Journal of Intelligent Manufacturing, 15, 777–785, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems

ADİL BAYKASOĞLU[1], LALE ÖZBAKIR[2] and ALİ İHSAN SÖNMEZ[3]

[1]*University of Gaziantep, Department of Industrial Engineering, 27310, Gaziantep, Turkey*
[2]*Erciyes University, Department of Industrial Engineering, Kayseri, Turkey*
[3]*CAG University, Department of Management, Tarsus—Mersin, Turkey*

Received October 2002 and accepted December 2003

A linguistic-based meta-heuristic modeling and solution approach for solving the flexible job shop scheduling problem (FJSSP) is presented in this study. FJSSP is an extension of the classical job-shop scheduling problem. The problem definition is to assign each operation to a machine out of a set of capable machines (the routing problem) and to order the operations on the machines (the sequencing problem), such that predefined performance measures are optimized. In this research, the scope of the problem is widened by taking into account the alternative process plans for each part (process plan selection problem). Probabilistic selection of alternative process plans and machines are also considered. The FJSSP is presented as a grammar and the productions in the grammar are defined as controls (Baykasoğlu, 2002). Using these controls and Giffler and Thompson's (1960) priority rule-based heuristic along with the multiple objective tabu search algorithm of Baykasoğlu *et al*. (1999) FJSSP is solved. This novel approach simplifies the modeling process of the FJSSP and enables usage of existing job shop scheduling algorithms for its fast solution. Instead of scheduling job shops with inflexible algorithms that cannot take into account the flexibility which is available in the job shop, the present algorithm is developed which can take into account the flexibility during scheduling. Such an approach will considerably increase the responsiveness of the job shops.

Keywords: Scheduling, grammars, optimization, flexibility, tabu search

1. Introduction

Job shop scheduling problem (JSSP) is a branch of production scheduling, which is among the hardest combinatorial optimization problems (Sönmez and Baykasoğlu, 1998). Not only it is hard, but even among the members of the latter class, it appears to belong to the more difficult ones (Van Laarhoven *et al*., 1992).

Many different approaches have been applied to JSSP, including: dispatching rules (Giffler and Thompson, 1960; Blackstone *et al*., 1982), mathematical programming (Sönmez and Baykasoğlu, 1998; Pinedo, 1995), artificial intelligence techniques (Kusiak and Chen, 1988), simulation methods (Baykasoğlu *et al*., 1998), heuristics (Kusiak, 1990) etc. Ben-Daya (1994) provides a good classification of the techniques applied to job shop scheduling. It has been recognized that schedule optimization using mathematical programming is very difficult, because of prohibitive computational time. It becomes more difficult to achieve an optimal result when the variety of parameters and constraints is increased (Maturana *et al*., 1997). Due to this fact, artificial intelligence based meta-heuristics (simulated annealing, genetic algorithms, tabu search etc.) and dispatching rule-based heuristics have gained significant research attention for solving scheduling problems. Many

researchers from all around the world applied meta-heuristics to scheduling problems. Maturana *et al*. (1997) employed genetic algorithms for solving JSSP. Cheng *et al*. (1996) gave a good review of genetic algorithms for job shop scheduling. Al-Fawzan and Al-Sultan (1996) applied tabu search to JSSP. Van Laarhoven *et al*. (1992) used simulated annealing for JSSP. Dorndorf and Pesch (1995) developed a dispatching rule-based genetic algorithm for JSSP. Many other example applications can easily be found in the literature.

Although an optimal solution algorithm for the classical JSSP has not been developed yet, there is a trend in the research community to model and solve a much more complex version of the problem. The problem is referred to as flexible job shop scheduling problem (FJSSP). The need to model and solve FJSSP is mainly emerged due to the fact that modern machine tools have considerable amount of overlapping capabilities. Modeling the scheduling problems by classical approach without taking into consideration the overlapping capabilities of machines does not reflect the reality of modern job shops. Moreover, predetermining machine routes without considering alternative routes that are resulting from overlapping machine capabilities can reduce the job shop performance considerably (Baykasoğlu, 2002).

FJSSP is an extension of the classical JSSP and incorporates all of difficulties and complexities of its predecessor JSSP. The classical problem definition for FJSSP is to assign each operation to a machine out of a set of capable machines (the routing problem), and to order the operations on the machines (the sequencing problem), such that a predefined performance measure is optimized. The FJSSP mainly presents two difficulties. The first one is the assignment of each operation to a machine, and the second one is to schedule of this set of operations in order to minimize a predefined criterion (Mesghouni *et al*., 1998).

The literature of FJSSP is considerably sparser than the literature of JSSP. Mesghouni *et al*. (1998) proposed a genetic algorithm based model for FJSSP. They explained implementation details of their genetic algorithm in detail by solving an example problem. Chambers (1996) developed a tabu search algorithm for FJSSP. Mastrolilli and Gambardella (2000) proposed some neighborhood functions for the FJSSP, which can be used in meta-heuristic optimization techniques. Jansen *et al*. (2000) developed approximation algorithms for FJSSP. It seems that previous research on FJSSP favors meta-heuristics. This is logical and understandable, since the problem independent nature of meta-heuristics provides a great opportunity to model complex optimization problems (Jansen *et al*., 2000). The most important issue in employing meta-heuristics for combinatorial optimization problems is to develop an effective ''problem mapping'' and ''solution generation'' mechanism. If these two mechanisms are devised successfully, then it is possible to find good solutions to a given optimization problem in an acceptable time.

In this research paper, a practical modeling and solution approach is proposed for modeling and solving FJSSP. The proposed approach makes use of grammars, multiple objective tabu-search (MOTS) and a dispatching rule-based heuristics. The present work is an extension of the work done by Baykasoğlu (2002). Baykasoğlu (2002) is the first one in the literature that presented how grammars can be used effectively to model and solve flexible job shop scheduling problems. In Baykasoğlu (2002) the problem is modeled as a single objective model and machine capabilities are defined and formulated using resource elements. Due to difficulties in defining machine capabilities in terms of resource elements and the need to consider multiple objectives in scheduling, the existing model is extended to take into account multiple objectives and operations instead of resource elements.

Employing grammars in FJSSP simplifies the problem modeling. Using the grammars also helps to extent the scope of the FJSSP by considering alternative process plans. In the present study, alternative process plans for each part (process plan selection problem) is also considered during scheduling. As an extension to the previous studies, probabilistic selection of alternative process plans and machines are also considered.

In this work, the FJSSP is presented as a grammar and the productions in the grammar are defined as controls (Baykasoğlu, 2002). Using these controls and Giffler and Thompson (1960) priority rule-based heuristic the multiple objective tabu search algorithm (Baykasoğlu *et al*., 1999) is employed to solve the FJSSP. This novel approach simplifies the modeling process of the FJSSP and enables usage of existing job shop scheduling algorithms for its solution. Instead of scheduling job shops with inflexible algorithms that cannot take into account flexibility available in the job

shop, employing novel algorithms and taking into account available flexibility during scheduling will considerably increase the responsiveness of the job shops.

2. Grammatical representation of FJSSP

By analogy to formal languages, the data for FJSSP (part process plans, processing requirements and machine tool capabilities etc.) can be represented by using context-free grammars (Fu, 1974). A grammar for defining the syntax of the data to be used in FJSSP is a four-tuple (Baykasoğlu, 2002):

$$G_S = \{\Sigma_S, \Pi_S, P_S, S\} \quad (1)$$

where:

Σ_S: A set of terminal symbols, representing the alternative machines for machining operations.

Π_S: A set of non-terminal symbols, representing parts, their process plans and processing requirements.

P_S: A finite set of production rules for determining the legal process plans and machines to be selected.

S: A non-terminal start symbol.

Productions (P_S) are of the form:

$$\Phi \rightarrow \Delta, \Omega_i \overset{\pi_{ij}}{\rightarrow} \Psi_{ij}, \Theta_i \overset{\mu_{ij}^k}{\rightarrow} \Lambda_{ij} \quad (2)$$

where Φ, Ω_i, Θ_i are non-terminals. Δ, Ψ_{ij}, are sets of non-terminals, Λ_{ij} are terminals. π_{ij} and μ_{ij}^k are controls. Controls are used to determine the production to be selected. Each production has a probability for selection that is determined by using simple preference equations.

As an example, consider the grammar of a family of three parts with their corresponding process plans defined in terms of four operations to be machined on five machines. The data for parts and machines is given in Tables 1–3.

The grammar for this production situation is defined as follows:

$\Sigma_S = \{M1,\ M2,\ M3,\ M4,\ M5\}$

$\Pi_S = \{P1,\ P2,\ P3,\ op1,\ op2,\ op3\}$

$S =$ Select a process plan for each part, and machines for each operation *(Select_Machines)*

Table 1. Part data

Parts	*Process plan number*	*Process plans and processing sequence*
*P*1	1	*op*1–*op*2–*op*3
	2	*op*1–*op*3–*op*2
*P*2	1	*op*2–*op*3–*op*1
	2	*op*3–*op*2–*op*1
*P*3	1	*op*1–*op*3
	2	*op*1–*op2*

Table 2. Machine-operation suitability data

	*op*1	*op*2	*op*3
*M*1	*		
*M*2	*		*
*M*3		*	
*M*4			*
*M*5		*	

Table 3. Processing time data

O / M	*op*1			*op*2			*op*3		
	*P*1	*P*2	*P*3	*P*1	*P*2	*P*3	*P*1	*P*2	*P*3
*M*1	2	4	1						
*M*2	4	3	2				1	3	2
*M*3				3	2	4			
*M*4							1	4	3
*M*5				4	1	5			

The set of productions P_S is given by:

Select_Machines $\rightarrow P1, P2, P3$

$$P1 \overset{\pi_{11}}{\rightarrow} op1_{P1}, op2_{P1}, op3_{P1}$$

$$P1 \overset{\pi_{12}}{\rightarrow} op1_{P1}, op3_{P1}, op2_{P1}$$

$$P2 \overset{\pi_{21}}{\rightarrow} op2_{P2}, op3_{P2}, op1_{P2}$$

$$P2 \overset{\pi_{22}}{\rightarrow} op3_{P2}, op2_{P2}, op1_{P2}$$

$$P3 \overset{\pi_{31}}{\rightarrow} op1_{P3}, op3_{P3}$$

$$P3 \overset{\pi_{32}}{\rightarrow} op1_{P3}, op2_{P3}$$

$$op1_{P1} \overset{\mu^1_{11}}{\rightarrow} M1(2)$$
$$op1_{P1} \overset{\mu^1_{12}}{\rightarrow} M2(4)$$
$$op1_{P2} \overset{\mu^2_{11}}{\rightarrow} M1(4)$$
$$op1_{P2} \overset{\mu^2_{12}}{\rightarrow} M2(3)$$
$$op1_{P3} \overset{\mu^3_{11}}{\rightarrow} M1(1)$$
$$op1_{P3} \overset{\mu^3_{12}}{\rightarrow} M2(2)$$
$$op2_{P1} \overset{\mu^1_{21}}{\rightarrow} M3(3)$$
$$op2_{P1} \overset{\mu^1_{22}}{\rightarrow} M5(4)$$
$$op2_{P2} \overset{\mu^2_{21}}{\rightarrow} M3(2)$$
$$op2_{P2} \overset{\mu^2_{22}}{\rightarrow} M5(1)$$
$$op2_{P3} \overset{\mu^3_{21}}{\rightarrow} M3(4)$$
$$op2_{P3} \overset{\mu^3_{22}}{\rightarrow} M5(5)$$
$$op3_{P1} \overset{\mu^1_{31}}{\rightarrow} M2(1)$$
$$op3_{P1} \overset{\mu^1_{32}}{\rightarrow} M4(1)$$
$$op3_{P2} \overset{\mu^2_{31}}{\rightarrow} M2(3)$$
$$op3_{P2} \overset{\mu^2_{32}}{\rightarrow} M4(4)$$
$$op3_{P3} \overset{\mu^3_{31}}{\rightarrow} M2(2)$$
$$op3_{P3} \overset{\mu^3_{32}}{\rightarrow} M4(3)$$

If the following set of controls are selected randomly for each part and their corresponding operations:

$$\{[(\pi_{11}), \mu^1_{11}, \mu^1_{22}, \mu^1_{32}], [(\pi_{21}), \mu^2_{22}, \mu^2_{31}, \mu^2_{11}], [(\pi_{31}), \mu^3_{12}, \mu^3_{32}]\}$$

Then, the resultant strings will be:

Part1—*M*1(2), *M*5(4), *M*4(1)
Part2—*M*5(1), *M*2(3), *M*1(4)
Part3—*M*2(2), *M*4(3)

This is simply a transformation of the FJSSP to a JSSP. Selecting different set of controls will produce different JSSP instants of the FJSSP. The resultant string can also be represented using a standard notation of JSSP as shown in Table 4.

The problem here is to find an optimal combination of controls. While determining the set of controls, a feasible schedule for the corresponding JSSP instant should also be determined simultaneously. Dispatching rule-based heuristics (Cheng *et al.*, 1997) can be used for searching such feasible schedules. This problem can easily be modeled using meta-heuristics (simulated annealing, tabu search, genetic algorithms etc.).

Table 4. Representation of the resultant string with a standard JSSP notation

	Operations		
	1	*2*	*3*
	Processing time		
*P*1	2	4	1
*P*2	4	1	3
*P*3	2		3
	Machine sequence		
*P*1	*M*1	*M*5	*M*4
*P*2	*M*1	*M*5	*M*2
*P*3	*M*2		*M*4

In the above model each process plan and machine has equal probability for selection. However, for some operations, some machines might be more favorable than the others. Moreover, for some parts some process plans might also be more favorable. In such cases, probabilistic selection is introduced to the controls of the grammar (i.e. stochastic grammars (Upton and Barash, 1988). Then, the productions for the process plan and machine selection is represented as:

$$P1 \overset{\pi_{11}=0.55}{\rightarrow} op1_{P1}, op2_{P1}, op3_{P1}$$
$$P1 \overset{\pi_{12}=0.45}{\rightarrow} op1_{P1}, op3_{P1}, op2_{P1}$$
$$\cdots\cdots\cdots$$
$$op1_{P1} \overset{\mu^1_{11}=0.67}{\rightarrow} M1(2)$$
$$op1_{P1} \overset{\mu^1_{12}=0.33}{\rightarrow} M2(4)$$
$$\cdots\cdots\cdots\cdots\cdots$$

In this example, probability of selection of Machine1 for operation1 of part1 is 0.67; on the other hand the probability of selecting Machine2 is 0.33. Notice that there are two alternatives for operation1 and sum of the probabilities is equal to 1. In this study

probabilities are computed by considering processing times. Operation1 of part1 has two machine alternatives $M1$ and $M2$ respectively. Operation1 takes 2 min on $M1$ and 4 min on $M2$. Therefore, probability to select $M1$ is equal to $1-(2/(2+4))$, and probability to select $M2$ is equal to $1-(4/(2+4))$. All other machine selection probabilities are computed in a similar manner. Determining process plan selection probabilities is not straightforward and requires some expert knowledge. Therefore its determination is left to the decision maker. But, sequence dependent setup times can be used in setting up these probabilities.

In the following section, the dispatching rule-based heuristic of Giffler and Thompson (1960) is summarized. The explanations given in the following section are adopted from (Cheng *et al*., 1997). Giffler and Thompson's heuristic is used in the present work to generate feasible schedules for the JSSP instances of FJSSP.

3. Giffler and Thompson algorithm for generating JSSP instances of FJSSP

Dispatching rules are probably the most frequently used heuristics for solving scheduling in practice because of their ease of implementation and their less time complexity (Cheng *et al*., 1997). Blackstone *et al*. (1982) gave an extensive summary and discussion of priority dispatching rules.

Consider a problem with 3 parts and these parts have in total 11 operations to be processed (i.e., $\#\,\text{entry} = \sum_{i=1}^{\#\,\text{parts}} \#\,\text{operations}_i$). In this case we will have a string with 11 numbers of entry $(p_1, p_2, p_3, \ldots, p_{11})$. An entry p_i shows one rule of the set of previously specified dispatching rules. The entry in the ith position says that a conflict in the ith iteration of the Giffler and Thompson algorithm should be resolved using the dispatching rule p_i. In other words, an operation from the conflict set has to be selected by rule p_i; ties are broken by a random choice.

The procedure for deduction a feasible schedule (for the JSSP instance of FJSSP) from a given set $(p_1, p_2, p_3, \ldots, p_{11})$ works as follows:

Notation:

PS_t = a partial schedule containing t scheduled operations.

S_t = the set of schedulable operations at iteration t, corresponding to a given PS_t.

σ_i = the earliest time at which operation $i \in S_t$ could be started.

ϕ_i = the earliest time at which operation $i \in S_t$ could be completed.

C_t = the set of conflicting operations in iteration t.

Procedure: Deduce a string for dispatching rule based encoding

Step 1: Let $t=1$ and begin with PS_t as the null partial schedule and S_t include all operations with no predecessors.

Step 2: Determine $\phi_t^* = \min_{i \in S_t}\{\phi_i\}$ and the machine m^* on which ϕ_i^* could be realized. If more than one such machine exists, tie is broken by a random choice.

Step 3: Form conflicting set C_t which includes all operations $i \in S_t$ with $\sigma_i \leq \phi_i^*$ that requires machine m^*. Select one operation from C_t by dispatching rule p_t and add this operation PS_t as early as possible, thus creating new partial schedule PS_{t+1}. If more than one operation exists according to the dispatching rule p_t, tie is broken by a random choice.

Step 4: Update PS_{t+1} by removing the selected operation from S_t and adding the direct successor of the operation to S_t. Increment t by one.

Step 5: Return to step 2 until a complete schedule is generated.

The implementation of the MOTS algorithm for the FJSSP is presented in the following section.

4. Implementing MOTS for solving FJSSP

Tabu search is a heuristic neighborhood search algorithm that is first suggested and applied by Glover (1990). The basic tabu search algorithm operates in the following way. It starts from a randomly selected or a known feasible solution S. From this initial solution, a set of neighborhood solutions, S^*, are generated using a number of previously determined movement strategies. The objective function is evaluated for each solution in S^*, and the best neighbor replaces the current solution, even though it may be worse than S: in this way it is possible to escape from the local minima (or maxima) of the objective function. The algorithm iterates, repeating the procedure with the new solution, until

some given stopping condition(s) is reached. However the algorithm, as described above, may recycle. To avoid this situation, certain attributes of the last κ replaced solutions are stored in a list, which is called the tabu list. The neighbors of the current solution that satisfy conditions given by the tabu list are systematically eliminated unless they meet an aspiration criterion, so at each iteration the algorithm is forced to select a point not recently selected. The main stages of a tabu search algorithm are; initial solution, generation of neighbors, selection, aspiration, and updating.

The idea of applying tabu search to multiple objective optimization, comes from its solution structure, in which it works with more than one solution (neighborhood solutions) at a time (Baykasoğlu *et al*., 1999). This situation gives us the opportunity to evaluate multiple objectives simultaneously. To enable the original tabu search algorithm to work with more than one goal (objective) the selection and updating stages are redefined. Other stages are identical to the original tabu search algorithm. Detailed explanations for solving multiple objective models using tabu search can be obtained from Baykasoglu *et al*. (1999). In this paper only the solution representation and neighborhood generation functions will be explained. The other functions are similar to original MOTS.

4.1. *Solution representation*

A solution is represented as a string that has two main parts. The first part is composed of set of control parameters of the grammar, and the second part is composed of set of dispatching rules. Figure 1 presents a sample configuration.

As an example consider the following string for the problem given in Section 2:

1 1 2 2 **1** 2 1 1 **2** 2 2 1 2 3 2 1 1 2 2

In this string the first entry means that process plan 1 (π_{11}) is randomly selected between the two alternatives for part1. The second entry means that machine1 (μ_{11}^1) is randomly selected for the operation of part1 among the two alternative machines etc. The 12th entry means that the dispatching rule1 (SPT) is randomly selected among the 3 alternative dispatching rules for the first position of dispatching rule set (12th in the whole string) etc. In the solution string, process plan selection controls (π_{ij}) are shaded and bolded, machine selection controls (μ_{ij}^k) are given in between process plan selection controls for the corresponding parts, dispatching rule set for the operations are underlined.

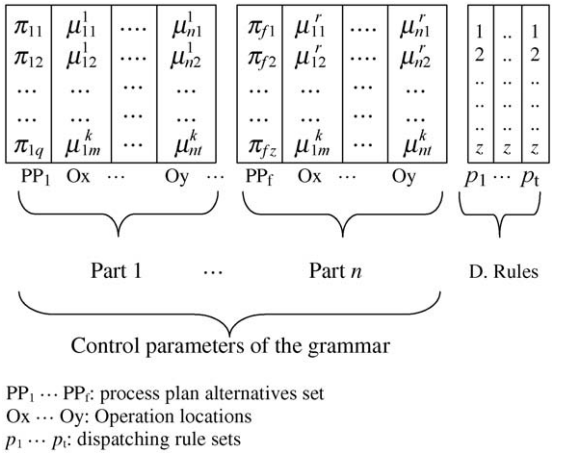

Fig. 1. Representation of the solution string in MOTS implementation of FJSSP

4.2. *Neighborhood generation*

A transition from one solution to another result in a neighbor solution. In this work, neighbor solutions are generated as follows:

> Select n candidate parts whose process plans will be reselected according to the predetermined probabilities. For each reselected process plan reselect machines for m number of operations according to the predetermined probabilities. Change k number of dispatching rules from the set of dispatching rules.

This simple movement strategy always generates feasible solutions and it is easy to implement. Once a new solution is generated the computer program maps the string to the corresponding JSSP and by using the Giffler and Thompson's heuristic the corresponding schedule and its performance is obtained.

An example of neighborhood move for the problem given in Section 2 is shown below. In the example, process plans of the first and second parts will be reselected. After selecting the process plans, the fist operation of the first part and the first and second operations of the second part will be reassigned to the alternative machines. Two dispatching rules in 12th and 13th positions will also be reselected.

(i) Before the neighborhood move

Solution string:
1 1 2 2 **1** 2 1 1 **2** 2 2 1 2 3 2 1 1 2 2
Makespan: **16**, Mean Tardiness: **70**

(ii) After the neighborhood move

Solution string:
2 2 2 2 **2** 1 2 1 **2** 2 2 2 3 3 2 1 1 2 2
Makespan: **14**, Mean Tardiness: **62**

Shifted positions are shown with italic letters.

Table 5. Part data

Parts	*Process plan number*	*Process plans and processing sequence*	*Due dates*
1	1	op1–2–3–5	13
	2	1–6–5	
2	1	1–2–4–3	20
	2	1–6–4	
3	1	1–5–3	14
	2	1–3–5	
4	1	1–4–3	7
	2	1–3–4	
5	1	5–4–3	12
	2	5–3–4	
6	1	1–4–5	5
	2	1–5–4	

Table 6. Machine-operation suitability data

	*op*1	*op*2	*op*3	*op*4	*op*5	*op*6
*M*1		*		*		
*M*2	*		*	*		*
*M*3		*			*	
*M*4	*		*		*	
*M*5		*		*	*	
*M*6	*	*		*		*

5. Example application

The proposed linguistic and dispatching rule-based MOTS algorithm for FJSSP is programmed in C. The MOTS algorithm is tested on a Pentium III-MMX model PC at 450 MHz (128 MB RAM). In this section, the execution of the program is presented by using a sample problem with the data given in Tables 5–7. Makespan, total tardiness and load balance is selected as the objective functions. 10 different dispatching rules are used in the solution string.

The proposed algorithm converged into 18 Pareto optimal solutions that are shown in Table 8.

Table 7. Processing time data

M	*O1*						*O2*						*O3*					
O	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*
*M*1							2	1	0	0	0	0						
*M*2	3	4	7	2	0	6							4	5	3	2	3	0
*M*3							1	2	0	0	0	0						
*M*4	2	5	6	1	0	5							5	6	2	3	3	0
*M*5							2	1	0	0	0	0						
*M*6	3	6	5	2	0	6	1	2	0	0	0	0						
	O4						*O5*						*O3*					
	P1	*P2*	*P3*	*P4*	*P5*	*P6*	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*
	0	4	0	1	2	1												
	0	5	0	2	1	2							6	5	0	0	0	0
							3	0	4	0	6	7						
							2	0	3	0	5	6						
	0	6	0	2	2	1	3	0	4	0	7	7						
	0	4	0	1	1	1							7	4	0	0	0	0

Table 8. The Pareto optimal solutions

Pareto optimal solutions		
Makespan	*Total tardiness*	*Load balance*
15.00	7.00	0.26
20.00	2.00	0.19
18.00	4.00	0.28
19.00	3.00	0.19
22.00	4.00	0.11
17.00	5.00	0.22
19.00	11.00	0.10
19.00	13.00	0.09
20.00	10.00	0.11
19.00	7.00	0.12
17.00	16.00	0.10
16.00	6.00	0.14
17.00	8.00	0.13
20.00	23.00	0.08
18.00	14.00	0.09
18.00	12.00	0.12
22.00	26.00	0.06
18.00	5.00	0.16

6. Discussion

The classical modeling of job shop scheduling problem does not reflect requirements of modern job shops. Machines in the modern job shops are very capable and there is a considerable amount of overlapping capability between machines. It is logical to conclude that management should make use of this flexibility while operating the job shop in order to maximize the job shop performance and responsiveness. Otherwise, the advantage of having very capable machines might disappear. On the other hand, consideration of flexibility in shop job scheduling will dramatically increase the complexity of the problem, which is already very hard to solve. This will certainly increase the cost of the solution. Therefore, new ways of solving this important real life problem must be investigated. An effort is made in this paper to model and solve the problem effectively by employing grammars.

7. Conclusion

In this research paper, an extended model for FJSSP is presented. The model makes use of the grammars from linguistics to represent the FJSSP data, and the dispatching rules for sequencing of operations. For the proposed optimization model, a meta-heuristic based on MOTS algorithm is developed.

Employing grammars simplified the model formation process for the FJSSP problem. This simplification in the model formation has enabled the development of meta-heuristic procedures like the present MOTS for the solution of the problem. In the paper, the proposed approach is explained in detail by solving an example problem. The proposed algorithm is programmed in C.

Although there is a huge amount of literature on classical JSSP, the FJSSP does not have a rich literature. Therefore, there is a still need to develop effective models and solution techniques for this complex problem. The grammars can be considered effective mechanisms form this point of view.

References

Al-Fawzan, M. A. and Al-Sultan, K. S. (1996) A tabu search algorithm for minimizing the makespan in a job shop scheduling. *Proceedings of the 5th Industrial Engineering Research Conference*, Minneapolis, MN, USA, May 1996, pp. 115–119.

Baykasoğlu, A. (2002) Linguistic based meta-heuristic optimisation model for flexible job shop scheduling. *International Journal of Production Research*, **40**(17), 4523–4543.

Baykasoğlu, A., Owen, S. and Gindy, N. (1999) A taboo search based approach to find the Pareto optimal set in multiple objective optimisation. *Journal of Engineering Optimization*, **31**, 731–748.

Baykasoğlu, A., Saad, S. M. and Gindy, N. (1998) A loading approach for cellular manufacturing systems. *FAIM'1998: 8th International Conference on Flexible Automation and Intelligent Manufacturing*, July 1–3 1998, Portland, Oregon, USA, pp. 215–226.

Ben-Daya, M. (1994) Solution methodologies for scheduling problems in flexible manufacturing systems. *International Journal of Manufacturing Systems Design*, **1**(4), 315–328.

Blackstone, J. H., Phillips, D. T. and Hogg, G. L. (1982) A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, **20**(1), 27–45.

Chambers, J. B. (1996) *Classical and flexible job shop scheduling by Tabu search*, Ph.D. Thesis, Operations Research and Industrial Engineering, The University of Texas at Austin.

Cheng, R., Gen, M. and Tsujimura, Y. (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms-1, Representation. *Computers and Industrial Engineering*, **30**(4), 983–997.

Dorndorf, U. and Pesch, E. (1995) Evolution based learning in a job shop scheduling environment. *Complex Systems*, **22**, 25–40.

Fu, K. S. (1974) *Syntactic Pattern Recognition*, Academic Press, London.

Giffler, B. and Thompson, G. (1960) Algorithms for solving production scheduling problems. *Operations Research*, **8**, 487–503.

Glover, F. (1990) Tabu search: a tutorial. *Interfaces*, **20**, 74–94.

Jansen, K., Mastrolilli, M. and Solis-Oba, R. (2000) Approximation algorithms for flexible job shop problems. *Proceedings of Latin American Theoretical Informatics (LATIN'2000)*, LNCS 1776, July 21, 2000, pp. 68–77.

Kusiak, A. (1990) *Intelligent Manufacturing Systems*, Prentice Hall, U.S.A.

Kusiak, A., Chen, M. (1988) Expert systems for planning and scheduling manufacturing systems. *European Journal of Operational Research*, **34**, 113–130.

Mastrolilli, M. and Gambardella, L. M. (2000) Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, **3**(1), 3–20.

Maturana, F., Gu, P., Naumann, A. and Norrie, D. H. (1997) Object oriented job-shop scheduling using genetic algorithms. *Computers in Industry*, **32**, 281–294.

Mesghouni, K., Hammadi, S. and Borne, P. (1998) On modelling genetic algorithms for flexible job-shop scheduling problems. *Studies in Informatics and Control*, **7**(1).

Pinedo, M. (1995) *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, New Jersey.

Sönmez, A. I. and Baykasoğlu, A. (1998) A new dynamic programming formulation of $(n*m)$ flowshop sequencing problems with due dates. *International Journal of Production Research*, **36**(8), 2269–2283.

Upton, D. M. and Barash, M. M. (1988) A grammatical approach to routing flexibility in large manufacturing systems. *Journal of Manufacturing Systems*, **7**(3), 209–221.

Van Laarhoven, P. J. M., Aarts, E. H. L. and Lenstra, J. K. (1992) Job shop scheduling by simulated annealing. *Operations Research*, **40**, 113–125.