Non-permutation flowshop scheduling in a supply chain with sequence-dependent setup times

Yasaman Mehravaran and Rasaratnam Logendran*

School of Mechanical, Industrial, and Manufacturing Engineering

Oregon State University

Corvallis, OR 97331-6001, USA.

Abstract

In this paper, we consider a flowshop scheduling problem with sequence-dependent setup times and a bicriteria objective to minimize the work-in-process inventory for the producer and to maximize the customers' service level. The use of a bicriteria objective is motivated by the fact that successful companies in today's environment not only try to minimize their own cost but also try to fulfill their customers' need. Two main approaches, permutation and non-permutation schedules are considered in finding the optimal schedule for a flowshop. In permutation schedules the sequence of jobs remains the same on all machines whereas in non-permutation schedule, jobs can have different sequence on different machines. A linear mathematical model for solving the non-permutation flowshop is developed to comply with all of the operational constraints commonly encountered in the industry, including dynamic machine availabilities, dynamic job releases, and the possibility of jobs skipping one or more machines should their operational requirements deem that it was necessary. As the model is shown to be NP-hard, a metasearch heuristic, employing a newly developed concept known as Tabu search with embedded progressive perturbation (TSEPP) is developed to solve, in particular, industry-size

Yasaman Mehravaran, Email: mehravay@onid.orst.edu
*Corresponding Author: Rasaratnam Logendran, Tel: 1-541-737-5239, Fax: 1-541-737-2600, E-mail: Logen.Logendran@oregonstate.edu

problems efficiently. The effectiveness and efficiency of the search algorithm is assessed by comparing the search algorithmic solutions with that of the optimal solutions obtained from CPLEX in solvable small problem instances.

Keywords

Flowshop; Bicriteria; Sequence-dependent setup time; Non-permutation scheduling; Mixed-integer linear programming; Tabu search with embedded progressive perturbations

1. Introduction and Motivation

Flowshop is one of the most commonly used arrangements in production scheduling and finding an optimal schedule of jobs in a flowshop has been a great interest for researchers and practitioners alike. In a flow shop, *n* jobs have to be processed on *m* machines and every job has to be processed at most once on a machine, and each machine can only process one job at a time. Processing of a job must be completed on the current machine before processing of the job is started on the succeeding machine. Assembly lines are one of the most common examples of flow shop. Other applications of this arrangement can be found in electronics manufacturing, and space shuttle processing (Onwubolu and Davendra (2006)). In this paper, we consider a flowshop scheduling problem with sequence-dependent setup times, which means that the setup time required of a job on a machine is dependent on the previously processed job on that machine. The job release times and machine availability times are dynamic so the job and/or machine may not be available at the start of the current planning horizon. In other words, some machines may be processing the jobs released in the previous planning horizon and some jobs may not have been released yet. Although job release times and machine availability times are considered to be dynamic, the problem still remains in the off-line scheduling domain. In most industry-scheduling problems, it is not practically possible to have all jobs and machines ready and available at the start of the planning horizon. By considering this property, we ensure that the proposed problem is investigated with settings typically observed in real problems in industry. Also machine skipping is allowed and it is dealt by applying the zero processing time concept.

In finding the optimal schedule for a flowshop, two main approaches are considered, permutation and non-permutation schedules. In permutation schedules the sequence of jobs is assumed to

remain the same on all machines. Ruiz and Maroto (2005) evaluated 25 different heuristics for finding permutation schedules in order to minimize the makespan. Srikar and Ghosh (1986) developed a mixed-integer program for solving a permutation flowshop with sequence-dependent setup time with either makespan or total completion time objective. Aggoune and Portman (2006) addressed the flow shop scheduling problem with limited machine availability to minimize the makespan.

Considering non-permutation flowshop schedules is more challenging. Liao and Huang (2010) developed a non-permutation schedule for minimizing the tardiness. They modeled the problem with three different mathematical models and they used a Tabu search-based algorithm for finding the flowshop schedule, but they only use a processing time and do not separate the setup and run times in their flowshop. Ying (2008) developed a greedy heuristic for solving non-permutation flowshop with the goal of minimizing the makespan. Minimizing the makespan is difficult but, Liao and Hung (2010) stated that the complexity of minimizing the total completion time and total tardiness is even more difficult. However, Armentano and Ronconi (1999) showed that lots of manufacturers are more interested in meeting the customers' due dates.

In this paper, a bicriteria objective is presented in order to minimize both the total completion time and total tardiness. Trying to optimize two contrasting objectives shows the coordination that must be observed between the producer and the customers in a supply chain. In this research, we are trying to find a schedule in which the producer's work-in-process inventory (WIP) cost is minimized, while the customers' service level is maximized. To make the situation even more realistic, a weight is assigned to each job which shows the importance of those jobs in terms of

WIP cost and service level. The bicriteria objective is normalized by another weight assigned to each criterion. Defining different normalized weights creates different scenarios. Scenarios are identified to convert the completion time and tardiness to their relevant cost. In reality, evaluating the cost of WIP or the cost of not meeting customers' due date is hard or even impossible and so in many researches, the corresponding total completion time or tardiness are used. This is perfectly valid as the unit cost of WIP or being tardy is assumed to be fixed and does not have any effect on the objective function when considering a single criterion objective. However, this is not a true statement in bicriteria scheduling as the unit cost of WIP and jobs being tardy are not the same. The problem is that these unit costs are really difficult to evaluate. Different scenarios that contain different normalized weights will help the manufactures to find a situation (scenario) that fits them and describes their cost best.

Bicriteria objective was previously used in Eren and Güner (2006), Mansouri et al. (2009), Koksalan and Burak Keha (2003), Moslehi et al. (2009), and Choua and Lee (1999). Eren and Güner proposed a mixed-integer programming model to find the optimum schedule for a single-machine problem with sequence-dependent setup times. Mansouri et al. (2009) introduced two algorithms in a two-machine flowshop with sequence-dependent setup. Koksalan and Burak Keha (2003) considered two different bicriteria objectives for a single-machine scheduling problem. None of these researches considered dynamic job release and dynamic machine availability.

To sum it up, we present a mathematical model and develop three search algorithms to find the optimal/near optimal non-permutation sequence for a flowshop in order to minimize the

weighted sum of total weighted completion time and total weighted tardiness. Sequence-dependent setup times are considered and the jobs are allowed to skip one or more machines. The job release times and machine availability times are considered to be dynamic.

The rest of this paper is organized as follows. In Section 2, the non-permutation scheduling problem is formulated with linear mixed-integer programming. In Section 3, four different initial solution finding mechanisms are developed and then the heuristic algorithm based on Tabu search with embedded progressive perturbation (TSEPP) is described. The data generation method for developing example problems is explained in Section 4. In Section 5, the application of the search algorithm to an example problem is shown. The statistical analysis to verify the difference between the initial solution finding mechanisms and the quality of the search algorithms is performed in Section 6. Section 7 evaluates the performance of the heuristic and compares the quality of final solution with the optimal solution obtained from CPLEX, and finally, in Section 8, we conclude and present the directions for future research.

2. Mathematical Model

Parameters:

$i = 1, 2\ldots, m$ index for machines

$j = 1, 2\ldots, n$ index for jobs

a_i = availability time of machine i

r_j = release time of job j

w_j = weight assigned to job j

d_j = due date of job j

p_{ij} = runtime of job j on machine i

s_{ikj} = setup time of job j, immediately following job k on machine i. k=0 means the reference job.

α = the weight attributed to the total completion time, where $0 \leq \alpha \leq 1$

β = the weight attributed to the total tardiness, where $\alpha + \beta = 1$, meaning the weights are normalized

$$f_{ij} = \begin{cases} 1, if\ job\ j\ requires\ processing\ on\ machine\ i \\ 0, otherwise\ (skip\ the\ machine) \end{cases}$$

Variables:

c_{ij} = completion time of job j on machine i

T_j = tardiness of job j on machine

$$y_{ikj} = \begin{cases} 1, if\ job\ k\ is\ scheduled\ anytime\ before\ job\ j\ on\ machine\ i, k < j \\ 0, otherwise \end{cases}$$

The objective function and the constraints are:

$$\min \quad z = \alpha \sum_{j=1}^{n} w_j c_{mj} + \beta \sum_{j=1}^{n} w_j T_j \tag{1}$$

Subject to:

$$c_{ij} \geq f_{ij}\left(a_i + s_{i0j} + p_{ij}\right), \quad i = 1,2,\dots,m.\ j = 1,2,\dots,n. \tag{2}$$

$$c_{ij} \geq f_{ij}\left(r_j + p_{ij}\right), \quad i = 1,2,\dots,m. \quad j = 1,2,\dots,n. \tag{3}$$

$$c_{ij} \geq c_{i-1\ j} + p_{ij} f_{ij}, \quad i = 2,3,\dots,m. \quad j = 1,2,\dots,n. \tag{4}$$

$$c_{ij} - c_{ik} + M\left(1 - y_{ikj}\right) \geq f_{ij}\left(p_{ij} + s_{ikj}\right),\ i = 1,2,\dots,m.\ j = 1,2,\dots,n. k = 1,2,\dots,n-1. \tag{5}$$

$$c_{ik} - c_{ij} + M y_{ikj} \geq f_{ik}\left(p_{ik} + s_{ijk}\right),\ i = 1,2,\dots,m.\ j = 1,2,\dots,n. k = 1,2,\dots,n-1. \tag{6}$$

$$T_j \geq c_{mj} - d_j, \quad j = 1,2,\dots,n. \tag{7}$$

$$c_{ij}, T_j \geq 0, \quad i = 1,2,\dots,m. \quad j = 1,2,\dots,n. \tag{8, 9}$$

$$y_{ikj} \in \{0,1\}, \quad i = 1,2,\dots,m.\ j = 2,3,\dots,n.\ k = 1,2,\dots,n-1.\ k < j \tag{10}$$

The objective function (1) focuses on minimizing the normalized sum of weighted completion time and weighted tardiness, complying with the bicriteria requirement. Constraints (2) and (3) calculate the completion time of a job, and there are three different cases to consider in the evaluation of completion time. In case 1, the machine is available after the job is released and so the job completion time is $a_i + s_{i0j} + p_{ij}$. In case 2, the job is released after the machine is available but its release time occurs prior to the completion of the setup on the machine for that job, so here again the job completion time is calculated with $a_i + s_{i0j} + p_{ij}$. In case 3, the job is released only after its setup has been completed on the machine, so $r_j + p_{ij}$ evaluates the job completion time. Constraints (4) calculate the completion time of a job on the current machine as greater or equal to the sum of its completion time on the previous machine and its run time on the current machine. If a job is skipping the current machine, the zero processing time concept is used and the job completion time on the current machine is set equal to its completion time on the previous machine. Constraints (5) and (6) jointly ensure that two jobs cannot be processed on a machine at the same time. The job tardiness is evaluated with constraints (7). Constraints (8) and (9) define the non-negativity requirements on variables, while constraints (10) define the binary requirements on the remaining variables.

3. Heuristic

The two-machine flow shop scheduling problem with minimization of sum of weighted completion times had been proven to be strongly NP-hard (Garey et al. (1976)). Likewise, the single-machine scheduling problem with minimization of sum of weighted tardiness had also been proven to be strongly NP-hard (Lenstra et al. (1977)). It is easy to see that each of these problems is a reduced version of the research problem considered in this research, and therefore

the fact that the research problem is also strongly NP-hard follows immediately. This explains the need to develop computationally efficient search heuristics for solving large problem instances as those commonly encountered in industry practice.

Most of the previous studies consider a permutation schedule for solving the flowshop scheduling problem. Although permutation schedules do not necessarily guarantee identifying an optimal solution, as noted below in Section 3.2, this approach is valid as long as each job has an operation on each of the *m*-machines considered in the problem, and there are *n!* different possible (permutation) sequences. In this research, a job can skip one or more machines should its operational requirements deem that it was necessary, and thus there is a stringent need to consider a search space that includes the non-permutation sequences in order to identify the best solution that is not only computationally efficient but also is effective. Consequently, there is a need to consider a search space that consists of $(n!)^m$ sequences. Even with limiting to permutation consideration, finding the optimal solution efficiently among *n!* sequences can be very challenging. Thus, solving the research problem addressed in this research, which includes non-permutation schedules, seems an impossible task. This has been the sole motivator for developing a non-conventional tabu-search algorithm, namely the Tabu search with embedded perturbation or TSEPP, for simplicity, and to the best of our knowledge this is for the first time an idea of this sort has been developed and applied for solving a notoriously complex bicriteria scheduling problem. The search heuristic starts with finding an initial solution (IS) and uses it to further explore the search space to finally find the best solution for the problem, as described next.

3.1. Initial Solution

An initial solution (IS) is needed in order to trigger the search algorithm. Logendran and Subur (2004) have shown that the quality of the IS can affect the quality of the final solution and thus developing a good quality IS is very important. In this research, we have two different contrasting goals and finding an IS can be difficult, if not impossible. On the one hand we are trying to find a schedule that minimizes the total weighted completion time for the producer and on the other hand we want to minimize the total weighted tardiness for the customers. In this research, two different sequences based on producer and customers' preferences are developed and then a normalized sequence is considered to be the IS.

- Producer Sequence (PS): the goal of the producer is to minimize the completion time. The shortest processing time (SPT) is proven to minimize the total completion time in single machine scheduling problems. The SPT rule is used here to find the producer's sequence. In order to apply this rule, the first step is to change the sequence-dependent setup into sequence-independent setup. To do so, the minimum setup time of a job on a machine ($smin_{ij}$) is used as a sequence-independent setup time. The idea behind using the minimum setup is that in the optimal solution it is expected that the assignment of jobs to machines to be such that the smallest setup is used. The job with the smallest value of $(smin_{ij} + p_{ij})/w_j$ is scheduled first in this sequence, meaning that the job with the largest processing time in the numerator and that with the largest weight in the denominator is given the highest priority. Ties are broken in favor of the job with the smaller job index. As a result of using this rule, *m* different sequences are created, and the one with the smallest total weighted completion time is assumed to be the PS.

- Customer Sequence (CS): for finding the best sequence for customers, 4 different methods are created and described as follows:
    - CS1: In this sequence, the earliest due date (EDD) rule is used. EDD is proven to develop the optimal schedule for single machine problem when the goal is to minimize the total tardiness. As a result, a unique sequence for jobs is generated in which the job with the smallest value of d_j/w_j is scheduled first.
    - CS2: Logendran et al. (2007) defined critical ratio as $d_j/((savg_{ij} + p_{ij})*w_j)$ for minimizing the total weighted tardiness and they showed that in an unrelated-parallel machine scheduling problem, critical ratio can provide a better quality initial solution. The average setup ($savg_{ij}$) here is used to convert the sequence-dependent setup to sequence-independent setup time. With *m* machines in a flowshop, *m* different sequences are generated for each stage and the sequence with the smallest total tardiness is selected.
    - CS3: In this sequence like in CS2, the critical ratio is used, but minimum setup time is used to convert sequence-dependency into sequence-independency. The ratio is calculated as $d_j/((smin_{ij} + p_{ij})*w_j)$.
    - CS4: This sequence is a combination of CS1 and CS3. The critical ratio used in CS3 is changed in order to generate a unique sequence for all stages as in CS1. The cumulative processing time is used in CS4 instead of the processing time as in CS3. The ratio is thus $d_j / \left(w_j \times \left(\sum_{i=1}^{m} smin_{ij} + \sum_{i=1}^{m} p_{ij}\right)\right)$.

From normalizing the best sequence for producer and customers, the initial solution (sequence) is found. The order of jobs in the producer's sequence is multiplied by α and added to the order of jobs in the customers' sequence multiplied by β (i. e. , $\alpha. PS + \beta. CS$). The result is a normalized

positional value of jobs and from these values the IS sequence is identified. Four different IS finding mechanisms are developed by combining the PS with 4 different CSs. The ISs are PS-CS1, PS-CS2, PS-CS3, and finally PS-CS4.

3.2. Search Algorithm

The search algorithm developed here is based on Tabu search (TS), which is one of the most powerful algorithms in finding the optimal/near optimal solution in hard-combinatorial problems such as the one investigated in this research. TS starts with perturbing the IS. The perturbations consist of two different moves, namely the swap move and insert move. In a swap move, the positions of two jobs are exchanged, whereas in an insert move a job is removed from its current position and is inserted into a new position in the sequence. The corresponding objective function value, evaluated as a result of performing each move, is saved in the temporary candidate list (TCL). The best value among the TCL is then selected and is entered into the candidate list (CL). The selected value cannot enter the CL if the same configuration has already been inserted into the CL, as doing so will produce the same series of "child" solutions that were generated before. The move that resulted in an entry into the CL is a tabu and this move cannot be performed in the next set of iterations. The number of iterations that a move remains being a tabu is determined by the tabu-list size (TLS). The tabu status can be overridden in the event that the objective function value of the selected entry to be admitted into the CL is better than the aspiration level (AL). The AL is the best solution that has been found so far by the search algorithm.

If the value of the current entry into the CL is better than its previous entry (i.e., smaller in objective function value), then the configuration has the potential of becoming a local optimum and is assigned a star (*). A potential local optimum will become a local optimum when the next entry into the CL has an objective function value worse than or equal to the potential local optimum, and will be given two stars (**). The entries into the CL that receive two stars deserve to be recognized as local optima. All of the local optima are admitted into the index list (IL). The best solution among the local optima is the best final solution for the problem. The algorithm is terminated when either the number of entries into the IL or the number of iterations without improvement (II) reaches the maximum number of entries into IL (MIL) and the maximum number of II (MII). This was the application of short-term memory (STM).

Even in a typical flowshop with no job skipping, sequence-dependent setup times, and a single-objective, there is no guarantee that the optimal solution would be obtained by limiting the investigation to include only the permutation sequences (Sikar and Ghosh (1986)). As noted before, this is further attested to by the fact that, if non-permutation sequences are also considered, there are a total of $(n!)^m$ sequences to consider, which is enormously large. Motivated by this fact, in evaluating the job completion time in TS, or more importantly in the identification of entries into the TCL (as well as the entries into the CL), we assume that the schedules are limited to only permutation-job sequences in all of the machines. We relax this assumption later during the search, as noted below, to include non-permutation sequences to enhance the quality of the best solution identified by the search algorithm. The decision to limit the search to only permutation sequences at this stage of the search (i.e., when identifying the entries into the TCL and CL) was made as a result of performing preliminary experiments with non-permutation

schedules, which showed that such an attempt can be computationally prohibitive even on small-size problems. In other words, while we are interested in developing a computationally efficient solution algorithm that is also capable of identifying high-quality (i.e., highly effective) solutions, what we do not want is to identify such solutions by investing prohibitively high computational times, as such algorithms would have very little use in industry practice.

The search algorithm that we develop in this research is referred to as Tabu search with embedded progressive perturbations (TSEPP). The reason it is termed TSEPP is that it allows for the possibility of performing perturbations on the best permutation sequence identified as described above, thus resulting in non-permutation sequences on machines 2 and after (i.e., through machine *m*) and eventually improving upon the quality of the solution. The TS provides the optimal/near optimal permutation schedule. This may not lead to identifying the optimal or even the near-optimal solution when stage skipping is allowed as in this research. The best permutation schedule found from the application of TS is assumed to be the sequence for the first machine/stage, and adjacent pair-wise swap moves are performed for machines 2 through *m*. Suppose that there were 4 jobs in a problem and the best permutation sequence identified by the end of the application of TS is 1-3-2-4. It is important to note that when the pair-wise adjacent swap moves are performed on 1-3-2-4 for machine 2, the permutation sequence 1-3-2-4 is preserved for machine 1 and machines 3 through *m*. Suppose that the best solution identified after performing the perturbations on sequence 1-3-2-4 as a seed for machine 2 is 3-1-2-4, which is now a non-permutation sequence. Subsequently, when adjacent pair-wise swap moves are performed for machine 3 using the permutation sequence 1-3-2-4 as a seed to generate its best non-permutation sequence, it is important to note that the sequence 1-3-2-4 and 3-1-2-4 will be

preserved for machines 1 and 2, and the permutation sequence 1-3-2-4 will be preserved for machines 4 through *m* and so on, until the non-permutation sequence, if any, that is applicable for machine *m* is identified. When such embedded progressive perturbations are performed on machines 2 and after, yet one machine at a time, we are seeking to identify a better objective function value if one is available.

There are four different ways of applying the TSEPP to the TS: 1. Applying it only to the final best solution obtained from TS, 2. Applying it to all of the entries inserted into the IL, 3. Applying it to the entries inserted into the CL, and 4. Applying it to the entries inserted into the TCL. Clearly, the most improvement would be obtained when the TSEPP is applied to the TCL entries and the least improvement would be when the TSEPP is applied only to the final best solution. As described above, preliminary experiments conducted by applying the TSEPP to TCL and CL resulted in prohibitively excessive computation times even on small problem instances as there is no restriction on the number of entries into the TCL and number of entries into the CL, determined by the number of jobs and machines in a problem. In this research we are interested in developing an algorithm, which is capable of solving large industry-size problems efficiently to identify effective solutions. Thus the TSEPP is applied to the entries into the IL.

In order to improve upon the quality of the solutions even further, the use of long-term memory (LTM) is exploited to restart the search algorithm. LTM-min diversifies the search into the region that has not been explored before and LTM-max intensifies the search into the region that has been explored before more frequently. A frequency matrix is used to keep track of the position of jobs in all configurations entered into the CL. The largest first row-wise value of the

matrix is selected in LTM-max and the selected job is fixed to its position until the end of the search. Likewise, in LTM-min the smallest first row-wise value is selected. The following shows the pseudo code for the TSEPP, and for further illustration the flowchart for the search algorithm is depicted in Figure 1.

```
Find the IS
Seed ← IS
AL←IS
While (restart≤2)
{
    While (IL<MIL||II<MII)
    {
        Do
        {
            Perform the perturbation (Swap/ Insert moves) only on the first stage of the seed
            TCL ← new solution
        }
        Find the best entry among TCL
        If (best move ≠ tabu move)
        {
            CL ← best solution
            Tabu move ← best move
            Initialize TCL
        }
        Else
        {
            Find the next best TCL
            CL ← next best solution
            Tabu move ← next best move
            Initialize TCL
        }
        If (CL is better than seed)
        {
            Assign a star (*) to the CL
            Seed ← CL
            AL←CL
            Update frequency matrix
        }
        Else
        {
            If (Seed has a star)
            {
```

```
                Assign another star to the seed (**)
                IL ← seed
                Seed ← CL
                Update frequency matrix
                Perform progressive perturbations
                Do
                {
                    Perform adjacent neighborhood search on the next stage of IL
                    If (the new configuration has a better value than IL)
                    {
                        Replace the configuration of that stage and the remaining stages with the
                        new configuration
                    }
                }
            }
            Else
            {
                Seed ← CL
                Update frequency matrix
            }
        }
    }
    Identify restart point using frequency matrix
    Initialize II, IL, CL, and frequency matrix
}
```

4. Data Generation

The algorithm introduced in this paper is applied to 8 different example problems, under 3 different structures. The number of jobs and machines define the structure of the problem. In small structure, the number of jobs and machines are generated from a uniform distribution in [8, 20] and [3, 5], respectively. The number of jobs and machines in medium structure are from U[21, 45] and U[6, 10], and in large structure, the number of jobs are from U[46, 60] and number of machines are from U[11,15]. A total of 24 different example problems were generated. There are three different scenarios applied to each example problem that defined the role of the producer and customers in decision making (scheduling). In 0.8-0.2 scenario, the

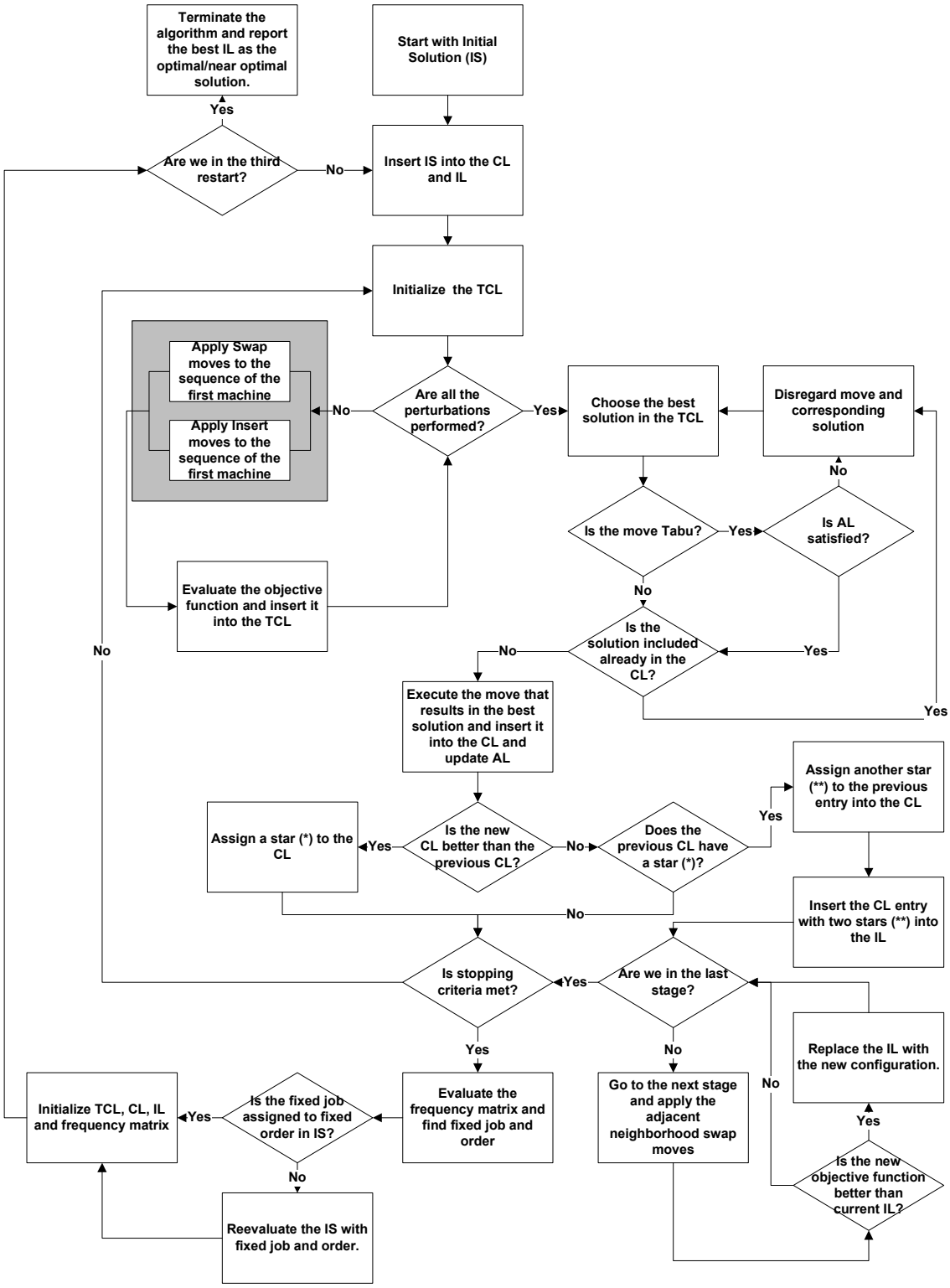

Figure 1. TSEPP flowchart

producer has 80% importance and the customers' share is only 20%. In 0.5-0.5, both the producer and customers have an equal importance, and finally in the scenario with 0.2-0.8, the customers are assumed to be more important.

Data generation is done in a similar way to that in Logendran et al. (2007) study. They introduced an unrelated-parallel machine structure whereas in this research the machines are arranged in a flowshop. The job setup times and run times are generated uniformly in [2, 40] and [1, 40], respectively and 20% of jobs are assumed to skip each stage. The skipped jobs are randomly assigned. An exponential distribution with an average equal to 20 is used to generate the job release times and machine availability times. The series of examples developed in this research have tight and narrow due dates. Due date tightness is defined as $\tau = 1 - \bar{d}/C_{max}$, where $\bar{d}$ is the average due date. The due date range is evaluated as $R = (d_{max} - d_{min})/C_{max}$, where d_{max} and $d_{min}$are the maximum and the minimum due dates. Both τ and *R* are set to 0.2. The due date is then generated from a composite uniform distribution with probability of τ the due date is from $U[\bar{d} - R\bar{d}, \bar{d}]$ and with probability *(1- τ)* it is from$U[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$. The maximum estimated completion time of job *j* in each stage *i* is evaluated with $C_{\max ij} = max[r'_{ij}, a_i + (\delta \times savg_{ij})] + p_{ij}$, where *r'ij* = updated release time of job *j* in stage *i*. Because this is a flowshop, the job release time is updated after the first stage. The updated release time in each stage is assumed to be equal to its completion time in the previous stage ($r'_{i+1\,j} = C_{\max ij}$). In this calculation the average setup time is used. In reality the best schedule tends to use the smallest setup times in transferring from one job to another, so the average setup time adjuster (δ) is introduced. For evaluating δ, the coefficient of variation for each job on each machine is

calculated with $CV = s/\bar{x}$, where s: standard deviation and $\bar{x}$: mean. If the problem was sequence-independent, then *CV*=0 and the adjuster would be equal to 1. Thus *CV*=0.01 is equivalent to $\delta = 0.9$ and *CV*=1.0 is equivalent to $\delta = 0.1$. A linear conversion to find the value of δ from *CV* is used. If the *CV*s evaluated for a machine were bigger than 1, the biggest value should be set equal to 1 and the other *CV*s should be normalized. The maximum estimated completion time (C_{max}) is evaluated as $C_{max} = \sum_{j=1}^{n} C_{\max mj}$, where m is the last stage of the flowshop. Example problem 2 in small structure is shown in Tables 1 and 2.

Job	**Weight**	**Release time**	**Due date**	**Machine availability time**				
				15	15	28	15	25
				M1	**M2**	**M3**	**M4**	**M5**
				Run time				
1	2	25	245	25	0	24	35	0
2	3	14	274	35	33	35	38	25
3	3	23	398	26	29	0	0	0
4	2	14	231	40	25	35	40	38
5	2	20	258	22	39	22	30	35
6	1	20	251	23	0	33	21	27
7	2	27	279	0	0	26	25	0
8	1	26	270	32	39	22	24	0
9	1	23	245	21	38	28	30	38
10	1	15	240	30	31	34	0	21

Table 1. Example problem

For Machine M1										
j / ***j'***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	36	30	10	11	7	0	2	32	11
2	37	0	24	11	4	20	0	22	30	28
3	36	14	0	5	27	36	0	32	4	10
4	8	35	33	0	16	6	0	40	4	32
5	8	37	37	11	0	14	0	6	29	21
6	19	8	8	22	26	0	0	3	6	39
7	0	0	0	0	0	0	0	0	0	0
8	11	28	17	3	4	8	0	0	2	40
9	24	22	9	29	22	31	0	24	0	37
10	1	9	30	8	37	6	0	34	37	0
R	21	30	2	35	15	29	0	12	34	8

Table 2. Sequence-dependent setup times of jobs

4.1. Algorithm Parameter Evaluation

The parameters of the search algorithm (TLS, MIL, and MIT) should be tuned and evaluated to perform the run on the example problems. The tuning process starts with TLS. The values of MIL and MIT are set to a large number and then the value of the TLS is varied from 1 to a large number, with an increment of 1. For all different TLS, the objective function value and maximum values of entries into the IL and iteration without improvement needed for this objective function value are recorded. The list of parameter values which returned the best objective function value or within 2% of the overall best is selected next. At the end, the values are fitted using Datafit 7.1 (1995) to develop the formulae in each structure by using regression. The equations obtained from curve fitting are listed below:

- Small
  - $TLS = INT\left(\frac{J}{M^2}\right) + 1$ (11)
  - $MIL = Round\ (1.8 * (\frac{J}{2} - \frac{M}{3}))$ (12)
  - $MII = Round(1.17\frac{J}{M})$ (13)
- Medium
  - $TLS = Round(20 - 0.35(J + M))$ (14)
  - $MIL = INT(\sqrt{J.M} + \frac{47}{M})$ (15)
  - $MII = INT(4.1 + \frac{J-M}{10})$ (16)
- Large
  - $TLS = Round(\frac{1.8M^2}{\sqrt{J}} - 8)$ (17)
  - $MIL = Round(\frac{4.5J+53}{M})$ (18)
  - $MII = Round(2.2\frac{M^2}{J})$ (19)

where *J* is the number of jobs, *M* is the number of machines, *Round* means that for values with a decimal part less than 0.5 only the integer part is accepted, and for values with a decimal part greater than or equal to 0.5 the integer part plus 1 is the desired value, and *INT* is the integer part of the value evaluated.

5. Application of the Search Algorithm

The IS finding mechanisms and the search algorithm described in Section 3 are applied to the small example problem 2 generated in Section 4. For brevity, we describe the application of the search algorithm only using PS-CS1 with α and β set to [0.8, 0.2] and using LTM-min. Similar explanations hold true for identifying the solution from the other IS finding mechanisms and search algorithms on other examples. The following evaluations are obtained using PS-CS1:

- The sequence of jobs from PS on each machine and the total weighted completion time (TWC) are: Machine 1 (M1): [J7, J3, J2, J5, J1, J4, J9, J6, J10, J8], TWC= 5886; M2: [J1, J6, J7, J3, J2, J4, J5, J10, J8, J9], TWC= 5559; M3: [J3, J2, J5, J1, J7, J4, J8, J6, J9, J10], TWC= 6688; M4: [J3, J10, J2, J7, J5, J1, J4, J6, J8, J9], TWC= 6644, and M5: [J1, J3, J7, J8, J2, J5, J4, J10, J6, J9], TWC= 6155. The best sequence is on M2 with the TWC equal to 5559.
- Using CS1 gives the following job sequence [J2, J4, J1, J5, J3, J7, J10, J9, J6, J8]. The total weighted tardiness (TWT) is also equal to 1892.
- Next the normalized sequence of jobs is evaluated by applying α and β weights to the sequence of jobs. The sequence of jobs is [J1, J6, J7, J2, J3, J4, J5, J10, J8, J9] and the corresponding normalized positional values are [1.4, 3.4, 3.6, 4.2, 4.2, 5.2, 6.4, 7.8, 9.2, 9.6]. The bicriteria objective function for the initial solution is 4927.8.

The STM search algorithm starts with the IS. Swap and insert moves are performed on the initial sequence. All the possible perturbations are performed on the IS and are saved in the TCL. The first swap move is between Jobs 1 (J1) and J2 (S 1-2), so the new sequence is [J2, J6, J7, J1, J3, J4, J5, J10, J8, J9] and the objective function is equal to 5539.8. This value is stored in the TCL. Also, the first insert move is to remove job 1 from the first position and inserting it into the fifth position (From I 1-1 To I 1-5), which results in an objective function value equal to 5037.2. In the interest of space, only some of the records are shown in Table 3.

TCL #	Swap Moves	Objective Function Value
1	S 1-2	5539.8
9	S 1-10	5586.2
13	S 2-6	5310
19	S 3-5	5194.6
24	S 3-10	5445

TCL #	Insert Moves		Objective Function Value
	From	To	
49	I 1-1	I 1-5	5037.2
66	I 3-5	I 3-3	4771.6
88	I 5-7	I 5-8	4984.8
100	I 7-3	I 7-1	4659.8
116	I 8-9	I 8-8	4913.8

Table 3. TCL entries

The smallest objective function value among the TCL is selected (100^{th} entry in the TCL with an objective function value 4659.8) and entered into the CL. This configuration is considered as a seed for the next perturbation. The seed is generated by removing J7 from the third position and inserting it into the first position. The opposite move (inserting J7 back into the third position) is considered a Tabu in the next iteration, as the TLS is set to 1. The entries into the CL and the IL are shown in Table 4. For evaluating the ILs from CLs in TS, because the objective function value of the first entry into the CL is better than the IS, a star is assigned to this configuration. In CL#2, the objective function value is still better than CL#1, and a star is assigned to CL#2. In the third iteration, the objective function value is worse than in CL#2 and so the second entry into the CL is considered as a local optimum and assigned another star. A CL entry with two stars is considered as a local optimum and is admitted into the IL. After finding the IL, the TSEPP algorithm verifies the best non permutation schedule from IL sequence as described in Section

3.2. In the first IL entry the best schedule is the permutation schedule and there was no improvement by using the TSEPP. This process is continued until the termination criteria (in this example the max IL entries=6) is met. The final solution is the best solution among the entries into the IL from TSEPP values. The final objective function value is 4425.6, which is the fourth entry into the IL.

CL #	Move	From	To	Objective Function value	IL #	TS Value	TSEPP Value
IS				4927.8	IS	4927.8	4927.8
1	Insert	I7-3	I 7-1	4659.8*			
2	Swap	S 1-5		4466**	1	4466	4466
3	Insert	I 9-10	I 9-7	4476			
5	Swap	S 8-10		4492			
6	Swap	S 1-9		4511			
9	Insert	I 10-10	I 10-8	4496**	2	4496	4496
10	Insert	I 2-4	I 2-66	4505.8			
11	Swap	S 8-9		4489.8*			
12	Insert	I 4-5	I 4-3	4474.6**	3	4474.6	4474.6
13	Insert	I 8-9	I 8-10	4490.6			
14	Insert	I 8-10	I 8-8	4504.6			
15	Insert	I 4-3	I 4-5	4498.8*			
16	Swap	S 1-6		4474.4*			
17	Swap	S 3-5		4435.6*			
18	Insert	I 9-10	I 9-8	4425.6**	4	4425.6	4425.6
19	Swap	S 3-7		4425.6			
20	Swap	S 8-9		4439.6			
21	Swap	S 9-10		4435.6**	5	4435.6	
22	Insert	I 10-9	I 10-6	4520.6			
23	Swap	S 1-4	I 3-1	4450.2**	6	4450.2	4450.2
24	Swap	S 3-7	I 6-10	4493.2			

Table 4. Entries into the CL and IL

After finding the solution from STM, the LTM-min search algorithm restarts the search to diversify the search to the region that has not been explored yet. In LTM-min, the smallest first row-wise value in the frequency matrix is selected and the selected job is fixed to the selected order. The frequency matrix in the first restart is shown in Table 5. In this matrix, the first best smallest row-wise value is in row 1 and column 1 with a value equal to 0. So J1 will be fixed to the first order throughout the entire restart. We need an IS to restart the search algorithm. In this

example, J1 is fixed to the first position and also in the IS J1 is assigned to the first position, so the same IS is used for restarting the search. In case that the fixed job is not assigned to fixed order in the IS, the fixed job is inserted into fixed order and the new configuration is used as the IS. The whole procedure is the same as STM. LTM-min performs two restarts for finding the best solution. The IL entries after the restarts are shown in Table 6. In this example, the best solution is obtained from the STM algorithm and is equal to 4425.6.

Order / Job	1	2	3	4	5	6	7	8	9	10
1	0	1	7	0	5	0	9	2	0	0
2	0	0	0	6	0	12	6	0	0	0
3	7	4	0	4	9	0	0	0	0	0
4	0	0	8	0	10	6	0	0	0	0
5	0	12	0	11	0	0	1	0	0	0
6	0	0	9	3	0	0	6	5	0	1
7	17	7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	6	12	6
9	0	0	0	0	0	0	2	4	6	12
10	0	0	0	0	0	6	0	7	6	5

Table 5. Frequency matrix

Restart				1			
IL#	IS	1	2	3	4	5	6
TS value	4928	4466	4496	4475	4426	4436	4444
TSEPP value	4928	4466	4496	4475	4426	4436	4444
Restart				2			
IL#	IS'	1	2	3	4	5	6
TS value	4928	4524	4426	4436	4440	4473	4521
TSEPP value	4928	4486	4426	4436	4440	4460	4521
Restart				3			
IL#	IS"	1	2	3	4	5	6
TS value	4466	4496	4475	4426	4436	4444	4450
TSEPP value	4466	4496	4475	4426	4436	4444	4450

Table 6. IL entries in LTM-min

6. Statistical Analysis

In this Section, the quality of the solutions found from IS finding mechanisms and those by the search algorithms are compared with the split-plot design described in Montgomery (2009). All the problem instances were run on a computer with Intel Core 2 Duo P8800 (2.66GHz/1066MHz FSB/3M L2 Cache) with 4GB, DDR3, 1067 MHz 2 Dimm RAM. On average, the algorithm was capable of solving small, medium and large problems in 1.93, 37.85, and 271.73 seconds, respectively, which shows the efficiency of the algorithm even in solving large industry size problems.

6.1. Initial Solution

In order to find if there is a difference between the qualities of the initial solutions found from IS finding mechanisms (PS-SC1, PS-CS2, PS-CS3, and PS-CS4), a split-plot design is developed. The problem structure and scenario are placed in the whole plot and the IS finding mechanisms are placed in the sub plot. The general model is:

$$y_{ijkl} = \mu + \tau_i + \beta_j + \gamma_k + (\beta\gamma)_{jk} + \theta_{ijk} + \delta_l + (\beta\delta)_{jl} + (\gamma\delta)_{kl} + (\beta\gamma\delta)_{jkl} + \epsilon_{ijkl} \quad \begin{cases} i = 1,2,\dots,8 \\ j = 1,2,3 \\ k = 1,2,3 \\ l = 1,2,3,4 \end{cases} \tag{20}$$

Where μ is the overall mean effect, τ_i represents the replicate effect, β_j is the effect of *j*th level of structure, γ_k represents the effect of *k*th level of scenario, θ_{ijk} is the whole-plot error, δ_l is the effect of *l*th level of IS finding mechanism, and $\epsilon_{ijkl}$is the subplot error. The ANOVA analysis is presented in Table 7.

Type III Sums of Squares

Source	Sum of Squares	DOF	Mean Square	F-Ratio	P-Value
Block (τ_i)	7.45E+09	7	1.06E+09		
Structure(β_j)	3.41E+11	2	1.70E+11	71.54	0.000
Scenario (γ_k)	1.14E+11	2	5.69E+10	23.90	0.000
Structure*Scenario $((\beta\gamma)_{jk})$	8.63E+10	4	2.16E+10	9.06	0.000
Whole Plot Error (θ_{ijk})	1.38E+11	58.00	2.38E+09		
IS (δ_l)	1.01E+08	3	3.38E+07	16.81	0.000
Structure*IS $((\beta\delta)_{jl})$	3.87E+07	6	6.44E+06	3.21	0.005
Scenario*IS $((\gamma\delta)_{kl})$	4.84E+07	6	8.07E+06	4.02	0.001
Structure*Scenario*IS $((\beta\gamma\delta)_{jkl})$	5.26E+07	12	4.38E+06	2.18	0.014
Sub Plot Error (ϵ_{ijkl})	3.80E+08	189.00	2.01E+06		
Total (corrected)	5.73E+11	287			

Table 7. ANOVA analysis performed on IS

The experiment was performed with STATGRAPHICS (2010) and the results show that there is a difference between the qualities of IS finding mechanisms, the interaction between ISs and structures, and the interaction between ISs and scenarios, at a 99% confidence interval. To uncover the difference between the qualities of ISs, Tukey's test is used. Because the interaction between the structure and the ISs and also scenarios and ISs are significant, the comparison between the means of ISs may be obscured by the interactions. So we perform the test for the means of algorithms by fixing the structure and scenario at a specific level. The results show that in small structure, PS-CS1 is the best IS finding mechanism. It beats PS-CS2 and PC-CS4, but there is no difference between PS-CS1 and PS-CS3. Both PS-CS3 and PS-CS4 outperformed PS-CS2, and there is no significant difference found between PS-CS3 and PS-CS4. PS-CS1 generates the best quality IS in medium structure and outperforms all other ISs. This performance is followed by PS-CS3, PS-CS4 and PS-CS2, although there was no significant difference found in the quality of PS-CS3 and PS-CS4. In large structure, the best performer is

also PS-CS1 and this performance is followed by PS-CS4, PS-CS3, and PS-CS2. The general comparison between the qualities of ISs is as follows: PS-CS1>>PS-CS4>>PS-CS3>>PS-CS2.

6.2. Final Solution

To uncover the difference between the qualities of the final solutions, again a split-plot design is used. By performing this experiment, we can find out if there is a difference between the qualities of solutions in terms of using different structure, scenarios, IS finding mechanisms and algorithms. In this design, structure and scenarios are placed in the whole plot and IS finding mechanisms and algorithms are placed in the subplot. The general model is:

$$y_{ijklm} = \mu + \tau_i + \beta_j + \gamma_k + (\beta\gamma)_{jk} + \theta_{ijk} + \delta_l + \omega_m + (\delta\omega)_{lm} + (\beta\delta)_{jl} + (\beta\omega)_{jm} + (\gamma\delta)_{kl} + (\gamma\omega)_{km} + (\beta\gamma\delta)_{jkl} + (\beta\gamma\omega)_{jkm} + (\beta\delta\omega)_{jlm} + (\gamma\delta\omega)_{klm} + (\beta\gamma\delta\omega)_{jklm} + \epsilon_{ijklm}$$

$$\begin{cases} i = 1,2,\dots,8 \\ j = 1,2,3 \\ k = 1,2,3 \\ l = 1,2,3,4 \\ m = 1,2,3 \end{cases} \quad (21)$$

Where μ is the overall mean effect, τ_i represents the replicate effect, β_j is the effect of *j*th level of structure, γ_k represents the effect of *k*th level of scenario, θ_{ijk} is the whole-plot error, δ_l is the effect of *l*th level of IS finding mechanism, ω_m represents the *m*th level of search algorithm, and $\epsilon_{ijklm}$is the subplot error. The ANOVA analysis is presented in Table 8.

The results from Table 8 show that there is a significant difference between the qualities of final solutions in terms of using different algorithms and also the interaction of structure and search algorithm is also significant. By applying the Tukey's test and fixing the structure at 3 levels of

small, medium and large, it was determined that in the small structure LTM-min is better than STM, but there is no significant difference between LTM-min and LTM-max, and between LTM-max and STM. In both medium and large structures, LTM-min outperformed both LTM-max and STM, and LTM-max is better than STM. In sum, LTM-min outperformed both LTM-max and STM, and LTM-max is better than STM.

Type III Sums of Squares

Source	Sum of Squares	DOF	Mean Square	F-Ratio	P-Value
Block (τ_i)	1.32E+10	7	1.88E+09		
Structure (β_j)	6.92E+11	2	3.46E+11	470.15	0.000
Scenario (γ_k)	2.39E+11	2	1.20E+11	162.57	0.000
Structure* Scenario ($(\beta\gamma)_{jk}$)	1.71E+11	4	4.28E+10	58.11	0.000
Whole Plot Error (θ_{ijk})	4.12E+10	56	7.36E+08		
Algorithm (ω_m)	7.78E+06	2	3.89E+06	18.34	0.000
IS (δ_l)	1.27E+06	3	4.23E+05	1.99	0.113
IS*Algorithm ($(\delta\omega)_{lm}$)	3.62E+05	6	6.04E+04	0.28	0.944
Structure*Algorithm ($(\beta\omega)_{jm}$)	5.11E+06	4	1.28E+06	6.03	0.000
Structure*IS ($(\beta\delta)_{jl}$)	1.91E+06	6	3.18E+05	1.50	0.175
Scenario *Algorithm ($(\gamma\omega)_{km}$)	1.37E+06	4	3.41E+05	1.61	0.170
Scenario *IS ($(\gamma\delta)_{kl}$)	7.89E+05	6	1.31E+05	0.62	0.714
Structure* Scenario *Algorithm ($(\beta\gamma\omega)_{jkm}$)	1.45E+06	8	1.82E+05	0.86	0.553
Structure* Scenario *IS ($(\beta\gamma\delta)_{jkl}$)	2.17E+06	12	1.81E+05	0.85	0.596
Scenario *Algorithm*IS ($(\gamma\delta\omega)_{klm}$)	1.42E+06	12	1.19E+05	0.56	0.875
Structure*IS*Algorithm ($(\beta\delta\omega)_{jlm}$)	1.15E+06	12	9.58E+04	0.45	0.942
Structure* Scenario *IS*Algorithm ($(\beta\gamma\delta\omega)_{jklm}$)	2.80E+06	24	1.17E+05	0.55	0.961
Sub Plot Error (ϵ_{ijklm})	1.47E+08	693.00	2.12E+05		
Total (corrected)	1.16E+12	863			

Table 8. ANOVA analysis performed on final solution

To conclude, although the results showed that there was a difference between the qualities of IS finding mechanisms, the ISs did not contribute to affecting the quality of final solution and this is in marked contrast to the previous research by Logendran and Subur (2004) with single criterion objectives.

7. Comparison with the Optimal Solution

In order to show the effectiveness and performance of the final solution obtained by search algorithms, the solutions were compared with the optimal solutions from CPLEX (2009). As the problem is NP-hard, solving medium and large problems would take prohibitively large computation times. Among the small problem instances, CPLEX was capable of solving 6 out of 8 small example problems optimally within a reasonable time (less than 8 hours which is equal to one shift in industry). The results are shown in Table 9.

Problem	Scenario	# jobs	# machines	# constraints	# binary variables	Optimal solution	CPLEX time (S)	TS best solution	TS time (S)
1	80%-20%	8	4	274	92	3088.8	8.44	3146.2	0.31
1	50%-50%	8	4	274	92	2254.5	6.11	2328.5	0.486
1	20%-80%	8	4	274	92	1401.8	6.52	1409.2	0.352
2	80%-20%	10	5	400	136	4312.6	302.75	4425.6	0.351
2	50%-50%	10	5	400	136	3180.5	711.34	3237.5	0.489
2	20%-80%	10	5	400	136	1937	116.66	2015.6	0.41
3	80%-20%	11	4	406	145	5849.6	412.8	5849.6	0.426
3	50%-50%	11	4	406	145	4037	1348.11	4037	0.591
3	20%-80%	11	4	406	145	2108.2	753.09	2108.2	0.462
4	80%-20%	11	5	513	184	7011.6	1623.11	7099.6	0.466
4	50%-50%	11	5	513	184	5011.5	21125.97	5014	0.443
4	20%-80%	11	5	513	184	2928.4	2910.67	2928.4	0.368
5	80%-20%	12	4	448	162	4666.2	1133.64	4720.8	0.444
5	50%-50%	12	4	448	162	3260	379.63	3300	0.827
5	20%-80%	12	4	448	162	1830	329.34	1890.8	0.776
6	80%-20%	14	3	374	136	6874	7634.17	6889.8	0.852
6	50%-50%	14	3	374	136	4717	23066.22	4717	1.339
6	20%-80%	14	3	374	136	2535.8	1860.53	2537.8	1.031

Table 9. Results from solving small problem instances with CPLEX

These 6 examples consisted of 3 different scenarios, and in total 18 different examples were run on the CPLEX. Each optimal solution is then compared with 12 different runs of search algorithms (4 ISs and 3 algorithms). The experimental design revealed that there is no statistical difference between solutions by using different IS finding mechanisms and only the search algorithms are meaningful, and therefore for evaluating the final solution performance only the search algorithms are used. The results show that the average difference between the optimal

solution and the LTM-min is only as low as 1.39%. This is followed by LTM-max and STM with 1.48% and 1.65%, respectively. In sum, the average difference between the best solution from search algorithms and the optimal solution is 1.10%, which attests very favorably to the high quality and effectiveness of the solutions found by the search algorithms aided by embedded progressive perturbations. Also, the average time taken by the TS to solve the problem instances is 0.58 seconds compared to 3540.51 seconds taken by the CPLEX, which shows that TSEPP is highly efficient in solving the research problem.

8. Conclusions and Future Work

In this paper, a new approach was developed for solving a non-permutation flowshop scheduling problem. The goal of the paper is to find a non-permutation schedule that minimizes producer's WIP and maximizes customers' service level. Three different scenarios were considered to demonstrate different levels of importance for the producer and customers. The flowshop is assumed to have sequence-dependent setup times and also the job release times and machine availability times are considered to be dynamic and stage skipping is allowed. The problem was then formulated as a linear mixed-integer programming model. As the model was shown to be NP-hard, a heuristic algorithm was developed to find the optimal/near optimal solutions efficiently. The search algorithm needed an IS to trigger the search, so four different IS finding mechanisms were developed based on producer and customers' preferences. The proposed heuristic has two parts. The first part is based on Tabu search and is capable of finding the optimal/near optimal permutation schedule. As the permutation schedule may not lead to identifying the optimal or the best near optimal solution for the proposed flowshop, in the second part, a progressive perturbation is embedded onto the IL solutions from TS to find the best non-

permutation schedule. The long-term memory was also applied to the heuristic to intensify or diversify the search. Eight different example problems in small, medium and large structure were generated to investigate the effectiveness of the search algorithm. The statistical analysis showed that the IS generated from PS-CS1 has the best quality and this is followed by PS-CS4, PS-CS3 and PS-CS2. However, the quality of final solution is not affected by the quality of IS. This is in contrast to the other studies with single criterion objective where the quality of final solution was directly related to the quality of IS. The experimental design revealed that there is a difference between the search algorithms used in different structures. In all structures, LTM-min developed a better quality final solution, but there was no difference between LTM-max and LTM-min and between LTM-max and STM in the small structure, whereas in medium and large structures LTM-max outperformed STM. Six small example problems, resulting in a total of 18 problems due to the three different scenarios considered, were solved optimally by CPLEX to identify the performance of the heuristic. The results showed that the average difference between optimal solutions and final solutions found by the heuristic is only 1.10%. In other words, the search algorithm is capable of finding very high quality final solution, both effectively and efficiently. Future research should focus on implementing the TSEPP on the CL entries in order to increase the quality of the final solution while maintaining its efficiency. Also developing a lower bounding mechanism to evaluate the quality of the heuristic in medium and large structures is recommended.

Acknowledgments

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

Aggoune, R., and Portmann, M.C., 2006. Flow shop scheduling problem with limited machine availability: A heuristic approach. International Journal of Production Economics, 99(1-2), 4-15.

Armentano, V.A., Ronconi, D.P., 1999. Tabu search for total tardiness minimization in flowshop scheduling problems. Computers and Operations Research, 26 (3) 219–235.

Choua, F., and Lee, C., 1999. Two-machine flowshop scheduling with bicriteria problem. Computers & Industrial Engineering, 36(3), 549-564.

Eren, T., and Güner, E., 2006. A bicriteria scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 179(1), 378–385.

Garey, M. R., Johnson, D. S, and Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research, 2(2), 117-129.

IBM, 2009. ILOG CPLEX Optimization Studio 12.2.

Koksalan, M., and Burak Keha, A., 2003. Using genetic algorithms for single-machine bicriteria scheduling problems. European Journal of Operational Research, 145(3), 543–556.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P., 1977. Complexity of machine scheduling problems. Annals of Discrete Mathematics, 1, 343-362.

Liao, L.M., Huang, C.J., 2010. Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. Applied Mathematics and Computation, 217 (2), 557–567.

Logendran, R., and Subur, F., 2004. Unrelated parallel machine scheduling with job splitting. IIE Transactions, 36(4), 359-372.

Logendran, R., McDonell, B., and Smucker, B., 2007. Scheduling unrelated parallel machine with sequence-dependent setups. Computers and Operations Research, 34(11), 3420-438.

Mansouri, A., Hendizadeh, H., and Salmasi, N., 2009. Bicriteria scheduling of a two-machine flowshop with sequence-dependent setup times. International Journal of Advanced Manufacturing Technology, 40(11), 1216–1226.

Montgomery, D.C., 2009. Design and Analysis of Experiments. Wiley, New York.

Moslehi, G., Mirzaee, M., Vasei, M., Modarres, M. and Azaron, A., 2009. Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. International Journal of Production Economics, 122(2), 763-773.

Oakdale Engineering, 1995. DATAFIT, Version 7.1.44.

Onwubolu, G., and Davendra, D., 2006. Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research, 171(2), 674-692.

Ruiz, R., and Maroto, C., 2005. A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research, 165(2), 479–494.

Srikar, B.N., and Ghosh, S., 1986. A MILP model for the n-job, m-stage flowshop with sequence dependent set-up times. International Journal of Production Research, 24(6), 1459-1474.

Stat Point Technologies INC., 2010. STATGRAPHICS Centurion XVI, Version 16.1.02.

Ying, K.C., 2008. Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. International Journal of Advanced Manufacturing Technology, 38 (3-4), 348–354.