Sequence-dependent group scheduling problem on unrelated-parallel machines

Mir Abbas Bozorgirad and Rasaratnam Logendran*

School of Mechanical, Industrial, and Manufacturing Engineering

Oregon State University

Corvallis, OR 97331-6001, USA.

Abstract

In this research we address a sequence-dependent group scheduling problem on a set of unrelated-parallel machines where the run time of each job differs on different machines. To benefit both producer and customers we attempt to minimize a linear combination of total weighted completion time and total weighted tardiness. Since the problem is shown to be NP-hard, meta-heuristic algorithms based on tabu search are developed to find the optimal/near optimal solution. For some small size yet complex problems, the results from these algorithms are compared to the optimal solutions found by CPLEX. The result obtained in all of these problems is that the tabu search algorithms could find solutions at least as good as CPLEX but in drastically shorter computational time, thus signifying the high degree of efficiency and efficacy attained by the former.

Keywords

Group Scheduling; Unrelated-parallel machines; Bi-criteria; Sequence-dependent setup time; Mixed-integer linear programming; Tabu search.

Mir Abbas Bozorgirad, Email: bozorgim@onid.orst.edu
*Corresponding Author: Rasaratnam Logendran, Tel: 1-541-737-5239, Fax: 1-541-737-2600, E-mail: Logen.Logendran@oregonstate.edu

1. Introduction

Scheduling problems were first considered in the mid-1950s, and since then, so many papers that deal with different aspects of scheduling have been published. A comprehensive survey on these problems has been reported by Allahverdi et al. (2008). The research reported in this paper considers three important properties of scheduling problems which are group scheduling, unrelated-parallel machines, and bi-criteria objective function.

Group scheduling follows from mainly applying the cellular manufacturing concept in which disaggregated manufacturing cells are formed to completely process different families of parts, and is usually performed in two levels: Outside and Inside. The outside level finds the desirable sequence of part families, i.e. groups; and the inside level finds the desirable sequence of jobs in each group (Logendran et al. (1995)). Group scheduling problems come with or without Group Technology Assumptions (GTA). GTA forces all jobs of a group to be processed contiguously, but scheduling without GTA allows the groups to be split into subgroups. Scheduling without GTA uses batch scheduling setup time, while that with GTA uses group scheduling setup time and is studied either as sequence-independent or sequence-dependent problem (Logendran et al. (2006a)). Sequence-independent setup time is when the required setup time for switching from one group to another, does not depend on the groups and their orders; otherwise the problem is a sequence-dependent problem.

On the other hand, parallel machine scheduling problems are mostly divided into three categories: Identical machines, P_m, where all machines have the same run time for each job; uniform machines, Q_m, where machines have different speeds, but each machine has a consistent rate; and unrelated machines, R_m, where each machine can work in different rates and has a different run time for each job (Allahverdi et al. (2008); Lin et al. (2011)).

The third important feature of this paper is to consider a bi-criteria objective function, as in many industries it is necessary to consider optimizing more than one criterion. This research tries to optimize the total weighted completion times, which favors the supplier’s interests, and at the same time tries to optimize the total weighted tardiness, which favors the customers’ interests.

Therefore, the problem being researched in this paper can be classified as $R|ST_{sd,b}, r_j|\sum w_j C_j + w_j T_j$ with respect to the three field notation of Graham et al. (1979). The rest of the paper is organized as follows. Section 2 reviews the literature, which mostly deals with the problem of interest. The problem is well stated in section 3, together with a mixed-integer linear programming model developed for this problem. Section 4 comprehensively describes the proposed search algorithm. Section 5 describes the data generation mechanisms, which have been used to generate the sample problems, and section 6 illustrates the search algorithm with the help of a sample problem. Sections 7 and 8 represent the experimental results and the comparison to the optimal solutions, respectively. Finally in section 9, a concise conclusion about the research is presented.

2. Related Work

Group scheduling is used in a wide variety of scheduling problems (Logendran and Nudtasomboon (1991); Logendran et al. (1995); Gelogullari and Logendran(2010); Logendran et al. (2005); Logendran et al. (2006b); Salmasi et al.(2010); Li et al. (2011); Xingong and Guangle (2010); Kuo and Yang (2006); and Yang and Chand (2008)). Li et al. (2011) tried to minimize an objective function which includes earliness, tardiness, due date assignment and flow time costs for a sequence-independent group scheduling problem on a single-machine. Logendran et al. (2006a) developed three search algorithms, based on tabu search, in order to minimize the total completion time for a two-machine sequence-dependent group scheduling problem. Later, Salmasi et al. (2011) extended their work to minimize the makespan of a sequence-dependent flow shop group scheduling problem. They developed a hybrid ant colony optimization algorithm (HACO) to solve the problem. Salmasi et al. (2010) developed a mathematical programming model to minimize the total flow time of a sequence-dependent flow shop group scheduling problem ($F_m|fmls, S_{plk}, prmu|\sum C_j$). Gelogullari and Logendran (2010) reported the only analytical research, which considered a carryover sequence-dependent flow shop group scheduling problem in the assembly of printed circuit boards in electronic manufacturing. Logendran et al. (2005) developed heuristics to solve the group scheduling problem with sequence-independent setups in flexible flow shops. Later, Logendran et al. (2006b) studied the group scheduling problem with sequence-dependent setups and developed search algorithms based on tabu search to efficiently solve industry-size problems. Shahvari et al. (2011) extended the research by Logendran et al. (2006b), and proposed a mathematical model for the same problem together with six metasearch algorithms to solve it heuristically.

With respect to the unrelated-parallel machine scheduling problems, Lin et al. (2006) classified these problems based on various correlation structures. Liaw et al. (2003), developed a Branch-and-Bound (B&B) algorithm to minimize the total weighted tardiness of a set of independent jobs on a set of unrelated-parallel machines. Logendran et al. (2007) developed different algorithms based on tabu search to minimize the total weighted tardiness of a sequence-dependent job scheduling problem on unrelated-parallel machines with consideration of dynamic job release and machine availability times. In another work, Logendran and Subur (2004) studied a different problem on unrelated-parallel machine scheduling with job splitting. Rocha et al. (2008) also developed a B&B algorithm to minimize a linear combination of makespan and total weighted tardiness of a sequence-dependent job scheduling problem on unrelated-parallel machines. Fanjul-Peyro and Ruiz (2010) proposed a set of iterative greedy local search based meta-heuristics to minimize the makespan of a set of independent jobs on unrelated-parallel machines. Gairig et al. (2007) developed a 2-approximaiton algorithm to minimize the makespan of a set of independent jobs on unrelated-parallel machines. Regarding the objective function, Huo et al. (2007) studied a single-machine problem to minimize the number of tardy jobs and maximize the weighted tardiness. Eren and Guner (2008) considered a two-machine flow shop scheduling problem to minimize the total weighted completion time together with the makespan of the sequence. Mohri et al. (1999) studied the minimization of maximum completion time and maximum lateness on three identical-parallel machines. Mehravaran and Logendran (2011) considered an unrelated-parallel machine job scheduling problem to *jointly* minimize the work-in-process inventory and maximize the customer service level in a supply chain. Lu and Logendran (2011) tried to minimize a linear combination of total weighted completion time and total weighted tardiness in a sequence-dependent flow shop group scheduling problem.

Besides the above mentioned papers, there are a few others that have investigated the unrelated-parallel machine scheduling problems along with batch (family) setup times (Chen and Wu (2006); Kim et al. (2002); Kim et al. (2003); Arnaout et al. (2006); Akkiraju (2001); Jeong et al (2001)). We emphasize that these investigations focus on job scheduling, and not group scheduling as reported in this paper. Therefore, they do not follow the GTA, which is indeed the case in many real world problems.

To the best of our knowledge, the problem of group scheduling together with a bi-criteria objective function on an unrelated-parallel machines environment has not been studied so far. Since this problem is well motivated by industry applications, we investigate it comprehensively in this paper. Also, the dynamic release of jobs and dynamic availabilities of machines have been considered for accurately modeling real world situations.

3. Problem statement

Consider the problem of scheduling N different jobs on m unrelated-parallel machines. All jobs are clustered in g different groups where each group contains n_i jobs and $\sum_{i=1}^{g} n_i = N$. Each job needs to be processed only by one machine, and all jobs of a group need to be processed on the same machine (GTA assumption). Different groups can be processed on different machines, but not all machines are capable of processing all jobs. Realistically, a group cannot be processed on a machine if there is at least one job in the group which cannot be processed on that machine.

The job release times are considered to be dynamic, that means not all the jobs are ready to be processed at time zero. The machine availability times are also considered to be dynamic, i.e. not all the machines are available at the beginning of the current planning horizon or time zero.

The objective of the problem is to *simultaneously* optimize both the total weighted completion times and total weighted tardiness of jobs, thus emphasizing a bi-criteria focus. These two criteria are merged with the help of α and β, where α reflects the importance or weight of producer and β reflects the importance or weight of customers. As these weights are assumed to be normalized, α and β should add to 1. The parameters, decision variables, and the model in itself are described below.

3.1. Parameters

g	Number of groups
b_{max}	Maximum number of jobs in each group; Despite the fact that each group can be composed of different number of jobs, in order to simplify the development of the model,

the same number of jobs (b_{max}) is considered for all groups. If one group contains less than this maximum number of jobs, some dummy jobs (with processing time of zero) will be considered to make up for the difference in that group.

b_i Number of jobs in group i

m Number of machines

$$t_{ijh}\begin{cases} 1)\ run\ time\ of\ job\ j\ in\ group\ i\ on\ machine\ h & if\ job\ j\ in\ group\ i\ can\ be\ processed\ by\ machine\ h \\ 2)\ M & if\ job\ j\ in\ group\ i\ cannot\ be\ processed\ by\ machine\ h \end{cases}$$

$i = 1, 2, \ldots, g;\ j = 1, 2, \ldots, b_{max};\ h = 1, 2, \ldots, m$

S_{pih} The setup time for group i on machine h if group p is the preceding group

$p = 0, 1, 2, \ldots, g; i = 1, 2, \ldots, g\ (i \neq p);\ h = 1, 2, \ldots, m$

($p = 0$ refers to the reference group or the last group in the previous planning horizon to be processed on the same machine.)

d_{ij} Due date of job j in group i; $i = 1, 2, \ldots, g; j = 1, 2, \ldots, b_{max}$

r_{ij} Release time of job j in group i; $i = 1, 2, \ldots, g; j = 1, 2, \ldots, b_{max}$

a_h Availability time of machine h; $h = 1, 2, \ldots, m$

w_{ij} The weight of job j in group i regarding the objective function;

$i = 1,2, \ldots, g; j = 1,2, \ldots, b_{max}$

α The weight of jobs regarding the producer

β The weight of jobs regarding the customers

3.2. Decision Variables

x_{ij} The completion time of job j of group i; $i = 1, 2, \ldots, g; j = 1, 2, \ldots, b_{max}$

Y_{ijq}	$\begin{cases}1 & \textit{if job } q \textit{ is processed after job } j \textit{ in group } i \\ 0 & \textit{Otherwise}\end{cases}$		i=1,2,…,g j,q=1,2,…,b_{max};j<q
z_{ih}	$\begin{cases}1 & \textit{if group } i \textit{ is assigned to machine } h \\ 0 & \textit{Otherwise}\end{cases}$		i=1,2,…,g h=1,2,…,m
C_i	The completion time of $group\ i$		$i = 0, 1, 2, \dots, g$
A_{pi}	$\begin{cases}1 & \textit{if group } i \textit{ is processed after group } p \\ 0 & \textit{Otherwise}\end{cases}$		$i, p = 1, 2, \dots, g; p < i$
TD_{ij}	$\begin{cases}x_{ij} - d_{ij} & \textit{if } x_{ij} - d_{ij} > 0 \\ 0 & \textit{Otherwise}\end{cases}$		i=1,2,…,g j=1,2,…,b_{max}

(TD_{ij} represents the tardiness of job j in group i)

3.3. Mixed-Integer Linear Programming Model (MILP)

$$Min\ Z = \alpha \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} w_{ij} x_{ij} + \beta \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} w_{ij} TD_{ij} \tag{1}$$

Subject to:

$$x_{ij} + M\left(1 - A_{pi}\right) + M(1 - z_{ih}) + M\left(1 - z_{ph}\right) \geq C_p + S_{pih} + t_{ijh} \tag{2}$$

$i, p = 1, 2, \dots, g; p < i;\ j = 1, 2, \dots, b_{max};\ h = 1, 2, \dots, m;\ M\text{: } a\ large\ number$

$$x_{pj} + M(A_{pi}) + M(1 - z_{ih}) + M\left(1 - z_{ph}\right) \geq C_i + S_{iph} + t_{pjh} \tag{3}$$

$i, p = 1, 2, \dots, g; p < i;\ j = 1, 2, \dots, b;\ h = 1, 2, \dots, m;\ M\text{: } a\ large\ number$

$$x_{ij} \geq \sum_{h=1}^{m} (a_h + S_{0ih} + t_{ijh}) z_{ih}\,; \qquad i = 1, 2, \dots, g;\ j = 1, 2, \dots, b_{max} \tag{4}$$

$$x_{ij} \geq \sum_{h=1}^{m} (r_{ij} + t_{ijh}) z_{ih}\,; \qquad i = 1, 2, \dots, g;\ j = 1, 2, \dots, b_{max} \tag{5}$$

$$x_{ij} - x_{iq} + M(Y_{ijq}) \geq \sum_{h=1}^{m} t_{ijh}(z_{ih}) \tag{6}$$

$i = 1, 2, \dots, g;\ j, q = 1, 2, \dots, b_{max}; j < q;$

$$x_{iq} - x_{ij} + M(1 - Y_{ijqk}) \geq \sum_{h=1}^{m} t_{iqh}(z_{ih}) \tag{7}$$

$i = 1, 2, \dots, g;\ j, q = 1, 2, \dots, b_{max}; j < q$

$\sum_{h=1}^{m} z_{ih} = 1\,; \qquad i = 1, 2, \dots, g$ (8)

$C_i \geq x_{ij}; \qquad i = 1, 2, \dots, g;\ j = 1, 2, \dots, b_{max}$ (9)

$TD_{ij} \geq x_{ij} - d_{ij}; \qquad i = 1, 2, \dots, g;\ j = 1, 2, \dots, b_{max}$ (10)

$x_{ij}, TD_{ij} \geq 0;\ z_{ij} \in \{0,1\};\ A_{pi} \in \{0,1\}\ (p < i);\ Y_{ijq} \in \{0,1\}$, (j<q)

$i, p, l = 1, 2, \dots, g;\ j, q = 1, 2, \dots, b_{max};\ \ h = 1, 2, \dots, m$

The objective function (1) minimizes the summation of weighted completion times and weighted tardiness. Constraints (2) and (3) are incorporated to find the sequence of groups. These constraints restrict the completion time of each job in each group to be greater than the completion time of the previous group plus the sequence-dependent machine setup time, and the required runtime for processing the job. These constraints are active if and only if both groups are processed on the same machine. Simultaneously these constraints assign values to A_{pi} variables, which determine the sequence of groups. Constraint (4) has been embedded to ensure that the completion time of each job in each group on any machine is greater than the machine availability time, plus setup time of that group on that machine, and the runtime of the job. This constraint considers the reference setup time, which is the setup time of a group on a machine when that group is the first group being processed, following the reference group, on that machine. All the other setup times have been considered in constraints (2) and (3). Constraint (5) ensures that the completion time of a job is greater than the job release time plus its runtime. Constraints (6) and (7) help to find the sequence of different jobs in each group. They ensure that the difference between the completion times of any two jobs in a group is greater than the processing time of the succeeding job. Constraint (8) restricts each group to be processed only on one machine. Constraint (9) computes the completion time of each group, which is greater than the completion times of all of its jobs. And finally constraint (10) is incorporated to find the tardiness of each job, which should be greater than or equal to both the completion time minus due date and zero.

4. Search Algorithm

It has been shown that a single-machine problem with sequence-dependent setup times and a bi-criteria objective function is amongst the strongly NP-hard problems (Eren and Guner (2006)). Since the problem investigated in this paper can be easily reduced to the above mentioned problem, it can be concluded that it too is strongly NP-hard. Therefore, there is no algorithm to find the optimal solution of this problem in polynomial time. This is the main characteristic of most complex scheduling problems, which made us investigate into developing a meta-heuristic algorithm for solving the research problem. Although exact algorithms such as branch-and-bound can be used to find the optimal solution for small size problems, they are computationally very inefficient for optimally solving large size (or industry size) problems or even identifying a good feasible solution.

Tabu search, introduced by Glover (1986), has been shown to be remarkably effective in solving combinatorial optimization problems (Glover and Laguna (1997); Logendran and Sonthinen (1997)). It is an iterative search which tries to overcome the limitations of local optimality. TS starts with an initial solution (IS), and moves through the neighborhood of this solution to find the best neighbor; then repeats this process to find better solutions. In the following, an IS finding mechanism is described, and then a detailed explanation of the tabu search algorithm, which best suits the problem reported in this paper is presented.

4.1. Initial Solution

Tabu search needs an IS to trigger the search. This IS can be any feasible solution, but Logendran and Subur (2004) showed that the quality of the final solution is sensitive to the quality of the IS. Therefore, two different mechanisms are developed here in order to identify effective initial solutions.

Since the objective function of the problem is bi-criteria, there are two different aspects in generating an effective sequence: producer's interest and customers' interest. From producer's point of view, the total weighted completion time of jobs should be minimized; and from customers' perspective, the total weighted tardiness of jobs needs to be minimized. In a single-machine job scheduling problem with setup and run times combined to evaluate a processing time, the first objective can be optimally attained by WSPT (weighted shortest processing time) rule, and an effective solution to the second objective can be attained by using WEDD (weighted earliest due date) order. Therefore, these two rules have been used as effective heuristics to identify the IS for the problem of this paper.

In order to deal with the bi-criteria objective function, two different sequences are generated: producer's sequence (PS) and customers' sequence (CS). These sequences are then merged to find a final sequence as the IS of the search, similar to the approach used by Mehravaran and Logendran (2011). This merging takes place by normalizing the positional values of PS and CS. In other words, the final sequence (as the IS) will be defined as $\alpha.PS + \beta.CS$ where $\alpha + \beta = 1$. It is worth noting that all these sequencing will have to be performed on two levels: the group level which finds the sequence of groups, and the job level which finds the sequence of jobs within each group. In the first proposed IS finding mechanism, the average run time of each group has been used as an adjustment in finding the groups' order of CS; but in the second mechanism, job release time has been used as an adjustment in both levels (group and job) of CS and PS. In both mechanisms, the group sequence will be determined first, and then with respect to this sequence, the job sequence in each group will be found. These mechanisms are described below:

Initial Solution 1 (IS1): The PS in group level for each machine is obtained from the following rule: $\bar{t}_{1h} + min\ \_S_{1h} < \bar{t}_{2h} + min\ \_S_{2h} < \cdots < \bar{t}_{ih} + min\ \_S_{ih}$; where $\bar{t}_{ih} = \frac{\sum_{j=1}^{b_i} t_{ijh}}{b_i}$ and $min\ \_S_{ih} = min_p\{S_{pih}\}$. But it follows the WSPT rule for job level: $\frac{t_{i1h}}{w_{12}} < \frac{t_{i2h}}{w_{i2}} < \cdots < \frac{t_{ijh}}{w_{ij}}$.

Also, the CS in group level follows: $\bar{d}_1 + \bar{t}_{1h} < \bar{d}_2 + \bar{t}_{2h} < \cdots < \bar{d}_i + \bar{t}_{ih}$; where $\bar{d}_i = \frac{\sum_{j=1}^{b_i} d_{ij}}{b_i}$; But it follows the WEDD rule in job level: $\frac{d_{i1}}{w_{12}} < \frac{d_{i2}}{w_{i2}} < \cdots < \frac{d_{ij}}{w_{ij}}$.

Initial Solution 2 (IS2): The PS for each machine is obtained from the following rule in group level:$\bar{t}_{1h} + min\ \_S_{1h} + \bar{r}_1 < \bar{t}_{2h} + min\ \_S_{2h} + \bar{r}_2 < \cdots < \bar{t}_{ih} + min\ \_S_{ih} + \bar{r}_i$; where $\bar{r}_i = \frac{\sum_{j=1}^{b_i} r_{ij}}{b_i}$. And the following rule is applied to the job level: $\frac{t_{i1h}+r_{i1}}{w_{i1}} < \frac{t_{i2h}+r_{i2}}{w_{i2}} < \cdots < \frac{t_{ijh}+r_{ij}}{w_{ij}}$. The CS in group level is obtained from the following rule: $\bar{d}_1 + \bar{r}_1 < \bar{d}_2 + \bar{r}_2 < \cdots < \bar{d}_i + \bar{r}_i$, and the following rule is applied to the job level: $\frac{d_{i1}+r_{i1}}{w_{12}} < \frac{d_{i2}+r_{i2}}{w_{i2}} < \cdots < \frac{d_{ij}+r_{ij}}{w_{ij}}$.

In group levels of both IS1 and IS2, the group sequence will be determined for each machine separately. Then, the groups will be assigned according to these sequences to the earliest available machine, and ties are broken in favor of the smallest machine index.

4.2. Algorithmic structure

The problem considered in this paper involves two levels of search: outside search and inside search. The former determines the order of groups on each machine, while the latter determines the order of jobs in each group for any given order of groups obtained from the outside search. Both of these levels are based on tabu search and the search process moves back and forth between these two levels as the search progresses.

Step 1: Find an initial solution, say IS. Name the group order as OIS. OIS is considered as the first seed for the outside tabu search.

Step 2: Generate the neighborhood of the outside seed by perturbing on groups, yet one move at a time. Two different types of moves are considered for this problem: exchange moves and insert moves. The exchange move simply exchanges the groups in two different positions on two different machines or on the same machine. The insert move inserts a group in another position, either on the current machine or on another machine.

Step 3 (inside tabu search): For all the solutions generated in the previous step, the inside tabu search needs to be performed in order to find the best objective function value of the related group order. For ease of understanding, this has been explained for OIS.

Similar to the outside level (steps 1 and 2), an initial order of jobs within each group is needed in order to trigger the inside tabu search. The related job order of OIS is considered as the inside IS and is referred to as IIS. This solution will be considered as the first seed of inside tabu search.

Step 3.1: Generate the neighborhood of IIS and evaluate their objective function with respect to the associated group order. The neighborhood of IIS consists of all of its neighboring solutions. The same as outside level, the moves consist of exchange moves and insert moves.

Step 3.2: Update the following parameters for the inside tabu search:

(1) Inside tabu list (ITL) and Aspiration level (IAL): The tabu list stores the recent moves for a certain number of iterations, which is referred to as the tabu list size. In each iteration, a move cannot be performed if it is still in the tabu list. This condition holds true unless a move leads to a solution better than the aspiration level. Aspiration level shows the best solution found so far. Therefore, if any move leads to a solution better than the aspiration level, disregarding it being tabu or not, the move will be made and the aspiration level will be updated to the new value.

(2) Inside candidate list (ICL): The best neighbor solution of a seed, in any iteration, is called candidate solution, and will be kept in a list named inside candidate list (ICL). The candidate solutions should not be repeated during the search. Therefore, if a neighbor solution is already stored in the ICL, it will not be considered again as a candidate. The candidate solution will be considered as the seed for the next iteration to repeat the perturbation. The IS is always the first entry into the ICL. If the next candidate has an objective function value better than the previous one, it will be assigned a star (*), which means this candidate has the potential of becoming a local optimum.

(3) Inside index list (IIL): Any local optimum found during the inside search will be kept in a list named IIL. To find the inside local optima, a candidate solution needs to be checked when it is admitted into the ICL. If the new entry into the ICL has an objective function value worse than the latest entry into this list, and if the latest entry has been already assigned a star (*), it will be assigned another star and will be considered as a local optimum. This local optimum (with two stars **) will be entered into the IIL. Similar to the ICL, the IS is always the first entry into the IIL. Unlike the ICL, IIL should have a predefined size in order to control the execution time of the search, which is referred to as Inside Index List Size (IILS).

(4) Inside number of iterations without improvement (INIWOI): In any iteration if the new entry into the ICL is not better than the latest entry into this list, increase the number of iteration without improvement for inside search (INIWOI) by one; otherwise set this number to zero. Similar to the IILS, the maximum number of iterations without improvement for inside search (MINIWOI) is also limited (and predefined), in order to control the execution time of the search.

Step 3.3: IILS together with the INIWOI act as the stopping criteria for the inside search. If these two variables reach their predefined values, the search is stopped; otherwise the latest entry to the candidate list will be considered as the next seed and the search will be directed to step 3.1.

Step 4: Repeat step 3 (inside tabu search) for all the solutions generated in step 2. Find the solution which has the best objective function value.

Step 5: Update the following parameters the same as what has been described for the inside search: Outside Tabu List (OTL), Outside Aspiration Level (OAL), Outside Candidate List (OCL), Outside Index List (OIL), and Outside Number of Iterations without Improvement (ONIWOI).

Step 6: The same as inside search, the stopping criteria need to be checked in this step. Therefore if either outside index list size (OILS) or ONIWOI reach their predefined values, the search will be stopped and the best solution found so far will be the best solution found by the tabu search; otherwise the latest entry into the outside candidate list will be considered as the next seed and the search goes to step 2.

4.3. Long Term Memory (LTM)

What has been described in the previous subsection is known as the short-term memory functionality of tabu search. Besides this functionality, tabu search also uses the long-term memory (LTM) in order to intensify the search (LTM-Max) in the regions where there is potential of identifying better solutions, or to diversify the search (LTM-Min) in the regions that have not been visited frequently in the past iterations. LTM functionality can be performed on both outside and inside tabu searches. In order to do so, a frequency matrix is needed to keep track of the number of placement of each group on any position of each machine (outside level) or the number of placement of each job on any position of each group (inside level). In each iteration of the inside tabu search, after finding the next candidate, the related cells of the inside frequency matrix need to be updated.

After the tabu search with short-term memory (either inside or outside) comes to an end, the search can be restarted using another initial solution which has been extracted from the frequency matrix. In order to find the restart point (the next initial solution to restart the search), LTM-Max selects the maximum number of the frequency matrix and places the job (inside level) or group (outside level) in the position where this max number refers to, then selects the second maximum number and continues this procedure until all the jobs or groups are placed in their positions. LTM-Min also works the same way, but in order to diversify the search to the regions that have not been visited frequently before, it selects the minimum numbers of the frequency matrix. It should be noted that the infeasible positions, as a result of some machines not capable of processing some jobs, should not be considered while selecting the minimum numbers in LTM-Min.

Laguna et al. (1993), used four restarts for their single machine scheduling problem, but preliminary investigations of test problems, revealed that 3 restarts for the outside search and 2 restarts for the inside search best fit the problem addressed in this paper. These numbers of restarts have also been successfully used previously by Gelogullari and Logendran (2010).

5. Data Generation

In this section some data generation mechanisms have been designed for different parameters of the model. When working with group scheduling problem, the workload of manufacturing cells plays an important role in determining the degree of complexity associated with the problem. Therefore, three different categories have been defined for the sample problems as: loosely-loaded, moderately-loaded, and tightly-loaded. Unlike in previous research, we define all anew the workload of cells as the ratio of number of groups to the number machines. The number of groups and number of machines are generated uniformly in [5,16] and [3,6], respectively. Then if this ratio is between 5/6 to 2^{-}, the problem is regarded as loosely-loaded; if the ratio is between 2 to 3^{-}, the problem is moderately-loaded, and finally if it is between 3 to 16/3 the problem is among the tightly-loaded problems. The number of jobs in each group is generated uniformly in U[2,6]. Job weights are also generated uniformly in [1,4].

The job release times and the machine availability times are generated from an exponential distribution with mean of 20 minutes. The machine capability is divided into three levels: least, medium and most capable (Logendran and Subur (2004); Pandya and Logendran (2010); Mehravaran and Logendran (2011)), which are eligible to process 50%, 70% and 85% of all jobs, respectively. In order to assign these capabilities to machines, if the number of machines is 3, then there will be one machine from each capability. If the number of machines is more than 3, say m, then $(m-3)$ uniform random numbers in [0,1] are generated. The numbers with a value less than 1/3, more than 2/3, and between 1/3 and 2/3 will be counted separately. The largest count will be the extra numbers of low capability machines, the smallest count will be the extra number of most capable machines, and the remaining count will be the extra number of medium capable machines. In order to generate the run times, three uniform random numbers in [1,10] are generated, $\alpha_i \ \forall\, i = 1,2,3$ for each machine type. The largest value is assigned to the least capable machines, the smallest value is assigned to the most capable machines, and the remaining one is assigned to the medium capable machines. The machine runtimes are then generated from a uniform distribution $[\alpha_i + 1, \alpha_i + 20]$.

Schaller et al. (2000) argued that the complexity of a scheduling problem is likely to depend on the ratio of setup time to run time. Following their suggestion, the setup times are generated uniformly in [1,40], [1,100] and [1,200] which approximately lead to the ratio of 2:1, 5:1 and 10:1, respectively, for the mean of setup times to the mean of run times.

Previous works (Pandya and Logendran (2010); Kim et al. (2002)) showed that defining proper due dates can positively affect the performance of the algorithms. They used two different factors in defining due dates: tardiness factor (τ), and due date range factor (R). The tardiness factor (τ) is used to create loose or tight due dates. It is defined as $\tau = 1 - \bar{d}/C_{max}$, where $\bar{d}$ is the average due date and C_{max} is the maximum completion time of all jobs. Large values of τ indicate tight due dates and small values of this factor lead to loose due dates. In addition to this factor, the due date range factor (R) controls the variability of due dates. The range factor (R) is defined as $(d_{max} - d_{min})/C_{max}$, where d_{max} is the maximum due date, among all the jobs, and d_{min} is the minimum one. Therefore, different combinations of τ and R provide different characteristics for randomly generated due dates. In this research, these factors have been set as $\tau = 0.5$ and $R = 0.2$ which provide medium and narrow range due dates. Then due dates are generated with probability of τ from uniform distribution in $[\bar{d} - R\bar{d}, \bar{d}]$, and with the probability of $(1-\tau)$ from uniform distribution in $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$ (Logendran and Subur (2004); Pandya and Logendran (2010); Kim, et al. (2002)). In order to estimate C_{max} for an unrelated- parallel machine scheduling problem, the following iterative equations have been developed:

$x_{i1} = \sum_{h=1}^{m}[\max\{r_{i1}, mr_i + \gamma \bar{S}_i\} + t_{i1h}]/m$

Where x_{i1} is the estimated completion time of job i in group 1. t_{i1h} is the run time of job 1 in group i. m is the total number of machines which are capable of processing the job. $\bar{S}_i$ is the average setup time of group i. γ was introduced by Logendran et al. (2007) as an adjustment to the average setup time. They argued that the maximum completion time is sensitive to the setup time of jobs, and using the average setup time instead of the sequence-dependent setup times would not provide an accurate estimate of C_{max}. Therefore, γ was defined as an adjustment. Furthermore, a coefficient of variation (CV) for the sequence-dependent setup times was introduced as $CV = S/\bar{x}$, where S is the sample standard deviation and $\bar{x}$ is the average. They suggested a linear relationship between γ and CV, which can be described by the interpolation of $(CV, \gamma) = (0.01, 0.9)$ and $(CV, \gamma) = (1, 0.1)$.

mr_i denotes when a machine is ready to start processing group i:

$$mr_i = \begin{cases} \min\left\{x_{p(n_p)}\right\} & \forall\, p = (i-1), (i-2), \dots, (i-m) \qquad If\ i > m \\ a_i & \qquad\qquad\qquad\qquad\qquad\qquad Otherwise \end{cases}$$

$x_{p(n_p)}$ is the estimated completion time of the last job in group p, and a_i is the availability time of machine i. Completion time of the other jobs in each group (not the first job) would be estimated as follows: $x_{ij} = \max\left\{r_{ij}, x_{i(j-1)}\right\} + \sum_{h=1}^{m} t_{ijh}\,/m$. Finally, C_{max} can be estimated as the maximum of all x_{ij}'s.

5.1. Tabu search parameter tuning

Before starting the tabu search, its parameters, i.e. tabu list size (TLS), index list size (ILS), and number of iterations without improvement (NIWOI), for both inside and outside searches, need to be evaluated. Inappropriate values for these parameters can drastically worsen the objective function value, or the computation time. Thus, a procedure needs to be developed and implemented to determine appropriate values for these parameters, which is commonly referred to as parameter tuning in published literature. Running several experiments showed that the value of each of these parameters depends on the structure of the problem, i.e. number of machines, number of groups, and average number of jobs in each group. Table 1 shows the empirical formulae which are obtained with the help of DATAFIT 7.1.44 (1995) for each of the three categories of problems defined in the previous section.

Table 1: Empirical formula for tabu search parameters

	Loosely-Loaded	Moderately-Loaded	Tightly-Loaded
OTLS	Round(0.4g-0.62m+1.5 $\bar{J}$)	Round(-0.58g+1.31m+1.48$\bar{J}$)	Round(0.28g+0.09m-0.13$\bar{J}$)
OILS	Round(0.32g+0.14$\bar{J}$)	Round(1.43g-0.91m-0.93$\bar{J}$)	Round(0.2g+3.31m-1.76$\bar{J}$)
ONIWOI	Round(0.33g-0.03m-0.14$\bar{J}$)	Round(0.49g-0.57m)	Round(0.03g+0.9m-0.26$\bar{J}$)
ITLS	Round(0.22g-0.27m+0.74$\bar{J}$)	Round(0.03g+0.19m+0.65$\bar{J}$)	Round(-0.07g+0.98m+0.15$\bar{J}$)
IILS	2	2	Round(0.02g+0.26m+0.23$\bar{J}$)
INIWOI	1	1	Round(0.33m-0.02$\bar{J}$)

g=number of groups; m=number of machines; $\bar{J}$ = the average number of jobs in each group
Round means rounding to the closest integer value.

6. Example Problem

For the purpose of better understanding, the application of the search algorithm is illustrated with an example problem. Tables 1 and 2 show the data related to this problem, which consists of three machines, five groups, and 3, 2, 2, 4 and 2 jobs in groups 1 to 5, respectively. This problem is amongst the loosely-loaded problems.

Table 2 shows the machine availability times, job weights, job release times, job due dates and job processing times on each machine, where ∞ means that the machine is not capable of processing the job. Table 3 shows the sequence-dependent setup times where group zero indicates the reference group.

Table 2: Summary data of the example problem

		Machine					
Machine Availability		M1	M2	M3			
		13	54	9			
Group	Job	Job Processing Time on Machine			Job Weight	Job Release Time	Job Due Date
1	1	∞	∞	39	3	19	128
1	2	∞	∞	6	3	57	140
1	3	∞	∞	3	2	44	128
2	1	42	26	34	3	39	139
2	2	19	45	27	2	39	133
3	1	∞	∞	25	1	5	134
3	2	17	26	38	1	27	134
3	3	15	7	17	2	40	87
4	1	∞	40	6	1	63	134
4	2	44	9	14	2	7	135
4	3	∞	39	7	2	3	140
4	4	9	32	33	3	13	132
5	1	31	34	23	2	4	68
5	2	∞	34	22	2	0	132

Table 3: Sequence-dependent setup times for the example problem

		Second Group				
Machine	First Group	1	2	3	4	5
1	0	52	51	17	58	35
1	1	0	36	30	37	60
1	2	23	0	37	5	21
1	3	60	57	0	44	21
1	4	58	35	9	0	9
1	5	1	5	6	13	0
2	0	7	46	55	1	36
2	1	0	7	34	57	31
2	2	31	0	1	25	36
2	3	56	24	0	49	39
2	4	17	46	29	0	39
2	5	13	1	27	29	0
3	0	59	8	42	13	29
3	1	0	46	34	37	41
3	2	49	0	8	38	26
3	3	14	37	0	41	41
3	4	18	51	2	0	35
3	5	24	5	50	1	0

Step 1: Use IS2 in order to find an initial solution for this problem. The PS and CS for the group level are:

PS: 2-5-4-3-1 on machine 1, 5-4-2-3-1 on machine 2, and 4-3-1-2-5 on machine 3.

CS: 5-3-4-1-2 on all three machines.

Merge PS and CS by using α and β to find a positional vector for each machine. For example, the positional vector for machine 1 is [4.6 2.6 3.2 3 1.6] where 4.6 (4.6=0.6*5+0.4*4) is the positional value of group 1 on machine 1. Therefore the final sequence of groups on machine 1 is 5-2-4-3-1. The final sequence for machine 2 and 3 are 5-4-3-2-1 and 4-3-1-5-2, respectively. Using these sequences, assign the groups to the first available machine one by one. As a result group 2 is assigned to machine 1, group 5 is assigned to machine 2, and groups 4, 3, and 1 are assigned to machine 3 in the mentioned order. This sequence is shown by OIS={2|5|4-3-1} where the vertical lines separate the groups on different machines and the dashes separate the groups on a machine. With respect to this sequence (and assignment) of groups, the PS and CS for job level need to be determined which will lead to the following sequence of jobs: IIS={1-2-3|1-2|2-3-1|4-2-1-3|2-1} where the vertical lines separate different groups and dashes separate different jobs within a group. The objective function associated with these sequences is 2816.4.

Step 2: Generate the neighborhood of OIS which are: n_1 ={2|5|3- 4- 1}, n_2 ={2|5|1- 3- 4}, n_3 ={2|5|4- 1- 3}, n_4 ={2|5|3- 1- 4}, n_5 ={2|5|1- 4- 3}, n_6 ={2|4|5- 3- 1}, n_7 ={|5- 2|4- 3- 1}, n_8 ={|5|2- 4- 3- 1}, n_9 ={|5|4- 2- 3- 1}, n_{10} ={|5|4- 3- 2- 1}, n_{11} ={|5|4- 3-1-2}, n_{12} ={2||5- 4- 3- 1}, n_{13} ={2||4- 5 -3 -1}, n_{14} ={2||4- 3- 5- 1}, n_{15} ={2||4- 3- 1- 5}, n_{16} ={2|4- 5|3- 1}, n_{17} ={2|5- 4|3- 1}. Both types of move, i.e. exchange and insert, are used in generating this neighborhood. For example, n_1 is generated from exchanging the position 1 on machine 3 with position 2 on the same machine, which can be shown by E(m3,p1,m3,p2); and n_{17}, is an insertion of group 4 (position 1 on machine 3) in position 1 on machine 2, which can be shown by I(m3,p1,m2,p2).

Step 3: Perform inside search for all of the above solutions in order to find their objective function value. In the interest of space, this step is performed only for OIS. The initial solution for the inside search is IIS= {1-2-3|1-2|2-3-1|4-2-1-3|2-1}.

Step 3.1: Generate the neighborhood of IIS and evaluate their objective function with respect to OIS. The generated neighboring solutions are: n_{01}= {2-1-3|1-2|2-3-1|4-2-1-3|2-1}, n_{02}= {3-2-1|1-2|2-3-1|4-2-1-3|2-1}, n_{03}= {1-3-2|1-2|2-3-1|4-2-1-3|2-1},n_{04}= {2-3-1|1-2|2-3-1|4-2-1-3|2-1}, n_{05}= {3-1-2|1-2|2-3-1|4-2-1-3|2-1}, n_{06}= {1-2-3|2-1|2-3-1|4-2-1-3|2-1}, n_{07}= {1-2-3|1-2|3-2-1|4-2-1-3|2-1}, n_{08}= {1-2-3|1-2|1-3-2|4-2-1-3|2-1}, n_{09}= {1-2-3|1-2|2-1-3|4-2-1-3|2-1}, n_{010}= {1-2-3|1-2|3-1-2|4-2-1-3|2-1}, n_{011}= {1-2-3|1-2|1-2-3|4-2-1-3|2-1}, n_{012}= {1-2-3|1-2|2-3-1|2-4-1-3|2-1}, n_{013}= {1-2-3|1-2|2-3-1|1-2-4-3|2-1}, n_{014}= {1-2-3|1-2|2-3-1|3-2-1-4|2-1}, n_{015}= {1-2-3|1-2|2-3-1|4-1-2-3|2-1}, n_{016}= {1-2-3|1-2|2-3-1|4-3-1-2|2-1}, n_{017}= {1-2-3|1-2|2-3-1|4-2-3-1|2-1}, n_{018}= {1-2-3|1-2|2-3-1|2-1-4-3|2-1}, n_{019}= {1-2-3|1-2|2-3-1|2-1-3-4|2-1}, n_{020}= {1-2-3|1-2|2-3-1|4-1-3-2|2-1}, n_{021}= {1-2-3|1-2|2-3-1|1-4-2-3|2-1},, n_{022}= {1-2-3|1-2|2-3-1|3-4-2-1|2-1}, n_{023}= {1-2-3|1-2|2-3-1|4-3-2-1|2-1}, n_{024}= {1-2-3|1-2|2-3-1|4-2-1-3|1-2}. The minimum objective function value in this neighborhood is 2645.4 and is associated with n_{02}.

Step 3.2: Update the parameters of inside tabu search: ITL = {E(g1,p1,p3) }, IAL = 2645.4, ICL = {{1-2-3|1-2|2-3-1|4-2-1-3|2-1}, {3-2-1|1-2|2-3-1|4-2-1-3|2-1}*}, IIL = {{1-2-3|1-2|2-3-1|4-2-1-3|2-1}}, and INIWOI = 0.

Step 3.3: The IILS is equal to 2 and the INIWOI = 1. As none of the stopping criteria have been reached yet, go to step 3.1.

Step 4: Repeat step 3 for all the solutions generated in step 2. The best neighboring solution is n_3 = {2|5|4, 1, 3} with an objective function value of 2318.4.

Step 5: Update the parameters of outside tabu search; OTL = {E(m3,p2,m3,p3)}, OAL = 2318.4, OCL = {{2|5|4, 3, 1}, {2|5|4, 1, 3}*}, OIL = {{2|5|4, 3, 1}}, and $ONIWOI$ = 0;

Step 6: OILS = Round (0.32*5+0.14*2.8) = 2 and ONIWOI = Round (0.33*5-0.03*3-0.14*2.8) = 1. As none of the stopping criteria have been reached yet, go to step 2. This example continues to one more outside iteration and then stops (ONIWOI = 1). The final solution is as follows:

Best order of groups found by outside tabu search = {2|5|4, 1, 3}

Best order of jobs found by inside tabu search = {3, 2, 1|2, 1|3, 1, 2|3, 2, 4, 1|1, 2}

Best objective function value found by tabu search = 2318.4

7. Experimental Results

Based on what has been described in section 5 (Data Generation), several experiments have been conducted in order to compare the performance of different tabu searches (STM, LTM-Max, LTM-Min), and different initial solution finding mechanisms (IS1, IS2) which are the factors of interest in this research. According to the previous works (Gelogullari and Logendran (2010); Schaller et al. (2000)) multiple factors are speculated to have effects on the performance of the search algorithms or the IS finding mechanisms. These factors are: problem structure with three levels: loosely-loaded, moderately-loaded, and tightly-loaded, setup to run time ratio with three levels: ratio = 2, 5, and 10, due date tightness (τ) with three levels: loose ($\tau = 0.2$), medium (τ=0.5), and tight ($\tau = 0.8$), and finally scenario which is the combination of coefficients of bi-criteria objective function (α, β) and that too has three levels: ($\alpha = 0.6,\ \beta = 0.4$), ($\alpha = 0.5,\ \beta = 0.5$), ($\alpha = 0.4,\ \beta = 0.6$). Therefore, there are six factors which have effects on the response variable (objective function value), and in order to find the effects of search algorithms and IS finding mechanisms, all the other four factors need to be blocked.

The questions of interest in this research are:

1. Do any of the search algorithms outperform the others?
2. Do any one of the IS finding mechanisms outperform the other?
3. Does the performance of search algorithms or IS finding mechanisms differ in different levels of other factors?

To have a better evaluation of the factors of interest, in a logical time, all levels of these factors need to be tested on the same example problem (experimental unit). Thus a split-plot design seems to be the best fit for this research as described in Montgomery (2009). The factors of interest are put in sub plot and the four other factors are placed in the whole plot. The statistical model for this design is:

$$\begin{aligned} y_{ijklmno} = {} & \mu + \tau_i + \alpha_j + \beta_k + \gamma_l + \delta_m + (\alpha\beta)_{jk} + (\alpha\gamma)_{jl} + (\alpha\delta)_{jm} + (\beta\gamma)_{kl} + (\beta\delta)_{km} \\ & + (\gamma\delta)_{lm} + (\alpha\beta\gamma)_{jkl} + (\alpha\beta\delta)_{jkm} + (\beta\gamma\delta)_{klm} + (\alpha\beta\gamma\delta)_{jklm} + \theta_{ijklm} + \varphi_n \\ & + \omega_o + (\varphi\omega)_{no} + (\alpha\varphi)_{jn} + (\beta\varphi)_{kn} + (\gamma\varphi)_{ln} + (\delta\varphi)_{mn} + (\alpha\beta\varphi)_{jkn} \\ & + (\alpha\gamma\varphi)_{jln} + (\alpha\delta\varphi)_{jmn} + (\beta\gamma\varphi)_{kln} + (\beta\delta\varphi)_{kmn} + (\gamma\delta\varphi)_{lmn} + (\alpha\beta\gamma\varphi)_{jkln} \\ & + (\alpha\beta\delta\varphi)_{jkmn} + (\beta\gamma\delta\varphi)_{klmn} + (\alpha\beta\gamma\delta\varphi)_{jklmn} + (\alpha\omega)_{jo} + (\beta\omega)_{ko} + (\gamma\omega)_{lo} \\ & + (\delta\omega)_{mo} + (\alpha\beta\omega)_{jko} + (\alpha\gamma\omega)_{jlo} + (\alpha\delta\omega)_{jmo} + (\beta\gamma\omega)_{klo} + (\beta\delta\omega)_{kmo} \\ & + (\gamma\delta\omega)_{lmo} + (\alpha\beta\gamma\omega)_{jklo} + (\alpha\beta\delta\omega)_{jkmo} + (\beta\gamma\delta\omega)_{klmo} + (\alpha\beta\gamma\delta\omega)_{jklmo} \\ & + (\alpha\varphi\omega)_{jno} + (\beta\varphi\omega)_{kno} + (\gamma\varphi\omega)_{lno} + (\delta\varphi\omega)_{mno} + (\alpha\beta\varphi\omega)_{jkno} \\ & + (\alpha\gamma\varphi\omega)_{jlno} + (\alpha\delta\varphi\omega)_{jmno} + (\beta\gamma\varphi\omega)_{klno} + (\beta\delta\varphi\omega)_{kmno} + (\gamma\delta\varphi\omega)_{lmno} \\ & + (\alpha\beta\gamma\varphi\omega)_{jklno} + (\alpha\beta\delta\varphi\omega)_{jkmno} + (\beta\gamma\delta\varphi\omega)_{klmno} + (\alpha\beta\gamma\delta\varphi\omega)_{jklmno} \\ & + \epsilon_{ijklmno} \end{aligned}$$

$i, o = 1,2$; and $j, k, l, m, n = 1,2,3$

Where μ is the overall mean effect, $\tau_i$is the replicate effect, α_j is jth level of structure, β_k represents the kth level of setup to run time ratio, $\gamma_l$is the lth level of due date tightness, $\delta_m$represents the mth level of scenario, φ_n is the nth level of search algorithm, and $\omega_o$represents the oth level of IS finding mechanism.

A total of 972 example problems have been tested and the analysis has been done on the natural logarithm transformation of the response variable, because the original values of response variable showed violations from constant variability assumption and normality assumption. But these violations were resolved completely by the natural logarithm transformation. R 2.13.0 (2011) was used to perform the analysis. The detailed ANOVA table is shown in Table 4, but the summary of statistical findings is presented here:

There is convincing evidence that there is a nonzero difference between different algorithms (p-value < 0.00001). Since the search algorithm factor has more than two levels, Tukey test is needed to compare different levels of this factor. As a result of this test, generally LTM-Max outperforms LTM-Min, and LTM-Min outperforms STM.

There is moderate evidence that IS1 works better than IS2 (p-value = 0.032). And there is moderate evidence of a non-zero difference between different levels of search algorithm per each *level of IS finding mechanism (interaction effect of search algorithm and IS finding mechanism) (p-value = 0.029);* Tukey test showed that for both IS1 and IS2, LTM-Max outperforms LTM-Min and LTM-Min outperforms STM, but when using IS2, superiority of LTM-Max is more noticeable than the other two.

There is also convincing evidence of a non-zero difference between different levels of search algorithm per each level of setup to run time ratio (interaction effect of search algorithm and setup to run time ratio) (p- value < 0.00001). The Tukey test again showed that for all three ratios of setup to run time (2, 5 and 10), the LTM-Max works better than LTM-Min and LTM-Min works better than STM. But the superiority of LTM-Max is more evident than the other two when setup to run time ratio is 2.

In addition to these effects, as it can be seen from Table 4, there are evidences for other high rank interactions (interactions of rank more than 2), but since the interpretation of these effects is really hard, they have not been mentioned in the above summary of statistical findings.

Table 4: ANOVA table for split plot design

Source (whole Plot)	df	Sum Sq	Mean Sq	F-Statistics	P-Value
Rep (or Blocks)	1	0.156	0.156	0.199846	0.656053
Str	2	250.066	125.033	160.1755	0
StoR	2	38.541	19.2705	24.68678	0
DD	2	39.361	19.6805	25.21202	0
Sc	2	6.702	3.351	4.292852	0.016945
Str:StoR	4	2.892	0.723	0.926211	0.453027
Str:DD	4	3.319	0.82975	1.062964	0.380392
StoR:DD	4	5.869	1.46725	1.879644	0.122071
Str:Sc	4	3.426	0.8565	1.097233	0.363729
StoR:Sc	4	4.158	1.0395	1.331668	0.265407
DD:Sc	4	7.028	1.757	2.250833	0.070847
Str:StoR:DD	8	5.26	0.6575	0.842301	0.568423
Str:StoR:Sc	8	5.189	0.648625	0.830931	0.578004
Str:DD:Sc	8	4.315	0.539375	0.690975	0.698327
StoR:DD:Sc	8	6.592	0.824	1.055598	0.402509
Str:StoR:DD:Sc	16	15.909	0.994313	1.27378	0.234678
Whole Plot Error	80	62.448	0.7806		
Source (Sub Plot)	df	Sum Sq	Mean Sq	F-Statistics	P-Value
IS	1	0.013	0.013	4.65106	0.031622
Str:IS	2	0.003	0.0015	0.536661	0.585113
StoR:IS	2	0.001	0.0005	0.178887	0.836267
DD:IS	2	0.006	0.003	1.073322	0.342842
Sc:IS	2	0.007	0.0035	1.252208	0.286977
Str:StoR:IS	4	0.002	0.0005	0.178887	0.949256
Str:DD:IS	4	0.009	0.00225	0.804991	0.52249
StoR:DD:IS	4	0.015	0.00375	1.341652	0.25374
Str:Sc:IS	4	0.017	0.00425	1.520539	0.195313
StoR:Sc:IS	4	0.003	0.00075	0.26833	0.89829
DD:Sc:IS	4	0.001	0.00025	0.089443	0.985732

Str:StoR:DD:IS	8	0.028	0.0035	1.252208	0.267224
Str:StoR:Sc:IS	8	0.007	0.000875	0.313052	0.961021
Str:DD:Sc:IS	8	0.006	0.00075	0.26833	0.975824
StoR:DD:Sc:IS	8	0.023	0.002875	1.0286	0.413569
Str:StoR:DD:Sc:IS	16	0.033	0.002063	0.737909	0.755078
Alg	2	0.3	0.15	53.66608	0
IS:Alg	2	0.02	0.01	3.577739	0.028825
Str:Alg	4	0.014	0.0035	1.252208	0.288286
StoR:Alg	4	0.134	0.0335	11.98542	0
DD:Alg	4	0.006	0.0015	0.536661	0.708881
Sc:Alg	4	0.002	0.0005	0.178887	0.949256
Str:IS:Alg	4	0.004	0.001	0.357774	0.838598
StoR:IS:Alg	4	0.005	0.00125	0.447217	0.774434
Str:StoR:Alg	8	0.003	0.000375	0.134165	0.997692
DD:IS:Alg	4	0.003	0.00075	0.26833	0.89829
Str:DD:Alg	8	0.046	0.00575	2.0572	0.038927
StoR:DD:Alg	8	0.012	0.0015	0.536661	0.828878
Sc:IS:Alg	4	0.006	0.0015	0.536661	0.708881
Str:Sc:Alg	8	0.041	0.005125	1.833591	0.069279
StoR:Sc:Alg	8	0.011	0.001375	0.491939	0.862012
DD:Sc:Alg	8	0.018	0.00225	0.804991	0.598467
Str:StoR:IS:Alg	8	0.006	0.00075	0.26833	0.975824
Str:DD:IS:Alg	8	0.014	0.00175	0.626104	0.755985
StoR:DD:IS:Alg	8	0.006	0.00075	0.26833	0.975824
Str:StoR:DD:Alg	16	0.046	0.002875	1.0286	0.424806
Str:Sc:IS:Alg	8	0.005	0.000625	0.223609	0.986588
StoR:Sc:IS:Alg	8	0.004	0.0005	0.178887	0.993673
Str:StoR:Sc:Alg	16	0.118	0.007375	2.638582	0.000585
DD:Sc:IS:Alg	8	0.008	0.001	0.357774	0.942061
Str:DD:Sc:Alg	16	0.092	0.00575	2.0572	0.009453
StoR:DD:Sc:Alg	16	0.084	0.00525	1.878313	0.020855
Str:StoR:DD:IS:Alg	16	0.026	0.001625	0.581383	0.898078
Str:StoR:Sc:IS:Alg	16	0.005	0.000313	0.111804	0.999995
Str:DD:Sc:IS:Alg	16	0.016	0.001	0.357774	0.990283
StoR:DD:Sc:IS:Alg	16	0.012	0.00075	0.26833	0.998171
Str:StoR:DD:Sc:Alg	32	0.142	0.004438	1.587621	0.024329
Str:StoR:DD:Sc:IS:Alg	32	0.025	0.000781	0.279511	0.999975
Subplot Error	405	1.132	0.002795		
Total	971	463.771			

Rep=Replicate; Str=Structure; StoR=Setup to Runtime Ratio; DD= Due Date
Sc=Scenario; IS=Initial Solution; Alg=Algorithm

8. Comparison to the optimal solution

In order to assess the quality of the proposed search algorithms, 15 example problems have been generated and solved by both tabu search algorithms and CPLEX 12.2 (2009). The result of these runs is summarized in Table 5. All these example problems are amongst the loosely-loaded or moderately-loaded problems, and they have different work-loads (ratio of number of groups to the number of machines).

As it can be seen from Table 5, CPLEX was able to find the optimal solution of the first 9 problems in less than half an hour. It solved problem 10, 11, and 12 optimally in around 1.5, 3 and 3.5 hours. But it took CPLEX more than 26 hours to solve problem 13 optimally, and even after more than 72 hours it was not able to find the optimal solutions for problems 14 and 15. For all these problems the proposed tabu searches were able to find a solution at least as well as what has been reported by CPLEX, but in less than 30 seconds. It may be worth noting that in all runs, the best solution found by all combinations of search algorithms and IS finding mechanisms, has been reported.

All the other workloads which are not listed here have number of groups and number of machines greater than or equal to these examples. And intuitively the greater numbers of groups and machines with the same average number of jobs will result in more complicated problems with longer computational time. Since CPLEX could not find the optimal solution for problems 14 and 15 after more than 72 hours, clearly there is no need to test problems with other workloads as they would probably take more than 72 hours to be solved.

Table 5: Comparison of the proposed tabu search and CPLEX

Example Problem	Structure	Work load	Number of groups	Number of machines	Total number of jobs	Tabu Search		CPLEX			
						Obj Func Value	Computational Time (sec)	Obj Func Value	Optimality	Lower Bound	Computational Time (sec)
1	Loose	1.00	5	5	17	3404	0.80005	3404	optimal		11.02
2	Loose	1.20	6	5	20	7000.8	0.54	7000.8	optimal		27.44
3	Loose	1.25	5	4	19	4449	0.36802	4449	optimal		48
4	Loose	0.83	5	6	19	6422.4	0.65604	6422.4	optimal		98.03
5	Moderate	2.00	6	3	20	3029	2.00011	3029	optimal		355.27
6	Loose	1.40	7	5	22	4874.4	1.67	4874.4	optimal		431.56
7	Loose	1.00	6	6	23	2834.4	3.01417	2834.4	optimal		878.56
8	Moderate	2.00	8	4	24	6684.8	3.5112	6684.8	optimal		1027.33
9	Loose	1.67	5	3	20	4945.8	0.69504	4945.8	optimal		1377.88
10	Moderate	2.33	7	3	23	11876.6	4.15824	11876.6	optimal		5903.08
11	Loose	1.75	7	4	25	8118	8.09546	8118	optimal		11053.4
12	Loose	1.50	6	4	21	6933.8	0.91605	6933.8	optimal		12559
13	Moderate	2.67	8	3	26	6607	20.5332	6607	optimal		94777.2
14	Loose	1.60	8	5	29	8327.2	8.01946	8385.4	feasible	7880.63	259714
15	Loose	1.17	7	6	32	7676.2	17.411	7696	feasible	6706.29	261768

9. Conclusions and future research

This research addressed a group scheduling problem (with GTA) in an unrelated-parallel machines environment. A bi-criteria objective function is considered to optimize the weighted sum of total weighted completion time and total weighted tardiness which benefit the producer and customers, respectively. The setup time for switching between the groups are considered to be sequence-dependent, and to better resemble the real world situations, job release times and machine availability times are considered to be dynamic. A mixed-integer linear programming model has been developed to optimally solve this problem. Since this is an NP-hard problem, 3 different meta-heuristic algorithms based on tabu search, together with two different IS finding mechanisms to initiate the search, are developed to find the optimal/near optimal solution.

Comprehensive methods have been described in order to generate sample problems, which truly represent the real world problems. Sample problems are divided into three categories with respect to the load of the workshop, i.e. the ratio of number of groups to the number of machines. The results of proposed tabu search are compared to the optimal solutions obtained from CPLEX for 15 example problems in loosely-loaded and moderately-loaded categories. Even though for some of these problems it took CPLEX several hours to find the optimal solution (in some examples it could not find the optimal solution even after 72 hours), the proposed algorithms found solutions at least as good as CPLEX in less than 30 seconds for all the example problems. This argues strongly in favor of the efficacy and efficiency of the proposed algorithms, but to better assess the performance of the algorithms even in medium and large problems, we recommend developing a lower bounding mechanism as a future area for research.

A total of 972 experiments have been conducted to identify which algorithms outperform the others, and which IS finding mechanism works better than the other. A split-plot experimental design helped to truly block the effects of the other parameters of the problem. As a result, it turned out that in general LTM-Max outperforms the other two algorithms, LTM-Min outperforms STM, and IS1 is mostly superior to IS2.

Acknowledgment: The research reported in this paper is funded in part by the National Science Foundation (USA) Grant no. CMMI-1029471. Their support is gratefully acknowledged.

References

Akkiraju, R. (2001). An agent-based approach for scheduling multiple machines. *Applied Intelligence, 14,* 135-144.

Allahverdi, A., NG, C., Cheng, T., & Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187,* 985–1032.

Arnaout, J., Rabadi, G., & Mun, H. (2006). A dynamic heuristic for stochastic unrelated parallel machine scheduling problem. *International Journal of Operations Research, 3,* 136-143.

Chen, J., & Wu, T. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equiment constraints. *Omega, 34,* 81-89.

DATAFIT, Version 7.1.44 (1995). Oakdale Engineering.

Eren, T., & Guner, E. (2006). A bicriteria scheduling with sequence-dependent setup times. *Applied Mathematics and Computations, 179*, 378-385.

Eren, T., & Guner, E. (2008). A bicriteria flowshop scheduling with a learning effect. *Applied Mathematical Modeling, 32*, 1719-1733.

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research, 207*, 55-69.

Gairig, M., Monien, B., & Woclaw, A. (2007). A faster combinatorial approximaiton algorithm for scheduling unrelated parallel machines. *Theoritical Computer Science, 380*, 87-99.

Gelogullari , C., & Logendran, R. (2010). Group-schedulingproblems in electronics manufacturing. *Journal of Scheduling, 13*, 177-202.

Glover, F. (1986). Future Paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13,* 533-549.

Glover, F., & Laguna, M. (1997). Tabu Search. *Boston:Kluwer Academic*.

Graham, R., Lawler, E., Lenstra, J., & Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5,* 287-326.

Huo, Y., Leung, J., & Zhao, H. (2007). Bi-criteria scheduling problems: Number of tardy jobs. *European Journal of Operational Research, 177,* 116-134.

IBM. (2009). ILOG CPLEX Optimization Studio, Version 12.2.

Jeong, B., Kim, S., & Lee, Y. (2001). An assembly scheduler for TFT LCD manufacturing. *Computers & Industrial Engineering, 41,* 37-58.

Kim, D., Kim, K., Jang , W., & Chen, F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing, 18*, 223-231.

Kim, D., Na, D., & Chen, F. (2003). Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing, 19*, 173-181.

Kuo, W., & Yang, D. (2006). Single-machine group scheduling with a time-dependent learning effect. Computers & Operations Research, *33*, 2099-2112.

Laguna, M., Barnes, J., & Glover, F. (1993). Intelligent scheduling with tabu search: An applicaiton to jobs with linear delay penalties and sequence-dependent setup costs and times. *Applied Intelligence, 3*, 159-172.

Li, S., Ng, C., & Yuan, J. (2011). Group scheduling and due date assignment on a single machine. *Internaitonal Journal of Production Economics, 130*, 230-235.

Liaw, C., Lin, Y., Cheng, C., & Chen, M. (2003). Scheduling unrelated parallel machines to minimize total weighted tardiness. *Computers & Operations Research, 30*, 1777-1789.

Lin, Y., Pfund, M., Fowler, J., & Montgomery, D. (2006). Classification of parallel machine environments under various correlation structures. *International Conference on Computers and Industrial Engineering, Taipei, Taiwan*, 1253-1261.

Lin, Y., Pfund, M., & Fowler, J. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research, 38*, 901-916.

Logendran , R., & Nudtasomboon, N. (1991). Minimizing the makespan of a group scheduling problem: a new heuristic. International Journal of Production Economics, *22*, 217-230.

Logendran, R., Mai, L., & Talkington, D. (1995). Combined heuristics for bi-level group scheduling problems. *International Journal of Production Economics, 38*, 133-145.

Logendran, R., & Sonthinen, A. (1997). A tabu search-based approach for scheduling job shop type flexible manufacturing systems. *Journal of Operational Research Society, 48*, 264-277.

Logendran, R., & Subur, F. (2004). Unrelated Parallel machine scheduling with job splitting. *IIE Transaction, 36*, 259-372.

Logendran, R., Carson, S., & Hanson, E. (2005). Group scheduling in flexible flow shops. *Internaitonal Journal of Production Econnomics, 96*, 143-155.

Logendran, R., Salmasi, N., & Sriskandarajah, C. (2006a). Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. *Computers & Operations Research, 33,* 158-180.

Logendran, R., deSzoeke, P., & Barnard, F. (2006b). Sequence-dependent group scheduling problems in flexible flow shops. *International Journal of Production Economics, 102,* 66-86.

Logendran, R., McDonell, B., & Smucker, B. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers & Operations Research, 34*, 3420-3438.

Lu, D., & Logendran, R. (2011). Bicriteria flow shop group scheduling with sequence-dependent setups. Proceedings (CD-ROM), 20th Annual Industrial Engineering Research Conference (IERC), Reno, NV.

Mehravaran, Y., & Logendran, R. (2011). Bicriteria supply chain scheduling on unrelated parallel machines. *Journal of Chinese Institute of Industrial Engineers, 28,* 91-101.

Mohri, S., Masuda, T., & Ishii, H. (1999). Bi-criteria scheduling problem on three identical parallel machines. *International Journal of Production Economics, 60-61*, 529-536.

Montgomery, D.C. (2009). *Design and Analysis of Experiments*. (7th ed.). New York: Wiley, (chapter 14).

Pandya, V., & Logendran, R. (2010). Weighted tardiness minimization in flexible flowshops. *Proceedings (CD-ROM), 19th Annual Industrial Engineering Research Conference (IERC), Cancun, Mexico.*

R version 2.13.0 (2011). The R Foundation for Statistical Computing.

Rocha, P., Ravetti, M., Mateus, G., & Pardalos, P. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research, 35*, 1250-1264.

Salmasi, N., Logendran, R., & Skandari, M. (2010). Total flow time minimization in a flowshop sequence-dependent group scheduling problem. *Computers & Operations Research, 37*, 199-212.

Salmasi, N., Logendran, R., & Skandari, M. (2011). Makespan minimization of a flowshop sequence-dependent group scheduling problem. *International Journal of Advanced Manufacturing Technology,* 56, 699-710.

Schaller, J., Gupta, J., & Vakharia, A. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research, 125*, 324-339.

Shahvari, O., Salmasi, N., Logendran, R., & Abbasi, B. (2011). An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research*, 1-18. doi:10.1080/00207543.2011.604051.

Xingong, Z., & Guangle, Y. (2010). Single-machine group scheduling problems with deteriorated and learning effect. *Applied Mathematics and Computation, 216*, 1259-1266.

Yang, W., & Chand, S. (2008). Learning and forgetting effects on a group scheduling problem. *European Journal of Operational Research, 187*, 1033-1044.