ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20130704132725

RULE BASED HEURISTIC APPROACH FOR MINIMIZING TOTAL FLOW TIME IN PERMUTATION FLOW SHOP SCHEDULING

Balasundaram Rathinam, Kannan Govindan, Baskar Neelakandan, Siva Sankar Raghavan

Original scientific paper

Production scheduling plays a vital role in the planning and operation of a manufacturing system. Better scheduling system has a significant impact on cost reduction and minimum work-in-process inventory. This work considers the problem of scheduling $n/m/F/\Sigma C_i$ using *Decision Tree* (DT) algorithm. Since this problem is known to be strongly *NP-hard*, this work proposes heuristic based methodology to solve it. The advantages of DT's are that the dispatching rule is in the form of *If-then else* rules which are easily understandable by the shop floor people. The proposed approach is tested on benchmark problems available in the literature and compared. The proposed work is a complement to the traditional methods.

Keywords: *decision tree algorithm and heuristics; flow shop scheduling; total flow time*

Na pravilu zasnovan heuristički pristup smanjenju ukupnog protoka vremena kod programiranja radova u permutacijskoj protočnoj radionici

Izvorni znanstveni članak

Programiranje radova u proizvodnji je od bitne važnosti u planiranju i funkcioniranju proizvodnog sustava. Unaprijeđeni sustav programiranja značajno utječe na smanjenje troškova i minimalni broj radnih postupaka. U ovom se radu razmatra problem programiranja $n/m/F/\Sigma C_i$ primjenom *Decision Tree* (DT) algoritma. Budući da je ovaj problem poznat kao veoma *NP-hard,* u radu se za njegovo rješenje predlaže metodologija temeljena na heuristici. Prednosti DT-a su u tome što je pravilo otpreme u obliku *If-then else* pravila koja radnici u radionici lako razumiju. Predloženi je pristup testiran na repernim problemima dostupnim u literaturi i uspoređen. Predloženi rad je dodatak tradicionalnim metodama.

Ključne riječi: *decision tree algoritam i heuristika; programiranje radova u protočnoj radionici; ukupni protok vremena*

1 Introduction

In many manufacturing and assembly facilities each job has to undergo a series of operations. Often these operations have to be done on all jobs in the same order implying that the jobs have to follow the same route. The machines are assembled in series and the environment is known as flow shop. It is characterized by a unidirectional flow of work with a variety of jobs being processed sequentially.The goal is to find a sequence of jobs for minimizing makespan, total flow time, idle time etc. The majority of flow shop scheduling research were on minimizing makespan. In recent years, scheduling problems with minimizing total flow time draw more attention from the researcher's community. This is due to the fact that smaller the value of total flow time better the utilization of resources and cost reduction. In this context, today's manufacturing environment with minimizing total flow time gives great practical importance. For finding optimal values of minimizing total flow time having *n-jobs* consist of *n!* sequences. Garey et al. [1] showed that flow shop scheduling problems are *NP-hard* even for moderate sized problems. As the problem size increases, *NP-hard* of the flow shop problem necessitates the development of heuristics and meta-heuristics to get good solutions. In this work, decision tree based heuristic methodology is proposed to discover the sequence for minimizing total flow time criterion.

2 Literature review

There are many heuristics and meta-heuristics that have been proposed over the years for solving the flow shop scheduling problems with the objectives of minimizing makespan, total flow time, idle time etc., of jobs considered either separately or simultaneously. The optimal or good solutions of *n*-jobs, *m*-machines flow shop scheduling can be obtained *via* non-traditional methods like Genetic Algorithm, Ant-Colony Algorithm, Scatter Search Algorithm and Hybrid Algorithms. The non - traditional methods often provide fast solutions to traditional flow shop problems but they do not demonstrate repeatability or provide an explanation of a solution that is developed. Also these methods require considerable amount of time for development of coding. In particular, for complex systems it may be difficult in practice to account for all relevant aspects in an optimization model (or) to elicit all relevant scheduling rules directly from an expert. These limitations have encouraged researchers to develop efficient heuristics. For all practical purposes, it is often more appropriate to look for a heuristics that generates a near-optimal solution at relatively minor computational expenses. This leads to the development of many heuristics. The various heuristics for makespan criterion are proposed by Johnson [2], Palmer [3], Campbell [4], Gupta [5], Nawaz [6], Valdimir Modrak [7] and Mircea Ancau [8]. For a total flow time criterion, Rajendran and Chaudhuri [9], Liu and Reeves [10] and Deepak Laha and Chakraborty [11] proposed various heuristics. Rajendran and Chaudhuri [9] proposed constructive heuristics based on the machine idle times and the job waiting times. Liu and Reeves [10] developed composite heuristics by appending jobs one by one using index function. The index function consists of weighted sum of total machine idle time and the artificial total flow time. Reeves [10] compared four versions of their composite heuristics with various existing heuristics and it was empirically shown that the composite heuristics are more efficient than the constructive heuristics. Deepak Laha and Chakraborty [11] proposed two composite heuristics H-1 and H-2 by hybridizing of (i) the constructive heuristics (ii) Simulated Annealing (*SA*) and (iii) the classic NEH algorithm for minimizing total flow time. In their algorithm, the jobs are sorted based on the ascending order of their total processing time on all machines and it is used as initial sequence for SA algorithm. The best solution generated by the SA is the initial sequence for heuristics H-1 and H-2 and it is improved by many iterations. The two composite heuristics produce better quality solutions than those produced by the composite heuristics of Liu and Reeves [10]. Recently, Hwang et al. [12] introduced optimal schedule block and polynomial time dynamic programming algorithm to solve total completion time in two machine flow shop scheduling problem with fixed job seqence. Kaizhou Gao et al. [13] presented two heuristics based on standard deviation and composite heuristic to solve no-wait flow shop scheduling problem for minizing total flow time criterion. Ji-Bo Wang et al. [14] proposed several dominance properties and some lower bounds to speed up the elimination process of a branch- and bound algorithm for two machine flow shop scheduling environment to minimize the total weighted completion time of jobs with decreasing linear deterioration. From the literature, it is evident the heuristic method of solving general *n-jobs* and *m-machines* flow shop scheduling problem for minimizing total flow time criterion received less attention from research community. Hence the proposed approach employed the heuristic method for solving total flow time criterion. The following section presents an overview of the data mining and decision tree algorithm.

2.1 Data mining

Data mining and knowledge discovery are emerging areas of research and it is a statistical method to learn unknown and useful *knowledge* from databases. It is the process of discovering interesting knowledge, such as patterns, associations, anomalies and significant structures of databases. Kuisak [15] showed that the use of data mining techniques in manufacturing began in early 1990's and it has gradually received attention from production community. Some of the most widely used data mining algorithms are Decision tree, Regression tree, Clustering, Neural networks etc. The knowledge discovered is often expressed in the form of *If-Then else* rules which has the advantage of being high level and symbolic knowledge representation and contribution towards the comprehensibility of the knowledge. Harding et al. [16] outlined applications of data mining in various fields of manufacturing engineering. Xiaonan Li and Sigurdur Olafsson [17, 18] presented decision tree based algorithm for discovering dispatching rule from production data for single machine scheduling. The authors have used Decision tree as a dispatching rule instead of famous rules such as Early Due Date [*EDD*] and Earliest Release Date [*ERD*]. In their later work, authors presented two phase approach by combining decision tree algorithm and genetic algorithm for finding weighted lateness in the single machine scheduling problem. Hyun-Seon Choi et al. [19] presented decision tree based approach for re-entrant hybrid flow shop problems with one or more parallel machines at each production stage. The case study was performed on Thin Film Transistor-Liquid Crystal Display (TFT-LCD) manufacturing line. The test results showed that the decision tree based approach is competitive to the simulation-based one with respect to various performance measures such as system throughput, mean flow time, mean tardiness and the number of tardy jobs. Atif Shahzad and Nasser Merbark [20] presented data mining based approach for job shop scheduling to discover set of rules capable of approximating the efficient solutions provided by the *tabu search* for minimizing the maximum lateness. In recent time, Shi Ling and Cheng Xue [21] proved that minimizing the total completion time is *NP-hard* in the strong sense through a reduction from the Numerical Matching with Target sums. The authors presented their algorithm for two - machine flow shop scheduling with a single server and equal server times. From the literature, most of the authors have used decision tree algorithm to discover the set of rules by analyzing efficient solutions from the meta-heuristics. These rules are used to duplicate the meta-heuristics algorithm to solve simillar problems. Only few authors used direct application of data mining algorithms to solve the scheduling problems. Hence the proposed approach uses direct application of data mining algorithm to solve total flow time for general *n-jobs* and *m-machines* flow shop scheduling problems.

2.2 Decision tree

Decision Tree (*DT*) is a supervised machine learning method for constructing prediction models from data. The advantages of *DT*'s are that they are easy to use and efficient. The rules can be generated that are easy to interpret and understand. A decision tree algorithm constructs a tree *T* from a set of data with many attributes. Quinlan [22] developed algorithm for construction of decision tree called Iterative Dichotomiser 3 (ID3) and improved version is C4.5. The algorithm searches through the attributes of the instances and extracts the appropriate splitting attributes that separate the given examples. If the attribute perfectly classifies the data sets, then algorithm stops, otherwise it recursively operates on them (where *the m = number* of possible values of an attribute) partitioned subsets to get their *best* attribute. The algorithm picks the best attribute and never looks back to reconsider earlier choices.

This work is organized as follows. The section 3 describes the formulation of the flow shop problem. The section 4 elaborates decision tree based approach for discovering knowledge in flow shop scheduling. The computational experiments of proposed method are applied to benchmark problems discussed in Section 5 and Section 6 includes results and discussion.

3 Problem formulation

The flow shop sequencing problem generally consists of *m*- machines and *n*-jobs, each job consists of *m* operations and each operation requires a different machine. *N*-jobs are to be processed in the same sequence on *m*-machines. The processing time of job *i* on the machine *j* is given by $P_{ij}(i = 1, 2,\dots, n; j = 1, 2,\dots, m)$.

The objective is to find the best sequence of jobs, which will give minimum total flow time.

3.1 Permutation flow shop representation

The permutation flow shop represents a typical case of the flow shop scheduling problem with a goal to determine the optimal schedule for n-jobs on m-machines. Let $C_{(Ji,\,k)}$ denote the completion time of job J_i on machine k and let $\{J_1, J_2,\dots, J_n\}$ denote a job permutation. The calculation of total flow time for n-jobs and m-machines flow shop problem is as follows:

$C_{(j1,\,1)} = P_{j1,\,1}$
$C_{(j1,k)} = C_{(j1,k-1)}+P_{j1k}, \quad k = 2, 3,\dots, m$
$C_{(ji,1)} = C_{(j(i-1),1)}+P_{ji1}, \quad i = 2, 3,\dots, n$
$C_{(ji,k)} = \max \{C_{(j(i-1),k)}, C_{(ji,k-1)}\}+P_{jik},$
$i = 2, 3,\dots, n,\ k = 2, 3,\dots, m$
Makespan $C_{\max} = C_{(jn,m)}$
Total Flow Time (TFT) = $\Sigma C_{(jn,m)}\ \ j = 1, 2,\dots, n.$

The standard notation for the flow shop problem is denoted by $n/m/F/\Sigma C_i$, considering the total flow time as the objective function.

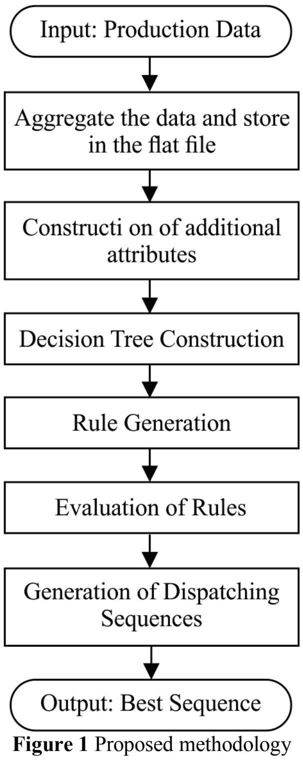

Input: Production Data
Aggregate the data and store in the flat file
Constructi on of additional attributes
Decision Tree Construction
Rule Generation
Evaluation of Rules
Generation of Dispatching Sequences
Output: Best Sequence

Figure 1 Proposed methodology

4 Proposed schema

Fig. 1 shows the proposed framework. In production data module, data related to the number of jobs, numbers of machines and processing time of each job are to be entered. Decision Tree is concerned with learning some target concept. In particular, given the *job*-1 and *job*-2, which job should be scheduled first? As far as flow shop scheduling is concerned with the total flow time of job order 1-2 and 2-1 is calculated. The job order having minimum total flow time value is scheduled first. Likewise, *job*-1 is compared with remaining $n-1$ jobs to schedule whether *job*-1 is scheduled first or not. Given this target concept, a classification problem can be defined as a pairwise comparison of jobs. That gives *job*-1 and *job*-2 instances in the data set for the classification problem includes all of the production data for those two jobs along with *a class attribute* that indicates whether *job*-1 (or) *job*-2 is scheduled first. For set of *n-jobs*, the first job is compared with $n-1$ jobs, the second job is to be compared with remaining $n-2$ jobs and henceforth. The total number of instances for n-jobs are represented in Eq. (1).

$$\sum_{j=1}^{n-1}(n-j). \tag{1}$$

Thus the flat file for the data set is constructed. The engineering of this database plays a critical role for the usefulness of the knowledge discovered. Hence this approach involves creation of additional attributes such as total processing time of each job and processing time difference on each machine in the same job is constructed to improve the scheduling decision. After construction of additional attributes, the decision tree algorithm is applied. The construction of Decision Tree for numerical data is illustrated in *Annexure-I*. From decision tree, decision rules are generated. In order to get the initial sequence, *job*-1 is compared to remaining $n-1$ jobs based on rules and its priority is checked. Likewise all the jobs are compared to the remaining jobs and accuracy of each job was found. Based on the accuracy of jobs, three passes are generated. A job which satisfies 100 % accuracy is placed in *pass*-1, the one with the accuracy greater than 50 % will occupy *pass*-2 and the remaining jobs will occupy *pass*-3. By combining three passes, the initial dispatching sequence is obtained. To get more population, jobs are swapped within their passes and the best value is found.

5 Computational experiments

To compare the proposed decision tree algorithm bench mark data sets are taken from Taillard [23] for this study. Xiao Xu et al. [24] developed asynchronous genetic local search algorithm for total flow time minimization of flow shop scheduling and compared the results with various non-traditional methods. The asynchronous genetic algorithm gives best value of total flow time for Taillard [23] data sets. Deepak Laha [11] compared his composite heuristics H-1 and H-2 with Reeves [10] composite heuristics and Simulated Annealing Algorithm. In this study, 60 problems are taken for analysis. All the problems were tested with population size of 1000 and twenty independent trials have been made. The Decision tree based algorithm was coded in Java and run on Pentium IV, 3-GHz processor with 1 Mb Ram. The Percentage of Relative Deviation (*PR*D) is used to measure the algorithm quality.

$PRD = [C_{(\text{Heu})}-C_{(\text{best})}] /(C_{(\text{best})}\times 100),$

where $C_{(\text{Heu})}$ represents total flow time obtained from various heuristics methods and $C_{(\text{best})}$ represents best values of total flow time. The average values of *PRD* for various composite heuristics and SA are calculated and are reported in the Tab. 1.

Table 1 Comparison of average value of PRD for different heuristics (*LR* = Liu and Reeves heuristics, *SA* = Simulated Annealing algorithm, *DL* = Deepak Laha's composite heuristics H-1, H-2 and *DT* = proposed decision tree based approach)

Problem size (Job × Machine)	*LR*	*SA*	*DL*		*DT*-proposed method
			H-1	H-2	
20 × 5	1,3610	10,599	0,5711	0,9256	0,9075
20 × 10	1,4329	7,7483	0,6397	0,6749	1,2385
20 × 20	1,2241	6,2548	0,8657	0,9610	1,0684
50 × 5	1,7256	13,512	1,6573	1,4886	2,5940
50 × 10	2,9070	15,2570	2,5682	2,1207	2,9993
50 × 20	2,9028	12,4847	2,1639	2,2493	3,6056
Average	1,9256	10,9761	1,4110	1,4033	2,0689

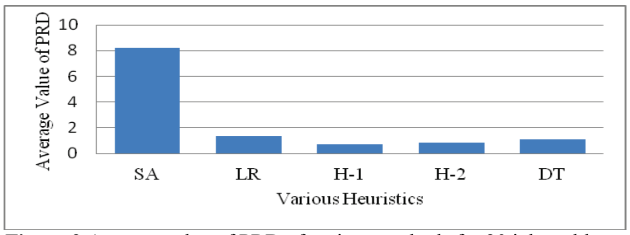


Figure 2 Average value of PRD of various methods for 20 job problems

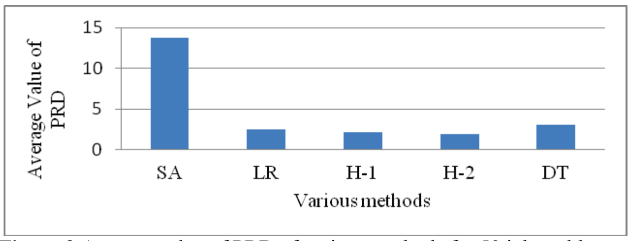


Figure 3 Average value of PRD of various methods for 50 job problems

Figs. 2 and 3 show the average value of *PRD* for 20 and 50 job problems.

Data engineering plays major role in building more accurate interpretable models and providing factors that are most important in scheduling decisions. Specifically, the attributes that are recorded during production process may not be the attributes that are not useful for constructing a decision tree. In most of the cases, it could be done manually using intuitive process. Tab. 2(a) shows the total number of instances generated for each data set and average value of tree size before and after data engineering. Tab. 2(b) shows average value of accuracy of tree for various datasets.

Figs. 4 and 5 show the average value of size and accuracy of tree for 20 and 50 job problems before and after data engineering.

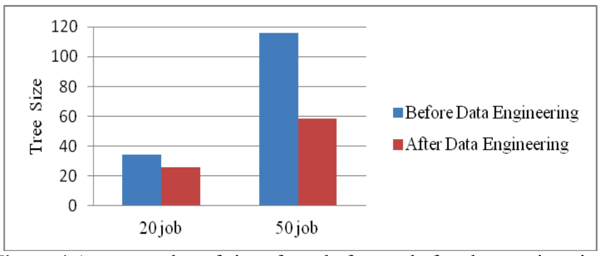


Figure 4 Average value of size of tree before and after data engineering

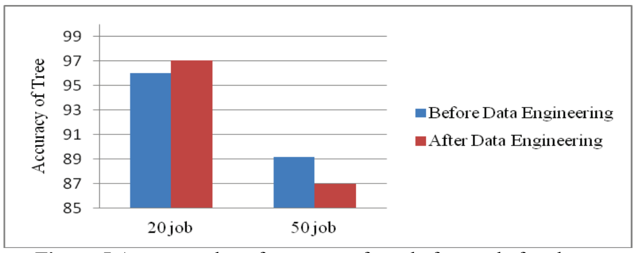


Figure 5 Average value of accuracy of tree before and after data engineering

Table 2(a) Size of decision tree for the data sets proposed by Taillard [23]

Sl. No.	Data set size	Total number of instances	Before data engineering			After data engineering		
			Minimum	Maximum	Average	Minimum	Maximum	Average
1	20 × 5	190	25	47	36	19	37	27,6
2	20 × 10		29	41	35,4	21	33	26,6
3	20 × 20		21	41	31,8	15	31	23,2
4	50 × 5	1225	103	133	119	25	107	63
5	50 × 10		101	129	118,2	21	81	54,4
6	50 × 20		101	125	110,2	41	87	58,4

Table 2(b) Accuracy of decision tree for the data sets proposed by Taillard [23]

Sl. No.	Data set size	Before data engineering			After data engineering		
		Minimum	Maximum	Average	Minimum	Maximum	Average
1	20 × 5	87,894	97,894	93,9995	93,157	99,473	97,6836
2	20 × 10	95,263	98,421	96,8418	94,736	98,947	97,3153
3	20 × 20	95,789	98,421	97,1575	85,368	99,473	96,1680
4	50 × 5	80,653	92,081	86,5790	72,163	97,469	85,0689
5	50 × 10	85,795	92,734	89,3872	74,285	90,040	89,4934
6	50 × 20	85,236	94,693	91,5517	82,857	92,653	86,3914

6 Results and Discussions

The decision tree based heuristic approach was tested with 60 benchmark problems of Taillard [23]. The results are compared with Simulated Annealing algorithm and other composite heuristics in literature. Tab. 1 shows the average value of *PRD* for all the 60 problems. The average value *PRD* of proposed work is 2,068, where as in *SA* it is 10,976. From this, it is evident that the proposed approach is better than the *SA*. Fig. 2 shows average value of *PRD* for 20 job problems (total number of problems - 30). The average value of PRD proposed method is 1,071, whereas in Liu and Reeve's [10] approach it is 1,339. As compared to Liu and Reeve's approach, the proposed method gives good solutions. But, as the job size increases more than 20, the proposed method average PRD is slightly higher than the Reeve's composite approach. The composite heuristics H-1 and H-2 give a lower value as compared to Reeve's heuristics and proposed method. Reeve used index function to append the job one by one and both forward & backward pairwise exchange of jobs are used as local search in their algorithm. Laha's [11] approach consists of hybridizing of constructive heuristics, SA and NEH algorithm were done. The SA was run from 475° to 20° and then the best value was improved in several iterations. The decision tree based approach developed in this work is alike, with improved heuristics and it is a single-pass method that requires very few computations. The advantage of the proposed approach is that the rules are in the form of *if-then else* rules that can be easily understandable by the shop floor people. To know the significance of additional attributes constructed for the tree, Tab. 2 shows the size and accuracy of tree before and after the addition of attributes. Fig. 4 shows average value of size of decision tree with and without considering additional attributes. The construction yields the decrease in size of the decision tree. For 20 job problems, the reduction in tree size is 25 % whereas for 50 job problem it is 49,39 %. Fig. 5 shows average value of accuracy of decision tree with and without considering additional attributes. For 20 job problems, construction of additional attributes yields an increase in accuracy 1,103 % higher than the basic attributes. For 50 job problems, construction of addtional attributes yields the reduction in accuracy of 2,554 % as compared to basic attributes. This indicates that the job size increases more than 20, there is a small reduction in accuracy with significant improvement in reduction of the tree size. Overall, the above attributes constructed for constructing tree give significant improvement in tree reduction with small change in the value of accuracy for large size problems.

7 Conclusions

A rule based heuristic approach is presented for flow shop scheduling that is a well-known combinatorial optimization problem. To compare the proposed work, 60 problems are analyzed for standard well-known bench mark problems and compared against simulated annealing algorithm and various composite heuristics. The expertimental results indicate that the proposed algorithm contributes significantly to the extremely challenging scheduling problem. Unlike existing heuristics available for flowshop, the prosed work is like rule (or) tree based structure in the form of *IF-THEN* else rules which are easily understandable by even semi- skilled workers. The advantages of *DT*s are easy to use and efficient.

In real time applications, more data and attributes are collected in a shop floor control system and tree constructed from these attributes will lead to better dispatching rules. Whereas, it is impossible to elicit all relevant aspects and *knowledge* of the scheduling to the other approaches. The proposed work can be extended in several directions. First, to construct the tree, other algorithms can be used. Second, to obtain optimal or near optimal solution, initial seed solution is taken from the proposed method and then any one of the meta-heuristics could be implemented. Third, as size of the job increases size of the decision tree also increases. The various instance selection methods could be implemented for reducing the tree size with good solution accuracy.

8 References

[1] Garey, M. R.; Johnson, D. S.; Ravi Sethi. The complexity of flow shop and Job shop scheduling. // Mathematics of Operations Research. 1, 2(1976), pp. 117-129.

[2] Johnson, S. M. Two and three stage production schedules with setup times included. // Naval Research Logistics Quarterly. 1, 1(1954), pp. 61-68.

[3] Palmer, D. S. Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum. // Operation Research Quarterly. 16, 1(1965), pp. 101-107.

[4] Campbell, H. G.; Dudek, R. A.; Smith, M. L. A heuristic algorithm for the n-job, m-machine sequencing problem. // Management Science. 16, 10(1970), pp. 630-637.

[5] Gupta, J. N. D. A functional heuristic algorithm for the flow shop scheduling problem. // Operation Research Quarterly. 22, 1(1971), pp. 39-47.

[6] Nawaz, M.; Enscore, E. E.; Ham, I. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. // OMEGA International Journal of Management Science. 11, 1(1983), pp. 91-95.

[7] Modrak, V.; Pandian, S. R. Flow shop Scheduling Algorithm to Minimize completion time of n-Jobs and m-Machines Problem. // Tehnicki vjesnik-Technical Gazette. 17, 3(2010), pp. 273-278.

[8] Ancau, M. On Solving Flowshop Scheduling problem. // Processing of the Romanian Academy – Series A. 13, 1(2012), pp. 71-79.

[9] Rajendran, C.; Chaudhuri. D. An efficient heuristic approach to the scheduling of jobs in a flow shop. // European Journal of Operational Research. 61, 3(1992), pp. 318-325.

[10] Liu, J.; Reeves, C. R. Constructive and composite heuristic solutions to the P││ΣCi scheduling problem. // European Journal of Operational Research. 132, 2(2001), pp. 439-452.

[11] Deepak Laha; Uday K. Chakraborty. An efficient heuristic approach to total flow time minimization in permutation flow shop scheduling. // International Journal of Advanced Manufacturing Technology. 38, 9-10(2008), pp. 1018-1025.

[12] Hwang, F. J; Kovalyov, M. F; Lin, B. M. T. Total completion time minimization in two-machine flow shop scheuding with a fixed job seqeunce. // Disctrete Optimization. 9, 1(2012), pp. 29-39.

[13] Kaizhou Gao; Quanke Pan; Suganthan, P. N; Junqing Li. Efficient heuristics for the no-wait flow shop scheduling problem with total flow time minimization. // International journal of Advanced Manufacturing Technology. 66, 9-12(2013), pp. 1563-1572.

[14] Ji-Bo Wang; Ming-Zheng Wang. Solutions algorithms for the total weighted completion time minization flow shop scheduling with linear deterioration. // International journal of Advanced Manufacturing Technology. 67, 1-4(2013), pp. 243-253.

[16] Kusiak, A. Data Mining: Manufacturing and service Applications. // International Journal of Production Research. 44, 18-19(2006), pp. 4175- 4191.

[16] Harding, J. A.; Shahbaz, M.; Srinivas and Kusiak. A. Data mining in Manufacturing: A review. // Journal of Manufacturing Science and Engineering – ASME. 128, 4(2006), pp. 969 -976.

[17] Xiaonan Li; Sigurdur Olafsson. Discovering Dispatching Rules Using Data Mining. // Journal of Scheduling. 8, 6(2005), pp. 515-527.
[18] Xiaonan Li; Sigurdur Olafsson. Learning Effective new single machine dispatching rules from optimal scheduling data. // International Journal of Production Research. 128, 1(2010), pp. 118-126.
[19] Hyun-Seon Choi; Ji-Su Kim; Dong-Ho Lee. Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line. // Expert systems with Applications. 38, 4(2011), pp. 3514-3521.
[20] Atif Shahzad; Nasser Merbark. Data mining based job dispatching using hybrid simulation-optimization approach for shop scheduling problem. // Journal of Engineering Applications of Artificial Intelligence. 25, 6(2012), pp. 1173-1181.
[21] Shi Ling; Cheng Xue-guang. Two- machine flow shop scheduling problems with minimizing the total completion times. // Applied Mathematical Sciences. 6, 39(2012), pp. 3437-3411.
[22] Quinlan, J. R. Induction of Decision Trees. // Machine Learning. 1, 1(1986), pp. 81-106.
[23] Taillard, E. Benchmarks for basic scheduling problems. // European journal of Operational Research. 64, 2(1993), pp. 278-285.
[24] Xiao Xu; Zhenhao Xu; Xingsheng Gu. An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization. // Expert systems with Applications. 38, 7(2011), pp. 7970-7979.

Appendix: Decision Tree construction for Numerical Data

Step: 1 (Input the data set)

Let us consider the 5 jobs × 2 machines flow shop problem. Tab. 3 shows the data set of 5 jobs and 2 machines problem.

Table 3A typical 5×2 flow shop problem

Machine & Job	J_1	J_2	J_3	J_4	J_5
M_1	7	3	5	1	6
M_2	5	6	2	2	6

Step: 2 (Flat file construction)

The data set is converted into flat file as illustrated in the section 4.0. i.e., for the given job order, 1-2 and 2-1 total flow time is calculated which is shown in Tabs. 4 and 5 respectively.

Table 4 Calculation of total flow time for Job 1 – 2

Job	Machine-1		Machine-2	
	Input	Output	Input	Output
1	0	7	7	12
2	7	10	12	18

Table 5 Calculation of total flow time for Job 2 – 1

Job	Machine-1		Machine-2	
	Input	Output	Input	Output
2	0	3	3	9
1	3	10	10	15

For the given job 1-2 the total flow time is 30, whereas in reverse case it is 24. Therefore in the given job 1-2, *job*-2 is scheduled first. Likewise, pairwise comparisons of remaining jobs have to be done and flat file has to be constructed. For five jobs, total number of instances is 10. The attributes J_1P_{m1}, J_1P_{m2} etc., are called *predictor variable* and the attribute, job-1 scheduled first is called *class attribute*. Where J_iP_{mj} represents job j_i is processed into machine m_j. In order to obtain more meaningful decision tree and hence dispatching rules, a better data file must be constructed before the decision tree induction. To illustrate the potential benefit of the approach, the following attributes are deliberately added that are believed to be helpful.

i) TP_1– total processing time of *job*-1 ($J_1P_{m1} + J_1P_{m2}$)
ii) TP_2– total processing time of *job*-2 ($J_2P_{m1} + J_2P_{m2}$)
iii) $P_{m1\text{diff}}$ – processing time difference of $J_1P_{m1} - J_2P_{m1}$
iv) $P_{m2\text{diff}}$ – processing time difference of $J_1P_{m2} - J_2P_{m2}$
v) $T_{\text{diff}} = TP_1 - TP_2$

Tab. 6 shows the resulting flat file for the given problem.

Table 6 Flat file for 5×2 flow shop problem

Instance No.	*Job*–1	J_1P_{m1}	J_1P_{m2}	TP_1	*Job*–2	J_2P_{m1}	J_2P_{m2}	TP_2	P_{m1}diff	P_{m2}diff	T_{diff}	*Job*–1 scheduled first
1	1	7	5	12	2	3	6	9	4	-1	3	No
2	1	7	5	12	3	5	2	7	2	3	5	No
3	1	7	5	12	4	1	2	3	6	3	9	No
4	1	7	5	12	5	6	6	12	1	-1	0	No
5	2	3	6	9	3	5	2	7	-2	4	2	Yes
6	2	3	6	9	4	1	2	3	2	4	6	No
7	2	3	6	9	5	6	6	12	-3	0	-3	Yes
8	3	5	2	7	4	1	2	3	4	0	4	No
9	3	5	2	7	5	6	6	12	-1	-4	-5	Yes
10	4	1	2	3	5	6	6	12	-5	-4	-9	Yes

After constructing flat file for the given dataset, the decision tree algorithm is applied. i.e., for each attribute information gain has been calculated and the attribute that possesses maximum information gain is taken as a root node.

Step: 3 (Decision tree construction and rule evaluation)

The value of information gain or entropy, defined by Claude E. Shannon as Entropy
$(P_1, P_2,\dots, P_n) = -P_1\cdot\log_2^{P1} - P_2\cdot\log_2^{P2} - \dots - P_n\cdot\log_2^{Pn}$
In general: info($[C_1, C_2,\dots, C_n]$) = entropy $(P_1, P_2,\dots, P_n)$

Iteration:

Step 3(i): Calculation of information gain for '*class attribute'*

Number of '*yes*' – 4

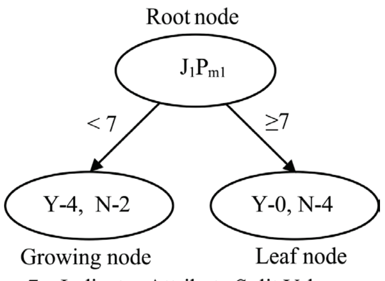
Number of '*No*' – 6

$$\text{Info }([4, 6]) = -4/10 \log_2^{(4/10)} - 6/10 \log_2^{(6/10)} = 0{,}5287 + 0{,}4421 = 0{,}9708 \textit{ (over all gain)}$$

Step: 3(ii) Finding root node (Calculation of information gain for predictor variable)

1) Calculation of Information gain and tree structure for attribute J_1P_{m1} Split of attribute J_1P_{m1} based on processing time

Processing time	No. of Yes	No. of No's
7	0	4
3	2	1
5	1	1
1	1	0



7 – Indicates Attribute Split Value

Info (4,2) = $-4/6 \log_2^{(4/6)} - 2/6 \log_2^{(2/6)} = 0{,}917$

Info (0,4) = $-4/4\log_2^{(4/4)} = 0$

Combined info {(4,2), (0,4)} = 6/10 × 0,917 + 2/10 x 0 = 0,5504

Gain of attribute J_1P_{m1}

= Overall info gain – combined info of J_1P_{m1}

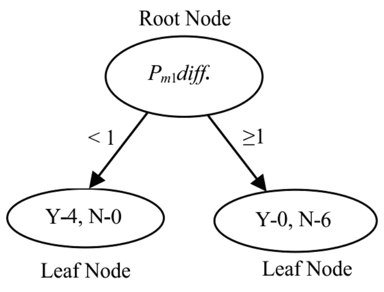
= 0,9708 – 0,5504 = 0,4204.

For finding suitable attribute split value (processing time) of each predictor attribute, the attribute values are arranged in ascending order and information gain has been calculated for both upward and downward directions. The value which gives maximum information gain has been chosen as an attribute split value. In the above attribute, the processing unit 7 gives maximum information gain. A likewise information gain of all the attributes is calculated as shown in Tab. 7.

Table 7 Information gain of various attributes for the flat file

Attribute name	Information gain
J_1P_{m1}	0,4204
J_1P_{m2}	0,09128
TP_1	0,41997
J_2P_{m1}	0,41997
J_2P_{m2}	0,12451
TP_2	0,28129
$P_{m1\text{diff}}$	0,60999 (max. value)
$P_{m2\text{diff}}$	0,01997
T_{diff}	0,41997

Among all the attributes, the attribute $P_{m1\text{diff}}$ possesses maximum information gain which has been selected as a root node. Fig. 6 shows the tree structure of attribute $P_{m1\text{diff}}$.



1 - Indicates Processing Time in the $P_{m1}diff.$

Figure 6 Decision tree constructed from the attribute $P_{m1\text{dif}}$

The decision rule derived from the above tree is as follows:

If $P_{m1\text{diff}} < 1$, *job*-1 *scheduled first* ---- Rule (1)

If $P_{m1\text{diff}} \geq 1$, *job*-2 *scheduled first* ---- Rule (2)

The left side branch of the root node (*rule*-1) correctly classifies instances 5, 7, 9 and 10. The right side of the root node (*rule*-2) correctly classifies remaining instances. As all the instances in the flat file are classified, the tree construction is stopped else it recursively operates until all the instances in the data set are classified. The '*Yes*' in the leaf node implies that *job*-1 is dispatched first and vice versa. The size of the above tree is 3 and having 100 % accuracy. The decision tree or alternatively the corresponding rules can be used directly to dispatch jobs. In order to get initial sequence, the above rules are applied to the data set and accuracy of each job was found. Tab. 8 shows accuracy of each job of example problem.

Table 8 Accuracy of jobs for the given 5×2 flow shop problem

Job	Accuracy (%)
J1	0
J2	75
J3	50
J4	100
J5	25

Based on the accuracy of the jobs, three phases are generated. Jobs which satisfy 100 % accuracy are placed in *pass*-1, if the accuracy is greater than 50 % they will occupy *pass*-2 and the remaining jobs will occupy *pass*-3. By combining three passes, dispatching sequence is obtained. Tab. 9 illustrates obtaining of initial sequence.

Table 9 Framing of initial sequence

pass-1	*pass*-2	*pass*-3
J_4	J_2, J_3	J_5, J_1

To obtain the best solution, jobs are randomly swapped within their pass for a given population and different dispatching sequence has been obtained. In the above example, *pass*-1 has one job and *pass*-2&3 have two jobs. To evaluate algorithms, the problem was tested with a trial run of 10 times with population size of 20. The sequence 4-2-3-5-1 gives minimum total flow time of 73 units. In the above example, if a tree is constructed with basic attributes, then the final tree is shown in Fig. 7.

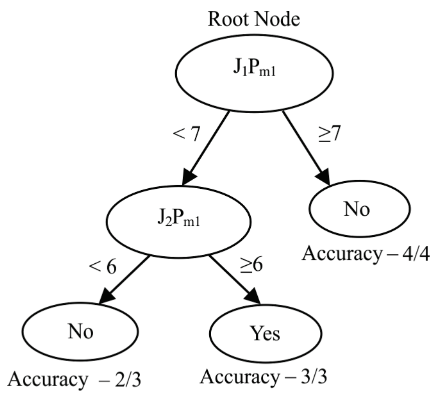


Figure 7 Decision Tree for the 5×2 problem set without addtional attributes

The size of the above tree is 5 and its accuracy is 90 % (except for instance no. 5 the remaining instances are correctly classified). The construction of additional attributes yields a 40 % reduction in tree size and 10 % increase in accuracy as compared to the basic attributes. From this it is evident that the additional attribute constructed for decision tree gives a significant reduction in tree size and good improvement in the accuracy of the tree.

Authors' addresses

Balasundaram Rathinam
Department of Mechanical Engineering,
ShriAngalamman College of Engineering and Technology,
Siruganoor, Tiruchirappalli, Tamilnadu, Pin – 621 105, India
E-mail: balasundaram_rbs@yahoo.co.in

Kannan Govindan, Professor
Department of Business and Economics,
University of Southern Denmark,
5230 Odense, Denmark
E-mail: gov@sam.sdu.dk

Baskar Neelakandan, Professor
Department of Mechanical Engineering,
M. A. M. College of Engineering, Siruganoor,
Tiruchirappalli, Tamilnadu, India
E-mail: baskarnaresh@yaoo.co.in

Siva Sankar Raghavan
Caterpillar Inc., Engineering Design Center,
Chennai, Tamilnadu, India
E-mail: siva_r_sankar@yahoo.com