
LPT-MRAF-CPT-SA	910.1	2130.7	2866.1	917.3	2070.5	2882.0
GA	868.5	2092.7	2852.5	876.4	2009.5	2868.7

Similar observations can be seen for the effect of the processing time-setup time ratio shown in Table 6. When the effect of setup time is minimal (i.e., the $\rho = 10$), the GA and heuristics have similar performance. For a smaller ρ, the GA outperforms all other heuristics.

Table 7. Effect of the number of jobs on the imbalance

	2M			5M		
Heuristic/n	20	40	60	20	40	60
RN-CPT	6.90	8.52	7.68	4.43	3.88	2.51
LPT-CPT	6.69	8.95	8.13	1.90	2.04	1.98
LPT-MRAF-CPT	3.94	4.68	3.33	3.04	1.61	1.87
LPT-MRAF-CPT-SA	7.99	7.90	8.79	3.47	4.00	4.02

Although the load balancing objective appears as a constraint in the SPM-LBSDS model, the performance of different heuristics is tested when the number of jobs (Table 7) and processing-setup time ratio (Table 8) is varied. In both cases, on average LPT-MRAF-CPT provides the best result. The only exception occurs in the 5M case when $\rho = 1$ and $\rho = 10$. In these cases LPT-CPT provides the lowest imbalance. However, the fact that when $|J|$ increases the imbalance decreases as in the parallel machine scheduling (Rajakumar et al., 2004) can not be observed.

Table 8 presents the effect of the number of jobs and processing-setup time ratio on the performance of the GA. When the processing time-setup time ratio, ρ, is small, i.e., the processing time is relatively less important than the setup time (in other words, setup time typically dominates the processing time), in the 2M case, the GA's performance is better compared to the "best" heuristic solution and improves as $|J|$ increases. On the other hand, in the 2M – $\rho = 10$ case in which processing time dominates the setup time, when $|J|$ increases, the gap between the GA solution and the best heuristic decreases. For the 5M case, it is observed that when $|J|$ increases, the GAP between the GA and the best heuristic solution decreases. Note that this observation is in line with the fact that Longest Processing Time dispatching rule performs well to balance loads on parallel machines when there is no setups.

Table 8. The GA performance as a function of processing time setup ratio and number of jobs with respect to the best heuristic solution

	2M			5M		
ρ	20	40	60	20	40	60
0.1	3.43	4.35	4.75	1.70	1.29	0.41
1.0	0.97	0.88	1.48	1.17	0.61	0.52
10.0	1.62	0.43	0.12	1.36	0.11	0.19

6. CONCLUSION

In this study, the scheduling of semi-related machines with sequence dependent setups and load balancing constraints is solved with the objective of minimizing sum of completion time on all machines. A mathematical programming model is proposed. Due to the computational complexity involved in solving the mathematical model, heuristics and a genetic algorithm have been developed to generate solutions within a reasonable period of time. Heuristics were also used to generate the initial set of solutions for the genetic algorithm, which resulted in reducing computational time as well. In order to test the effectiveness of the developed methodology, different scenarios have been generated. It is found out that the genetic algorithm improves the heuristic solutions significantly when the processing time setup time ratio is small. Furthermore, the most promising heuristic is the LPT-MRAF-CPT-SA which switches from CPT to SA (or vice a versa) based on the current relative imbalance.

A direct extension of this research is having a multiobjective mathematical model in which both minimization of total completion time and minimization of imbalance are two objectives that can considered. A multi-objective genetic algorithm approach can be developed to solve this problem as a part of future research. Another extension can be determining a lower bound to test the effectiveness of the genetic algorithm. Furthermore, a new heuristic which will take into account the processing times and setups at the same time can be developed. A more detailed experimental design to determine if there is a clear pattern between the parameters of the experimental design and the objective can be done.

REFERENCES

1. Allahverdi, A., Gupta, J.N.D., and Aldowaisan, T. (1999). A review of scheduling research involving setup consideration. *International Journal of Management Science*, 27: 219-239.
2. Arnaout, M.J. and Rabada, G. (2005). Minimizing the total weighted completion time on unrelated parallel machines with stochastic times. *Proceedings of the 2005 Winter Simulation Conference*, pp. 2141-2147.
3. Becker, C., Scholl, A., and Dolgui, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168: 694-715.
4. Chen J.F. (2005). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, 16: 285-292.
5. Chen J.F. (2006). Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups. *International Journal of Advanced Manufacturing Technology*, 29: 557-563.
6. Chen, Z.L. and Powell, W.B. (2003). Exact algorithms for scheduling multiple families of jobs on parallel machines. *Naval Research Logistics*, 50: 823-840.
7. Dietrich, B.L. and Escudero, L.F. (1989). On solving a 0-1 model for workload allocation on parallel unrelated machines with setups. *Proceedings the 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, pp. 181-186.
8. Fowler, J.W., Horng, S.M., and Cochran, J.K. (2003). A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. *International Journal of Industrial Engineering*, 10(3): 232-243.
9. Gascon, A. and Leachman, R.C. (1998). A dynamic programming solution to the dynamic, multi-item, single-machine scheduling problem. *Operations Research*, 36(1): 50-56.
10. Glass, C.A., Potts, C.N., and Shade, P. (1994). Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, 20(2): 41-52.
11. Hillier, M.S. and Brandeau, M.L. (2001). Cost minimization and workload balancing in printed circuit board assembly. *IIE Transactions*, 33(7): 547-557.
12. Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.
13. Kurz, M.E. and Askin, R.G. (2001). Heuristic scheduling of parallel machines with sequence-dependent setup times. *International Journal of Production Research*, 39(16): 3747-3769.
14. Lee, Y.H. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100: 464-474.
15. Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems*, Springer Series in Operations Research and Financial Engineering.
16. Rabadi, G., Moraga, J.R., and Al-Salem, A. (2006). Heuristics for unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17: 85-97.
17. Rajakumar, S., Arunachalam, V.P., and Selladurai, V. (2004). Workflow balancing strategies in parallel machine scheduling. *International Journal of Advanced Manufacturing Technology*, 23: 366-374.
18. Randhawa, S.U. and Kuo, C.H. (1997). Evaluating scheduling heuristics for non-identical parallel processors. *International Journal of Production Research*, 35: 969-981.
19. Tamaki H., Hasegawa Y., Kozasa J., and Araki M. (1993). Application of search methods to scheduling problem in plastics forming plant: A binary representation approach. *Proceedings of the 32nd IEEE Conference on Decision and Control*, pp. 3845-3850.
20. Tiwari, M.K. and Vidyarthi, N.K. (2000). Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. *International Journal of Production Research*, 38(14): 3357-3384.
21. Turkcan, A., Akturk, S.M., and Storer, H.R. (2003). Non-identical parallel CNC machine Scheduling. *International Journal of Production Research*, 41(10): 2143-2168.
22. Uzsoy, R. and Ovacik, I.M. (1995). Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times. *International Journal of Production Research*, 33(11): 3173-3192.
23. Van Hop, N. and Nagarur, N.N. (2004). The scheduling problem of PCBs for multiple non-identical parallel machines. *European Journal of Operational Research*, 158: 577-594.
24. Yu, L., Shih, M.H., Pfund, M., Carlyle, M.W., and Fowler, W.J. (2002). Scheduling of unrelated parallel machines: An application to PWB manufacturing. *IIE Transactions*, 34: 921-931.