

# A COMPARISON OF PROCEDURES FOR SCHEDULING GROUPS OF JOBS IN A FLOWSHOP TO MINIMIZE TOTAL TARDINESS

Jeffrey E. Schaller, Eastern Connecticut State University, Department Of Business  
Administration, 83 Windham ST., Willimantic, CT 06250, 860 465 – 5226,  
[SCHALLERJ@EASTERNCT.EDU](mailto:SCHALLERJ@EASTERNCT.EDU)

## ABSTRACT

This paper considers the problem of scheduling in a flowshop to minimize total tardiness when family setups exist. Three existing heuristics that were used to minimize total tardiness in flowshops were modified to handle family setups and a genetic algorithm was proposed for the problem. The heuristics were tested on problems of various sizes in terms of the number of jobs and families, number of machines, setup distributions, and due date tardiness distributions. The results of the test showed that the existing heuristics are inconsistent and the genetic algorithm is better.

**Keywords: Scheduling, Sequencing, Branch-and-Bound, Heuristics.**

## INTRODUCTION

In many operations obtaining economies of scale are gained by grouping similar jobs together. The motivation for grouping sometimes relates to the existence of changeover times, or setup times, on the machines. For example, jobs may belong to families that require the same tooling. As a result a job does not need a setup when following another job from the same family, but a known “family setup time” is required when a job follows a member of some other family.

An important consideration when sequencing and scheduling a set of jobs is completing each job on or before the customer’s due date. To address this consideration, this research seeks to identify and compare methods for sequencing a set of jobs in a flowshop with significant family setup times that will minimize the total tardiness of the jobs.

Most research on flowshops considers only permutation schedules. In this research only permutation schedules are considered.

Formally, suppose there is a set of  $n$  jobs belonging to  $F$  setup families to be processed in a flowshop. Let  $f_j$ , and  $d_j$  represent the setup family and the due date of job  $j$  ( $j = 1, \dots, n$ ) respectively. Let  $p_{jm}$ ,  $S_{jm}$ , and  $C_{jm}$  represent the processing time, setup time, and completion time of job  $j$  ( $j = 1, \dots, n$ ) on machine  $m$  ( $m = 1, \dots, M$ ). The tardiness of job  $j$ ,  $T_j$  is defined as:  $T_j = \max \{C_{jm} - d_j, 0\}$ , for  $j = 1, \dots, n$ . The objective function,  $Z$ , can be

expressed as:  $Z = \sum_{j=1}^n T_j$ .

Streams of literature that are most relevant to the problem addressed in this paper are minimizing total tardiness in a flowshop and scheduling flowshops with family setup times. [4] and [2] propose exact branch-and-bound algorithms for flowshops with two or more machines to minimize total tardiness. [7] reviewed and tested over 40 heuristic methods for the m-machine flowshop problem. Based on their tests [7] found that the neighborhood searches developed by [5] were the best heuristics and simulated annealing algorithms developed by [6] and [3] were the best meta-heuristics. [1] provides a review of flowshop scheduling research with setup times. Most of the papers identified in this review consider the objective of minimizing makespan.

## OPTIMAL BRANCH-AND-BOUND PROCEDURE

In this section a branch and bound procedure for the problem is described. This procedure is referred to as O. In the branch-and-bound procedure a node in the branching tree corresponds to a potential partial job sequence. Branching adds a job to the last unassigned position (at the end) in a partial sequence. For each of the nodes generated in the procedure, a lower bound on the sum of tardiness of all jobs is computed. A depth-first branching rule is used so the node with the most jobs in the corresponding partial sequence is selected for branching. Ties are broken in favor of the node with the minimum lower bound.

### Lower Bound for a Partial Sequence

Let  $\sigma$  represent an initial partial job sequence and  $p$  be the number of jobs in  $\sigma$ . To obtain a lower bound for the total tardiness of the completion of  $\sigma$  two elements are considered: the set of jobs included in  $\sigma$  and the set of jobs that have not yet been sequenced. The following notation is used. Let  $LC_{[jM]}(S)$  be a lower bound for the completion time of the job in position  $j$  of the sequence  $S$  on machine  $m$  and  $d_{EDD[j]}$  be the due date of the job sequenced in the  $j$ th position if the jobs are sorted in earliest due date order ( $d_{EDD[j]} \leq d_{EDD[k]}$  if  $j < k$ ).

Theorem 1 is used to help develop a lower bound on the total tardiness of the jobs in  $\sigma'$ .

$$\text{Theorem 1. } \sum_{j=p+1}^n \max \{LC_{[jM]}(S^*) - d_{EDD[j]}, 0\} \leq \sum_{j=p+1}^n \max \{C_{[jM]}(S^*) - d_{[j]}(S^*), 0\}.$$

Let  $LB(\sigma')$  equal a lower bound on the total tardiness for the job in  $\sigma'$ .

$$LB(\sigma') = \sum_{j=p+1}^n \max \{LC_{[jM]}(\sigma') - d_{EDD[j]}(\sigma'), 0\} \text{ where } d_{EDD[j]}(\sigma') \text{ are the due dates of}$$

the jobs in  $\sigma'$  sorted in earliest due date order (for  $j = 1, \dots, n'$ ).

For this lower bound to be operational a lower bound for the completion time of each job on machine  $M$  in the post partial sequence consisting of jobs in the set  $\sigma'$  is required ( $LC_{[jM]}(\sigma')$  for  $j = p+1, \dots, n$ ). Two lower bounds are computed and the greater of these lower bounds is set equal to  $LC_{[jM]}(\sigma')$ . Both lower bounds are similar to lower bounds

used in the problem without setups but these lower bounds incorporate setup times into the calculation.

### **Dominance Property**

A dominance property that helps to reduce the search tree in the branch-and-bound algorithm is used in the algorithm. The dominance property is based on a dominance property for minimizing total tardiness in a flowshop without family setups for adjacent jobs but is only applied to pairs of jobs that belong to the same setup family.

## **HEURISTIC PROCEDURES**

Four heuristic procedures are described in this section. The first three procedures are based on procedures that were found to be effective for minimizing total tardiness in flowshops without family setups.

### **Neighborhood Search Procedure**

[5] developed two neighborhood searches for minimizing total tardiness in flowshops without family setups. One of the searches used an exchange operator to define the neighborhood and the other used an insert operator. The neighborhood search in this paper uses both of these operators to define the neighborhood. This procedure is referred to as NS in this paper. First an initial sequence needs to be developed and then the sequence is improved by searching the neighborhood of the sequence. The neighborhood search process is repeated until none of the moves in a neighborhood offer an improvement. The initial sequence in this procedure is found using a method that is similar to that used in [6] with some modification to consider family setups.

**Initial Sequence.** An initial sequence is created in two steps. First a list of the jobs is created. Then an insertion algorithm is used to create a sequence. To create a list of the jobs first the families are sorted in non-decreasing order of the average of the due dates of the jobs within each family. The jobs are then sorted in non-decreasing order of due dates within each family.

**Neighborhood Search Improvement Procedure.** The neighborhood search improvement procedure attempts to improve a sequence by performing neighborhood searches. Two neighborhood searches are performed. The first search looks at all of the possible exchanges of pairs of jobs. The second search is an insertion search.

### **Parthasarathy and Rajendran (1997)'s Simulated Annealing Algorithm**

[6] developed a simulated annealing algorithm for minimizing weighted total tardiness in flowshops with sequence dependent setups. [7] modified the procedure for minimizing total tardiness in flowshops and found it to be one of the most effective procedures. The procedure starts with an initial seed sequence that was formed by sorting jobs in earliest

due date order and then uses simulated annealing to improve the sequence. A key characteristic of this algorithm is the use of a random insertion perturbation scheme (RIPS) during each iteration of the procedure. For the details of this procedure see [6]. This procedure is referred to as PRSA in this paper.

### **Hasija and Rajendran (2004)'s Simulated Annealing Algorithm**

[3] developed a simulated annealing algorithm for minimizing total tardiness in flowshops. [7] found this procedure to be one of the most effective procedures for minimizing total tardiness in flowshops. The procedure starts by creating  $m$  sequences. These sequences are created by developing modified due dates for each job on each machine and then sorting these modified due dates in EDD order. The total tardiness of each sequence is calculated and the best sequence is selected and a job-index-based-insertion-scheme (JIBIS) is used to improve the sequence. This sequence is used as the seed sequence in the simulated annealing algorithm. Two perturbation schemes are used during each iteration of the simulated annealing algorithm. The first perturbation scheme is a job-shift-based (JSB) perturbation scheme. The second perturbation scheme is called the probabilistic-step-swap (PSS) perturbation scheme. During the simulated annealing process the 10 best sequences found are archived. When the simulated annealing algorithm terminates the job-index-based insertion scheme is performed on each of the 10 archived sequences and the best one found becomes the final solution. For the details of this procedure see [3]. This procedure is referred to as HRSA in this paper.

### **Genetic Algorithm**

In this section a new genetic algorithm for the problem is described. In a genetic algorithm an initial population of chromosomes is first created and then successive populations (or generations) of chromosomes are created using some methodology until a stopping condition is met. A chromosome corresponds to a sequence of jobs. The  $j$ th gene in a chromosome corresponds to the job in the  $j$ th position of a sequence.

**Initial Population.** An initial population of 100 (population size) chromosomes (sequences) is created. Each chromosome (sequence) is created by first generating a random number between 0 and 1 for each job and then sorting the numbers corresponding to each job (lowest to highest) to create the sequence of jobs (chromosome).

**Creating the Next Generation.** A mating process is used to create a new generation from the existing population. Pairs of chromosomes in the existing population mate and create two children. The first two chromosomes in the population mate and then the next two and so on. To create children from of a pair of chromosomes two crossover points are first generated. The crossover points are randomly generated. The two children chromosomes are evaluated in terms of the total tardiness of the associated sequence. If one of the two children chromosomes has a lower total tardiness than found so far the incumbent value is updated. The two chromosomes with the lowest total tardiness (among the two children and two parents) are selected to go into the next generation and the two with the highest total tardiness are eliminated.

Stopping Criteria. Four stopping criteria are used: 1) incumbent = 0, 2) the average tardiness of the population is equal to the incumbent value. 3) The mean absolute deviation of total tardiness for the population is less than 1 % of the average total tardiness for the population. 4) A number of successive generations have been created but there is no improvement in the incumbent value.

## COMPUTATIONAL TESTS OF THE PROCEDURES

### Data and Performance Measures

The procedures described in the previous section were tested on problems of various sizes in terms of the number of families and jobs within each family, number of machines, for four sets of distributions of due date range and tightness, and three sets of distributions for family setup times. Each problem set consists of 10 problems. Two levels of  $F$  (2 and 3), three levels of  $n_f$  (3, 4, and 5) and two levels of  $m$  (4 and 8) were tested. The processing times of the jobs for each machine were generated using a uniform distribution over the integers 1 and 100. The setup times on each machine for each family were randomly generated using a uniform distribution. Three setup distributions were used: 1 and 200, 1 and 100 and 1 and 50. The due dates for the jobs were also randomly generated using a uniform distribution over the integers  $MS(1 - r - R/2)$  and  $MS(1 - r + R/2)$ , where  $MS$  is the minimum makespan found for the problem, and  $R$  and  $r$  are two parameters called due date range and tardiness factors. Four sets of these parameters were used:  $R = 0.5$  and  $r = 0.5$  (set 1),  $R = 1.0$  and  $r = 0.5$  (set 2),  $R = 0.5$  and  $r = 0.25$  (set 3),  $R = 1.0$  and  $r = 0.25$  (set 4).

The measures of performance used to evaluate the procedures for the test are CPU time required to generate a solution and the percentage error (% Error) of the total tardiness of the solution generated by each procedure. The procedures were coded in Turbo Pascal and were tested on a HP LP1965 2.4 GHz PC to obtain the CPU time.

### Results

CPU Time. The heuristic procedures proved to be efficient in solving the test problems. All of the heuristics averaged less than three seconds for all problem sets. The O procedure required more time to generate solutions and the time increased rapidly as problem size increased. The O procedure averaged over 500 seconds per problem for some of the problem sets with three families and five jobs per family (total of 15 jobs).

% Error. The GA procedure was the best on 102 of the 144 problem sets and was tied for best on 22 other problem sets. The GA procedure's average error was greater than 10 % on only one problem set (10.86 %). The NS procedure's average error was greater than 10 % on 25 problem sets. The PRSA procedure's average error was greater than 10 % on 24 problem sets. The HRSA procedure's average error was greater than 10 % on 38 problem sets.

## CONCLUSION

In this paper three heuristics that were found to be effective for minimizing total tardiness in flowshops were tested for flowshops with family setups. Also a genetic algorithm was proposed for the problem.

The procedures were tested on problems of various sizes in terms of the number of families included and the number of jobs per family, three distributions of family setups, and four sets of distributions that determine the tightness of due dates and the range of due dates. The solutions generated were compared against optimal solutions for the problems.

The results of the tests showed that the heuristics that worked well for flowshops without family setups were very inconsistent when family setups exist. The proposed genetic performed consistently well and is recommended for the problem.

## REFERENCES

- [1] Cheng, T. C. E., J. N. D. Gupta, and G Wang, "A review of flowshop scheduling research with setup times," *Production and Operations Management*, 2000, vol. 9, no. 3, pp. 262 – 282.
- [2] Chung, C. S., J. Flynn, and O. Kirca, A branch and bound algorithm to minimize the total tardiness for m-machine permutation flowshop problems, *European Journal of Operational Research*, 2006, 174, 1, 1.
- [3] Hasija, S., and C. Rajendran, Scheduling in flowshops to minimize total tardiness of jobs, *International Journal of Production Research*, 2004; 42: 2289 – 2301.
- [4] Kim, Y., Minimizing total tardiness in permutation flowshops, *European Journal of Operational Research*, 1995, 85, 541 – 555.
- [5] Kim, Y., H. G. Lim, and M. W. Park, Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process, *European Journal of Operational Research*, 1996, 91, 124 – 143.
- [6] Parthasarathy, S., and C. Rajendran, A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs – A case study. *Production Planning and Control*, 1997; 8: 475 – 483.
- [7] Vallada, E., R. Ruiz, and G. Minella, Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics, *Computers & Operations Research*, 2008; 4: 1350.