A Meta-Heuristic Algorithm for Flexible Flow Shop Sequence Dependent Group Scheduling Problem

Omid Shahvari[a], Nasser Salmasi[b], Rasaratnam Logendran[c]

[a] Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

[b] Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

[c] School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331, USA

Abstract

In this research, the flexible flow shop sequence dependent group scheduling problem (FFSDGS) with minimization of makespan as the criterion ($FF_m|fmls, S_{sd}|C_{max}$) is investigated. Since the problem is shown to be NP-hard, meta-heuristic algorithms are required to efficiently solve industry size problems. Thus, six different meta-heuristic algorithms based on tabu search (TS) are developed to efficiently solve for near optimal solutions of the proposed problem. By using randomized complete block design and considering the objective function value of each algorithm as the criterion, the best algorithm among the proposed ones is identified. The experiments are performed by solving the available test problems in the literature. Then, the performance of the best developed meta-heuristic algorithm is compared with the existing algorithm in the literature. A comparison based on paired t-test shows that the average makespan of the proposed algorithm in this research is better than the average makepan of the existing algorithm in the literature. In particular, defining the neighborhoods to allow for the possibility of processing the same group on more than one machine, when multiple machines are available for a stage, contributed to identifying better quality solutions than have been identified in the past. In other words, this research investigates into the possibility of moving away from defining the neighborhoods for groups in a traditional sense where a group is assigned to only one machine in a stage to more than one machine.

Keywords: Flexible Flow Shop Scheduling, Sequence Dependent Group Scheduling, Meta-Heuristics, Tabu Search

1. Introduction

Flexible flow shop sequence dependent group scheduling (FFSDGS) problem has received a great deal of attention during recent years due to its increasing applicability in the real world. It has several advantages compared to the regular flow shop scheduling problems, wherein the bottleneck stages can include more than one machine in parallel in order to eliminate the bottleneck production constraints. Huang and Li [1] presented two heuristics for minimizing the makespan of a flexible flow shop scheduling problem with two stages and sequence independent setups under group technology assumptions. The group technology assumptions in group scheduling require that jobs that belong to a group are processed contiguously on a machine. Andre et al. [2] used the "coefficient of similarity" between the jobs in their development of a heuristic for minimizing the makespan of an FFSDGS problem with three stages. The generalized version of the flexible flow shop group scheduling problem with '*m*' stages and sequence independent setups for minimizing the makespan was first attempted by Logendran et al [3]. They investigated three constructive heuristic algorithms based on LN heuristic (Logendran and Nadtasomboon [4]) and PT heuristic (Petrov [5]). Logendran et al. [6] developed three algorithms based on tabu search (TS) for solving the FFSDGS problem with *m* stages. Three different initial solution finding mechanisms were developed to aid in their search algorithms. For analyzing both makespan and computation time, a detailed statistical experiment based on the split-plot design was performed. To the best of our knowledge this is the only available research for solving the FFSDGS problem.

2. Assumptions and Motivation

In this research, it is assumed that *g* groups including a different number of jobs in each group are assigned to a manufacturing flexible flow shop cell with *m* stages for processing. At least one of the *m* stages has two or more

identical parallel machines, thus satisfying the flexibility requirement in a flexible flow shop. The goal is to find the best sequence of processing the groups as well as the jobs belonging to each group in order to minimize the makespan. The problem can be classified as $FF_m|fmls, S_{sd}|C_{max}$: flexible flow shop with m stages (FF_m); group (family) scheduling problem (*fmls*); sequence dependent setups (S_{sd}); and makespan minimization (C_{max}). Other assumptions made in this research are:

- There is unlimited buffer between stages.
- All machines are available at the beginning of the planning horizon.
- The problem belongs to anticipatory scheduling. In other words, the setup of a machine in a stage to process a group can be started before any job belonging to the group physically arrives at that stage.
- All groups are not required to be processed in the same sequence in all stages. That is, non-permutation scheduling is permissible for groups.
- Group technology assumptions are valid in this research. In other words, the jobs that belong to a group cannot to be split into several sublots for processing on machines.
- All jobs are available for processing on machines at the beginning of the planning horizon.
- The reference group '*R*' is a group which was processed as the last group on a machine in the previous planning horizon. Also the processing of any of the *g* groups on a machine in the current planning horizon should follow the reference group. Thus, for finding the best sequence of groups as well as jobs, the required setup time of each group with respect to reference group should be considered.
- The sequence of jobs belonging to each group can be different in different stages.

The tabu search algorithms used in Logendran et al. [6] were developed based on search algorithmic principles applied in a traditional sense. That is, given a group sequence, the neighborhoods for the group level (outside) search allowed for the possibility of assigning a group to a machine, depending upon the constraints of the problem and the availability of a machine, and that machine-group pair was assumed fixed during that iteration of the search. Clearly, this situation only arises when a stage has more than one machine because for stages with one machine, the machine-group pairs would remain fixed for a given group sequence. The purpose of this research is to exploit the possibility of expanding the search to generate neighborhoods that would allow for the possibility of assigning a group to more than one machine in a stage, thus enhancing the possibility of identifying better quality solutions for the FFSDGS problem. The job level (inside search) is performed using swap moves, similar to the approach used in Logendran et al. [6]. Consequently, this research has led to developing six different types TS algorithms as presented in Section 3.

3. Meta-heuristic Algorithm - Tabu Search

Gupta and Darrow [7] recognize that even the two machine, sequence dependent makespan minimization job (not group)-scheduling problem is NP-hard. Thus the fact the FFSDGS problem with *m* stages is strongly NP-hard follows immediately. Since it is NP-hard, a two level meta-heuristic algorithm based on TS has been developed to heuristically solve the problem. At the first level, which is called the outside search in this research, the assignment of groups to the machines in each stage as well as the sequence of groups assigned to a machine in a stage are determined. In the second level, which is called inside search in this research, the sequence of jobs belonging to a group is determined. The steps associated with the developed TS algorithm are presented next.

3.1 Initial Solution (IS)

In order to begin the search, the algorithm needs an initial feasible solution which is called an initial seed. Logendran et al. [6] proposed three different initial solution finding mechanisms. Although no statistically significant differences were found by employing different IS funding mechanisms in Logendran et al. (2006), the third IS finding mechanism identified better quality initial solutions. For this reason, the third method is retained in this research to explore finding a better IS for a given problem. The method is briefly described next.

Longest processing time (LPT) heuristic is regarded as an effective algorithm for minimizing the makespan with known worst-case-bound (Pinedo [8]). The method exploits the use of LPT in the IS found at both levels. Based on Logendran et al. [6], a *key stage* must be determined for obtaining an IS. A stage is considered the key stage if it has the longest cumulative processing time (LCPT) for all groups in that stage. The LCPT for each stage is calculated as the sum of the minimum setup time for each group in that stage plus the run times for jobs in all groups. For this purpose, the minimum sequence dependent setup time required to changeover from the remaining (*g*-1) groups and the reference group to each of the *g* groups must be determined on each of the *m* stages of the flexible flow shop. Once LCPT for each stage is found, the key stage is obtained as that which has the longest LCPT. If there is a tie for LCPT, then the stage that has smallest stage number is selected. After determining the key stage, the job sequence for each group is identified based upon the longest run time (LRT)

on the key stage. Ties are broken in favor of the smallest job number. The group sequence is identified based upon LCPT on the key stage. The LCPT for each group is determined as the sum of the run time for jobs within that group and the minimum sequence dependent setup time for that group on the key stage. If there is a tie for LCPT in the key stage, then the group that has the smallest group number is selected.

3.2 Neighborhood Finding Mechanisms

As non-permutation scheduling is permissible in this research, the IS determined above is used to represent the group sequence and the job sequence in each group in the *first stage*. The group and job sequences in the remaining stages are determined based upon the First-In-First-Out (FIFO) rule, non-permutation scheduling, Method 2 (described below) for assigning groups to machines, and objective function value (makespan). The group and job sequences determined on all machines in all stages are regarded as the initial seed. With the initial seed in hand, the search is continued to find the next seed by evaluating the quality of the neighborhood solutions of the current seed. For the outside search, the neighborhood solutions for the group sequence of each seed are obtained by applying numerical moves along with insert moves at each stage. A numerical move is performed by changing the position of two groups in a stage by exchanging a group's position with that of another group's position that is *larger*, while maintaining the same positions for the remaining groups. An insert move is performed by removing a group scheduled to be processed on one machine and inserting it in any available position, either on the same machine or another machine, in the same stage. For instance, consider a problem with seven groups (G_1, G_2, G_3, G_4, G_5, G_6, and G_7), and two stages. Assume that in the first stage there is one machine (M_{11}), and in the second there are two machines (M_{21}, M_{22}). Focusing on stage 2, assume also that in the current seed the first three groups (G_1, G_2, and G_3) are assigned to M_{21} and are scheduled to be processed in the numerical order sequence: G_1 - G_2 - G_3. The last four groups are assigned to M_{22} and are scheduled to be processed in the numerical order sequence: G_4 - G_5 - G_6 - G_7.

In order to find the neighborhoods of stage 2, the position of the first group (G_1) is exchanged with the position of the other groups in stage 2. Also, G_1 is removed from its current position and is scheduled to be processed by inserting it in any available position on M_{21} and M_{22}. In other words, by inserting G_1 after G_2, and after G_3, both on M_{21}, and by inserting G_1 before G_4, after G_4, before G_5 and so on, all on M_{22}, the neighborhoods can be derived. As it can be seen, the neighborhood in which the positions of G_1 and G_2 are exchanged on M_{21} is generated by both the numerical move and insert move. To eliminate this redundancy, we ignore the one generated by insert move. In summary, by performing numerical moves and insert moves for G_1 in this example, 12 neighborhoods (six neighborhoods by exchanging the position of G_1 with other groups' position, and six neighborhoods by inserting G_1 in any available position among the groups on both machines) can be derived. The neighborhood of inside search (sequence of jobs within a group) is obtained by performing swap moves of jobs on the current machine, since all jobs of a group are supposed to be processed on the same machine. The swap move is obtained by exchanging two adjacent jobs of a group, including the cyclical exchange. The cyclical exchange is obtained by exchanging the first and the last job of each group in a seed, while maintaining the same positions for the remaining jobs. If there is more than one machine in the next stage, as is the case here, two different methods can be used to determine which machine a group should be assigned to. The first method (Method 1) focuses on assigning the group to the earliest available machine. In Method 2, used in this research, the group is assigned to a machine that has the minimum sum of setup time of the group based on the last group processed on that machine and its completion time, also on the same machine. Logendran et al. (2006) used Method 1.

Two different constructs are used to establish the neighborhoods of a seed for outside search. The first (Construct 1) is based upon the sequence of groups scheduled to be processed *on each machine in each stage* of a seed. In the above example, it is represented by G_1 - G_2 - G_3 on M_{21} and G_4 - G_5 - G_6 - G_7 on M_{22}, for stage 2. Construct 2 uses the group sequence on each machine of a stage having more than one machine to first identify a seed (sequence of groups) representative of that stage. In other words, to which machine of a stage a group is assigned to is disregarded in the second construct. The completion time of the first job belonging to each group in that stage is used to develop the group sequence. Suppose that the rank-ordered completion times in the above example are J_{11} - J_{21} - J_{41} - J_{31} - J_{51} - J_{61} - J_{71} with the smallest being J_{11} (the first job of G_1). The seed for stage 2 with Construct 2 is thus represented by G_1 - G_2 - G_4 - G_3 - G_5 - G_6 - G_7, while that with construct 1 is represented by G_1 - G_2 - G_3 on M_{21} and G_4 - G_5 - G_6 - G_7 on M_{22}. The neighborhoods of these seeds are obtained by performing numerical moves and insert moves. Observe that both constructs will result in the same seed for a stage with one machine.

3.3 Algorithmic Structure

The IS is considered as a seed for both outside and inside search. The inside search is performed to find the best job sequence based on the current group sequence. The first entry into the inside candidate list (ICL) is the job

sequence in the IS. The inside search is initiated by completely evaluating the neighborhoods of the job sequence in IS based on the objective function value. It is applied for the current group sequence in IS to determine the best or near optimal job sequence. These neighborhoods generated by swap moves must be checked against the inside tabu list and the neighborhood which has its move included in the tabu list, must be excluded. But this filtering for a neighborhood can be ignored if its value is better than the inside aspiration level. The inside aspiration level is the best value found among all the neighborhoods of the current inside search. Also, the neighborhoods that are included in the candidate list must be excluded from further consideration. Then, among other neighborhoods, the neighborhood that has the best value (smallest makespan) is chosen as the next seed and added to the candidate list as its next entry. If more than one neighborhood has the same value, then the neighborhood is chosen in favor of the first best solution. The move that leads to the next seed is regarded as tabu and is inserted into the inside tabu list (ITL). The tabu list should be updated. So if ITL is filled, the move which was forbidden for the longest duration (the oldest move in the ITL) should be replaced with the most recent move in the ITL. If the makespan for a seed was either equal or less than the value of entries immediately before and after in the ICL, then this seed can be inserted into the inside index list (IIL). IIL is a subset of ICL, which includes the local optima. For stopping the inside search, two criteria are used: the maximum number of entries into the inside index list (MIILS) and the maximum number of inside iterations without improvement (MIIWI). When a feasible solution is added to the ICL, if its value is not less than the value of the previous member of the ICL, the value of inside iterations without improvement is increased by one; otherwise this counter is reset to zero. Several test problems were generated to empirically determine appropriate values for inside search parameters for the range of problems investigated in this research. These test problems were then solved by meta-heuristic algorithms by applying different values for each parameter to find the best value for each parameter. In the interest of space, such parameter values identified are not reported here.

When the inside search is completed based on MIILS or MIIWI, the best sequence of jobs determined is considered as the best solution for the current group sequence and the search is switched to the outside search. The algorithmic steps of the outside search are similar to those of the inside search. Thus, the outside search starts by determining the neighborhoods of the group sequence in IS with numerical moves along with insert moves. The first entry into the outside candidate list (OCL) is the group sequence in the IS. Then, for each of the new group sequences identified in the outside neighborhood, the inside search is performed to find its best job sequence. As with the inside search, these neighborhoods must be checked against the outside tabu list (OTL) and OCL by comparing their values to the outside aspiration level. Considering the available neighborhoods and by using the first best solution strategy, the best among them (smallest makespan) is chosen as the next seed and added to the OCL as the second entry. The move that leads to the next seed is regarded as tabu and is inserted into the OTL. The OTL, outside index list (OIL), outside aspiration level, and number of outside iterations without improvement should be updated. Also, for stopping the outside search, both the maximum number of entries into the outside index list (MOILS) and the maximum number of outside iterations without improvement (MOIWI) are used. As for the inside search, several test problems were solved for identifying appropriate values for outside search parameters. Such parameter values are not reported here.

Preliminary experiments were conducted on test problems with long-term memory based on maximum frequency (LTM-MAX) and on minimum frequency (LTM-MIN). They showed a better performance for LTM-MAX, so it is retained and LTM-MIN was disregarded. With LTM-MAX, the search is restarted to explore more by intensifying the search. In this problem, the outside search plays an important role in obtaining the best solution than inside search. Thus, the proposed algorithm uses LTM-MAX only for outside search. A four dimensional matrix is used to gather the required information for LTM in outside search by using the OCL of solutions throughout the search. The concept of “slots” is introduced to facilitate establishing the frequency matrix. It is assumed that there are g slots to process the g groups on each machine of each stage. Each group at each stage must be assigned to only any one of these slots. Likewise, each slot also can be assigned to only any one of the groups or is not assigned to any of the groups. Thus, the first dimension of the frequency matrix is assigned to each stage. The second one is assigned for each available machine-slot. The third one is for the machine number and the fourth one is for the group number. The value of each member of this matrix reveals the number of times that a group is assigned to a particular available machine-slot. Based on this information, the long term search is performed by fixing the group in the machine-slot with the maximum frequency and the search is restarted. In this algorithm, the number of restarts with LTM-MAX for the outside search is set equal to one. During the restart process of the outside search, the OTL, OCL, OIL, and the number of outside iterations without improvement are reset to zero. When the outside search is completed based on MOILS or MOIWI with or without LTM, the entire search is terminated. For a given problem, the best solution determined during solving the problem by the meta heuristic algorithms is deemed as the final best solution.

3.4 Characteristics of the Proposed TS Algorithms

The outside neighborhoods and the inside neighborhoods are considered in all of the stages. For each outside neighborhood, the group sequences on machines, from the first stage to that preceding the stage used for developing the outside neighborhood, will remain unchanged. But the group sequences on machines in each stage that follow the stage used for developing the outside neighborhood can change from what they were before. Accordingly, during the inside search, the revised job sequences are determined *only* for the group sequences in the affected stages. Such curtailment in inside search can result in savings in computation time. With characteristics of the outside search serving as the defining factor, six different TS algorithms are developed in this research. The first algorithm (TS1) uses Construct 1 to develop the outside neighborhoods and the inside neighborhoods based on numerical moves and swap moves, respectively. The second algorithm (TS2) is the same as TS1, but the outside neighborhoods are based on insert moves. The third algorithm (TS3) is the same as TS1, but the outside neighborhoods are based on a combination of numerical moves and insert moves. The fourth algorithm (TS4) is the same as TS1, but uses Construct 2. The fifth algorithm (TS5) is the same as TS4, but the outside neighborhoods are based on insert moves. The sixth algorithm (TS6) is the same as TS4, but the outside neighborhoods are based on a combination of numerical moves and insert moves.

4. Performance of TS algorithms

Fifteen different test problems, which were used by Logendran at al. [6] defined in a range shown in Table 1, are solved to compare these algorithms and the makespan is recorded. All of the TS algorithms are programmed in Microsoft C and implemented on a server with 3.07 GHz, 1 GB RAM memory, and windows XP operating system. A randomized complete block design is used with the algorithms regarded as treatments and the 15 test problems regarded as blocks (Montgomery [9]). The Minitab software is used to establish the normal probability plot of residuals for makespan. The plot showed prefect normality, further supporting the use of analysis of variance based techniques with a randomized complete block design. At a 5% significance level, TS3 is found to be the best algorithm, outperforming the other algorithms.

Table 1. Performance of TS Algorithms

Problem number	Problem size			Objective function value (makespan)					
	Number of stages	Number of groups	Number of jobs in all groups	TS1	TS2	TS3	TS4	TS5	TS6
1	4	8	62	2155	2155	2149	2155	2180	2158
2	4	8	62	2136	2144	2136	2136	2154	2136
3	4	8	62	1139	1148	1153	1148	1148	1139
4	6	8	47	1957	1958	1949	1957	1958	1949
5	6	8	47	1945	1945	1935	1945	1987	1941
6	6	8	47	1047	1047	1054	1047	1047	1047
7	6	9	65	2476	2478	2418	2476	2476	2478
8	6	9	65	2431	2433	2414	2431	2468	2448
9	6	9	65	1310	1342	1300	1343	1387	1342
10	5	6	46	1719	1719	1701	1737	1737	1719
11	5	6	46	1466	1466	1455	1466	1466	1466
12	5	6	46	961	973	942	970	973	961
13	4	7	41	1889	1889	1873	1889	1896	1889
14	4	7	41	1749	1749	1749	1749	1749	1749
15	4	7	41	1001	1003	1001	1003	1009	1003

5. Performance of TS3 compared to Logendran et al. [6]

The performance of TS3 is further investigated by comparing its solution with the best solution obtained by Logendran et al [6] on available test problems. Logendran et al. [6] chose the following ranges for small, medium, and large problems: number of stages is varied from 2 to 3, 4 to 6, and 7 to 9 for small, medium and large problem instances, respectively; number of groups is varied from 3 to 5, 6 to 9, and 10 to 12 for small, medium and large problem instances, respectively; and that for the number of jobs is varied from 2 to 5, 6 to 9 and 10 to 12 for small, medium and large problem instances, respectively. In order to perform an objective comparison, the same test problems as those in Logendran et al. [6] are used only for *medium* size problems in the interest of time and space. The results are presented in Table 2.

The difference in the last column of Table 2 is evaluated as: makespan from TS3 – makepsan from Logendran et al.'s algorithm. At 5% significance level, a one-way hypothesis test is used to investigate whether or not the average makespan evaluated by TS3 is smaller than that by Logendran et al. [6]. With a P-value of 0.00056903, the results show that TS3 outperforms the algorithm by Logendran et al [6], in being able to identify a better makespan for the medium size problems solved.

Table 2. Comparative Performance of TS3

Problem number	Problem size			The makespan from Logendran et al.'s algorithm	The makeapsn from TS3	Difference
	Number of stages	Number of groups	Number of jobs in all groups			
1	4	8	62	2180	2149	-31
2	4	8	62	2138	2136	-2
3	4	8	62	1158	1153	-5
4	6	8	47	1949	1949	0
5	6	8	47	1941	1935	-6
6	6	8	47	1070	1054	-16
7	6	9	65	2453	2418	-35
8	6	9	65	2464	2414	-50
9	6	9	65	1343	1300	-43
10	5	6	46	1707	1701	-6
11	5	6	46	1466	1455	-11
12	5	6	46	970	942	-28
13	4	7	41	1889	1873	-16
14	4	7	41	1749	1749	0
15	4	7	41	1009	1001	-8

6. Conclusions

In this paper a more effective and yet efficient tabu search algorithm has been developed for solving the FFSDGS problem with makespan minimization as the criterion. Specifically, two different constructs have been used in the outside search to consider the outside and inside neighborhoods in all of the stages, by employing Method 2 to assign a group to a machine in a stage having more than one machine. Investigating into the possibility of assigning a group to more than one machine in a stage has enabled defining the neighborhoods for outside search quite so differently than previous research, thus significantly contributing to the improvement in quality of the solutions. While experimentation is needed with small and large problem instances to claim superiority of solutions across all problem sizes, the comparison performed on medium size problems shows that the quality of the solutions produced by TS3 outperforms that by Logendran et al. [6].

References

1. Huang, W. and Li, S. (1998), A two-stage hybrid flowshop with uniform machines and setup times, Mathematical and Computer Modeling, 27, pp. 27-45.
2. Andre´s, C., Albarracı´n, J.M., Tormo, G., Vicens, E. and Garcı´a- Sabater, J.P. (2005), Group technology in a hybrid flowshop environment: A case study, European Journal of Operational Research 167, pp. 272–281.
3. Logendran, R., Carson, S. and Hanson, E. (2005), Group scheduling in flexible flow shops, International Journal of Production Economics, 96(2), pp. 143-155.
4. Logendran, R. and Nudtasomboon, N. (1991), Minimizing the makespan of a group scheduling problem: A new heuristic, International Journal of Production Economics, 22, pp. 217-230.
5. Petrov, V.A. (1966), "Flow Line Group Production Planning", Business Publications, London.
6. Logendran, R., deSzoeke, P. and Barnard, F. (2006), Sequence-dependent group scheduling problems in flexible flow shops, International Journal of Production Economics, 102, pp. 66-86.
7. Gupta, J.N.D. and Darrow, W.P. (1986), The two-machine sequence dependent flowshop scheduling problem," European Journal of Operational Research, 24, pp. 439-446.
8. Pinedo, M. (1995), "Scheduling Theory, Algorithms, and Systems", Prentice Hall, Englewood Cliffs, New Jersey.
9. Montgomery, D.C. (2009), "Design and Analysis of Experiments", Wiley, New York.