J Syst Sci Syst Eng (Feb 2017) 26(1):50-76 ISSN: 1004-3756 (Paper) 1861-9576 (Online)
DOI: 10.1007/s11518-016-5297-1 CN11-2983/N

A HYBRID APPROACH FOR MINIMIZING MAKESPAN IN PERMUTATION FLOWSHOP SCHEDULING

Kannan Govindan[1] R.Balasundaram[2] N.Baskar[3] P.Asokan[4]

[1]Center for Sustainable Engineering Operations Management, Department of Technology and Innovation, University of Southern Denmark, Odense M, Denmark-5230
kgov@iti.sdu.dk (✉)

[2]Department of Mechanical Engineering, K.Ramakrishnan College of Engineering, Tiruchirappalli, Tamilnadu, India, Pin-621112
balasundaram_rbs@yahoo.co.in

[3]Department of Mechanical Engineering, Saranathan College of Engineering, Tiruchirappalli, Tamilnadu, India, Pin 620012
baskarnaresh@yahoo.co.in

[4] Department of Production Engineering, National Institute of Technology, Tiruchirappalli, Tamilnadu, India, Pin-620015
asokan@nitt.edu

Abstract

This work proposes a hybrid approach for solving traditional flowshop scheduling problems to reduce the makespan (total completion time). To solve scheduling problems, a combination of Decision Tree (DT) and Scatter Search (SS) algorithms are used. Initially, the DT is used to generate a seed solution which is then given input to the SS to obtain optimal / near optimal solutions of makespan. The DT used the *entropy function* to convert the given problem into a tree structured format / set of rules. The SS provides an extensive investigation of the search space through diversification. The advantages of both DT and SS are used to form a hybrid approach. The proposed algorithm is tested with various benchmark datasets available for flowshop scheduling. The statistical results prove that the proposed method is competent and efficient for solving flowshop problems.

Keywords: Flowshop scheduling, makespan, decision tree algorithm, scatter search algorithm, hybrid algorithm

1. Introduction

Production scheduling plays a vital role in the successful operation of the planning and control department in an organization. It offers great theoretical challenges to researchers due to its combinatorial nature. In flowshop problems, it is commonly understood that all jobs are processed through machines in identical order (Permutation Flowshop). Among various objectives for flowshop scheduling, minimization of makespan refers to maximization of throughput and usage of resources (Baker 1974). Hence, it is not surprising that most of the research during recent decades concentrated on minimization of makespan. Johnson (1954) proposed a simple heuristics to minimize makespan for two machines problems. Owing to the simplicity of Johnson's algorithm many researchers extended this idea to general flowshop problems. Johnson's algorithm can be extended to solve three machines problems only if it obeys certain primary conditions. If the number of machines is more than three, then these problems are difficult to solve. The development of computer technology resulted in meta-heuristics which are used to solve flowshop problems to obtain good solutions. This work uses a hybrid approach to solve permutation flowshop problems on the makespan criterion.

2. Literature Review

For minimizing makespan, many variations of heuristics and meta-heuristics have been proposed over the years. The heuristic method of solving flowshop problems is divided into two methods: i) simple or constructive heuristics ii) improvement heuristics. Simple heuristics begin with an ordered sequence of jobs based on exact rules or decisions. The feasible solutions that exist are enhanced in improvement heuristics by executing a given procedure. Table 1 offers a summary of noteworthy heuristics for the makespan criterion.

Table 1 Summary of constructive and improvement heuristics for $F_m||C_{max}$

Author	*Year*	*Algorithm*	*Type*	*Remarks*
Nawaz et al.	1983	NEH	C	Sum of total processing time of all machines are used to develop initial seed. Best two-job partial sequence has been calculated. Total Number of sequences generated is $[n\ (n+1)/2]-1$.
Gupta & Stafford	2006	Review of flowshop scheduling and various approaches		
Agarwal et al.	2006	NEH + ALA	I	NEH/CDS are used to generate the initial seed and adaptive learning approach is applied to solve the problem.
Kalczynski & Kamburowski	2007	KK	I	NEH algorithm performance was enhanced by using special tie breaking rules.
Rad SF et al.	2009	RA	I	The algorithm used NEH and Local Search (LS) algorithm.
Modrák & Pandian	2010	MP	I	Converting m-*machines* problem into 2 -machines problem by set of procedures and solved by Johnson's method.
Mircea Ancau	2012	MA	I	Two algorithms are proposed. The first algorithm uses Selective Greedy heuristics and the second uses stochastic features to escape from local optima.

C- Constructive heuristics, I- Improvement heuristics

Agarwal et al. (2006) used slope index, CDS and NEH to build a good starting solution and applied weight parameter to obtain improved solutions. The test result showed that their upper-bound solutions were superior to many problems. Kalczynski & Kamburowski (2007) showed their algorithm produced a better than average error ratio over the NEH algorithm for Taillard (1993) problem sets. Rad SF et al. (2009) proposed five heuristics that outperformed NEH in most cases and demonstrated that using their proposed heuristics as a seed yielded better results.

Mircea Ancau (2012) developed two heuristics and numerical results proved that their algorithms were superior for large size problems as compared with NEH algorithm, but their algorithms required a relatively high computational degree. Among various heuristics, for general *n - jobs & m - machines* NEH (Nawaz et al. 1983) is known as an efficient heuristic due to its simplicity and solution quality. In general, heuristics often provide better results for tiny size problems, but for large size problems, due to computation complexity, they require more time to provide good solutions. To overcome this difficulty and to obtain near optimal / optimal solutions, meta-heuristics and hybrid algorithms are used. Meta-heuristics generally start from the initial population developed by heuristics and proceed to meet the stopping criterion. There is ample research with meta-heuristics for the Permutation Flowshop Scheduling Problems (PFSP). Table 2 summarizes the important papers on makespan criterion using meta-heuristics and hybrid algorithms.

Table 2 Summary of meta-heuristics and hybrid algorithms for $F_m||C_{max}$

Author	*Year*	*Algorithm*	*Type*	*Remarks*
Wang & Zheng	2003	HGA	H	Genetic algorithm with SA and local search
Ying & Liao	2004	ACS	M	Ant Colony Optimization
Rajendran & Ziegler	2004	M-MAS PACO	M	Two algorithms are proposed based colony optimization with the min - max system and modified initialization
Tasgetiren M.F. et al.	2004	PSO_{VNS}	H	Implementation of Hybrid Algorithm (Variable Neighborhood Search with Particle Swarm Optimization)
Tasgetiren M.F. et al.	2007	PSO_{SPV}	H	Particle Swam Optimization is combined with shortest position value
Liu B. et al.	2007	PSO_{MA}	H	PSO based with memetic algorithm
Rajkumar & Shahabudeen	2008	1) SA + PS 2) SA + RIPS 3) SA + CRPIS 4) SA + RIPS + PS 5) SA + CRPIS + PS	H	Five algorithms are proposed based on Simulated Annealing. The entire algorithm used NEH as an initial seed solution

Author	*Year*	*Algorithm*	*Type*	*Remarks*
Saravanan M. et al.	2008	NEH + SS	H	The initial seed generated from NEH algorithm acts as an input to Scatter Search (SS) algorithm
Jarboui B. et al.	2008	H-CPSO	H	PSO combined with improvement procedure
Pan Q. et al.	2008	DDE	M	Discrete Differential Evolution algorithm
Laha & Chakraborty	2009	NEH + SA + C	H	Uses a combination of Simulated Annealing, NEH, and Composite heuristics
Chen S. et al.	2009	ACGA	H	The convergence criterion of GA was improved by using artificial chromosomes
Chen S. et al.	2012	1) SGA 2) Self-Guided GA 3) ACGA 4) GMA 5) MGGA	1- M 2,3,4,5 - H	Five algorithms were proposed based on Genetic Algorithm. The new strings are improved by probabilistic model
Liu F.Y. & Liu S.Y.	2013	HDBAC	H	The new algorithm was developed and implemented by combining heuristic and meta heuristic algorithms
Chang P.C. et al.	2013	p-ACGA	H	Blocking mining with enhanced recombination operator was applied to GA. New populations are generated by heuristics, weighted gene structure and it is improved by local search algorithm

M – Meta-heuristics H−Hybrid algorithms

Wang & Zheng (2003) proposed a crossbreed algorithm in which NEH is used to produce trial solution and mutation operators are replaced by SA algorithms. Rajendran & Ziegler (2004) implemented two ACO algorithms which incorporated the concept of summation rule and local search. Liu et al. (2007) developed PSO-based memetic algorithm by combining NEH and a SA based local search. Rajkumar & Shahabudeen (2008) proposed five algorithms based on a simulated annealing algorithm with different neighborhood schemes. Saravanan et al. (2008) applied scatter search algorithm where the initial seed solution was obtained from NEH heuristics. Pan et al. (2008) developed novel discrete differential evolution to solve general flowshop scheduling problem. Laha & Chakraborty (2009) proposed a new hybrid algorithm using a simulated annealing algorithm in conjunction with NEH heuristics. Chen et al.

(2009) used simulated populations to improve the GA and it was generated based on weighted gene structure. Chen et al. (2012) developed five different variations of Genetic Algorithms based on probabilistic model. Liu F.Y. & Liu S.Y. (2013) developed a hybrid discrete artificial bee colony algorithm which used NEH and Greedy Randomized Adaptive search procedures to generate an initial population. The algorithm used insert, swap and variable neighborhood search functions to generate new solutions. The author used 120 problems from Taillard (1993) and 21 problems from Reeves (1995) datasets to compare their algorithm with various meta-heuristics. Chang et al. (2013) proposed an enhanced GA where a block mining method was used to locate common structures (blocks) from a set of high fitness chromosomes. The blocks were newly updated by high fit chromosomes. The author used 12 problems from Taillard (1993), 21 problems from Reeves (1995) and 8 problems from Carlier (1978) datasets to compare their algorithm with various meta-heuristics.

2.1 Scheduling Using Data Mining Techniques

In order to solve scheduling problems, there are many approaches from operations management and intelligent techniques. Typically, their performance depends on the situation of the system: no single rule exists to satisfy all the conditions of scheduling problems. To overcome this difficulty, the best dispatching rule for each situation should be used. For each circumference, finest machine learning technique is widely used. Ayutg et al. (1994) presented different machine learning methods to solve scheduling problems. To ascertain all the aspects of system, Data Mining (DM) is helpful. DM consists of Association, Classification, Clustering, and Regression. The DM techniques for classification are i) Bayesian, ii) Decision Tree (DT), iii) Support Vector Machines (SVM), and iv) ANN. The black-box patterns generated by Bayesian, ANN and SVM techniques are difficult to understand. In contrast, DTs are easily understandable and intuitively interpretable due to their graphical representation (Han & Gamber 2006). The aim of the DT is to find the structural information available in the dataset. The decision rules are formed based on DT algorithms and these rules are easily understandable. In the 1990s, DM algorithms attracted the manufacturing society. Harding et al. (2006) reviewed various applications of the data mining approach in the production sector. Table 3 includes some researchers' contributions to the area of DM techniques in manufacturing sectors.

It is evident from the literature that the application of data mining algorithms and a scatter search algorithm for PFSP receive less consideration from the researchers. It is also shown that most researchers used an initial seed solution from the NEH algorithm for their meta-heuristics. The advantages of both DT and SS algorithms are considered and a hybrid approach is proposed in this work. The main contributions of this proposal are summarized as follows:

1) To convert the given dataset into high level knowledge.

2) To generate a seed solution from the heuristics based method developed in Step 1.
3) To find the best solution, the seed solution produced from the DT based method acts as initial population to the SS algorithm.
4) To explore the structural information contained in the flowshop scheduling dataset.

Table 3 Summary and contributions of data mining techniques in production scheduling

Author	*Year*	*Algorithm*	*Remarks*
Li & Olafsson	2005	DT	Presented decision tree based algorithm for discovering dispatching rules from production data of single machine scheduling. In their work, authors used the decision tree as a dispatching rule instead of rules like Earliest Due Date [EDD], Earliest Release Date [ERD] etc.
Wang	2007	-	This paper discussed the implementation of DM techniques in manufacturing sectors.
Shukla et al.	2008	DT + C-fuzzy	Scheduling and Planning are hybridized with the help of multi-agent systems. DT is formed using C-fuzzy.
Kumar & Rao	2009	ACO + DM	Rules are framed for scheduling problem by Data Mining (DM) and recommended that scheduling can be done using DM instead of the ACO algorithm.
Li & Olafsson	2010	GA+DT	The single machine problem is solved by a hybrid approach to reduce the maximum lateness.
Choi et al.	2011	DT	Parallel machine problems are presented and solved by decision tree based approach.
AtifShahzad & Nasser Merbark	2012	TS+DM	Job shop scheduling is solved by DM technique and the results are compared with tabu search algorithm. The DM demonstrated better results.
Balasundaram et al.	2014	DT	DT was proposed to solve optimization of minimizing makespan and total flow time with equal priority.
Balasundaram et al.	2015	DT	Total flow time of permutation flowshop problems was solved using DT.

3. Proposed Approach

In this work, the initial seed produced by the DT (first phase) is given as an initial population of the SS algorithm (second phase) to obtain the best solution. The flowchart of the proposed hybrid model is shown in Fig. 1.

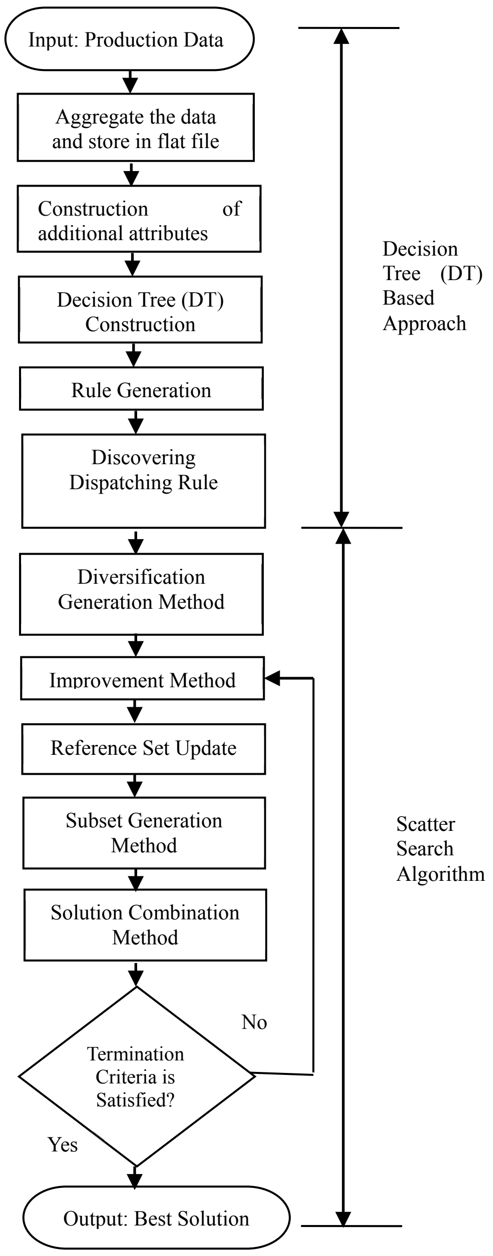

Input: Production Data
Aggregate the data and store in flat file
Construction of additional attributes
Decision Tree (DT) Construction
Rule Generation
Discovering Dispatching Rule
Diversification Generation Method
Improvement Method
Reference Set Update
Subset Generation Method
Solution Combination Method
Termination Criteria is Satisfied?
No
Yes
Output: Best Solution
Decision Tree (DT) Based Approach
Scatter Search Algorithm

Figure 1 Flow chart of proposed method

Data related to number of machines, jobs, and process times are to be entered in data module. The DT learns some target concept. For example, if two jobs are included, which one should be scheduled first? For makespan criterion, makespan is calculated for jobs with sequences 1-2 and 2-1. The job sequence having the minimum makespan is scheduled first. Likewise, the first job is compared to residual jobs and a decision is made whether or not to dispatch the first job. Similarly, the second job is compared to remaining jobs, and so on. Li & Olafsson (2010) represented that number of instances produced by n-jobs in this way:

$$\sum_{i=1}^{n-1} n-i.$$

Likewise, jobs are compared to each other and a flat file has been constructed. In order to get useful knowledge, various additional attributes are constructed and then the DT algorithm is applied to the flat file. From DT, decision rules are generated. These rules are applied to the dataset and each job is checked for its priority. If the accuracy of the jobs is 100%, it will occupy pass-1: otherwise, it is in pass-2 position. By combining two passes, the initial seed solution is generated. This initial seed is given as an input to SS algorithm, whose process is given below:

Step 1: Diversification Generation Method

The method develops a collection of initial populations (P_{size}), using a solution generated by DT. The initial seed generated from the DT consists of two passes. To develop different trial solutions, jobs are randomly swapped within their passes. For example, in a six-job problem, if jobs 1 & 3 occupy pass-1, the remaining jobs occupy pass-2. By randomly swapping jobs within their passes, different trial solutions are obtained. In this work, P_{size} is given as user input to the algorithm. Fig. 2 illustrates the result of obtaining a different seed solution as population.

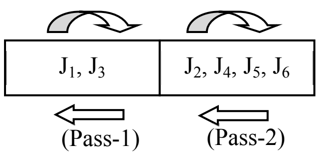


Figure 2 Illustration of obtaining a different initial seed solution

Step 2: Improvement Method

The populations generated in Step 1 are improved by this method, which provides an input in one or more trial solutions and tries to enhance them by generating one or more improved solutions. Each trial solution formed in Step 1 is improved by this method. In this work, similar to Step 1, jobs are swapped within their passes for a given number of times (via user input). For each swapping objective function value is checked. If there is no enhancement in the trial population, the improved solution is the same as that of the input solution.

Step 3: Reference Set Update Method

This step is achieved by selecting the '*best*' ones from all obtained solutions. It maintains b best solutions found in every iteration and it is constant parameter. The value of b (reference set) consists of b_1 (high quality solution) and b_2 (diverse solution). To find diverse solutions (b_2), *Euclidean* distances between b_1 and residual populations are considered. Specifically, this step looks for solutions that are not currently in $b_1$and *Euclidean* distances with maximum of minimum distances being selected as a b_2.

Step 4: Subset Generation Method

This method combines all solutions in the present reference set. Different subset collections are framed by combining two or more solutions from the reference set. They are:

Type I: all 2-element subsets. The different combinations of 2 element subsets are (1, 2), (1, 3), (1, 5),$\cdots$ etc.

Type II: all 3-element subsets. Eg. (1, 2, 3), (1, 2, 4),$\cdots$ etc.

Type III: all 4-element subsets. Eg. (1, 2, 3, 4), (1, 2, 3, 5),$\cdots$ etc.

Type IV: subsets consisting of 5 to b.

In this work Type I, i.e., all 2-element subsets, are taken and it consists of $(b^2-b)/2$ pair-wise solution combinations.

Step 5: Solution Combination Method

The Solution Combination method creates new solutions from each subset generated Step 4. Let there be two reference solutions, x^I and x^{II}, and various solutions are generated by combining these two reference sets. They are

C_1: $x^I - d$
C_2: $x^I + d$
C_3: $x^{II} - d$
C_4: $x^{II} + d$

Where $d = ran \times (x^{II} - x^I)/2$ and *ran* is integer between (0, 1).

Let us consider, b_1=2 (say P, Q) and b_2 = 2 (say X, Y). Then $b = b_1 + b_2 = 4$.

The number sub-set generated from b_1 and b_2 is 6. (ie.,$\{(b^2-b)/2 = (16-4)/2 = 6\}$).

Parameters	*Various 2-element sub-set combinations are*
b_1- (P, Q) b_2 - (X,Y)	(P,Q), (P,X), (P,Y), (Q,X), (Q,Y) and (X,Y)

P, Q represents two reference solutions. In the solution combination method, these two reference solutions will create 4 new solutions as explained above. Therefore, if b_1=2 and b_2 = 2, it will create 6 sub-sets, and each sub-set will create 4 trial solutions. Therefore, in total 24 trial solutions are generated at the end of the solution combination method. The distinguished feature of the scatter search algorithm, for search diversification and intensification, is different from other evolutionary methods and serves as an effective way to solve various optimization problems. The value of b_1 and b_2 is taken in such a way that the solutions produced in Step 5 should be higher than those of the initial population. The search continues until stopping criteria are met. By keeping the elements in pass-1 intact after the improvement method, search space is reduced.

3.1 Numerical Illustration of Decision tree construction with Scatter Search algorithm

Step 1: Input the dataset

Let us consider 5 jobs and 2 machine problem proposed by Baker (1974) which has been taken as a benchmark problem to test the proposed approach: this dataset has an optimal solution of 24 units of makespan. Table 4 shows the dataset of 5 jobs and 2 machines problem.

Table 4 A typical 5x2 flowshop problem proposed by Baker (1974)

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

Step 2: Basic attribute & class attribute construction

Tables 5 and 6 show the makespan of job sequence 1-2 and 2-1.

Table 5 Calculation of makespan for job sequence 1-2

Job	M_1		M_2	
	Input	Output	Input	Output
J_1	0	3	3	9
J_2	3	8	9	11

Table 6 Calculation of makespan for job sequence 2-1

Job	M_1		M_2	
	Input	Output	Input	Output
J_2	0	5	5	7
J_1	5	8	8	14

For the given job sequence 1-2*,* the makespan is 11 and it is 14 for the reverse case. Therefore, for any given sequence of two jobs, the minimum makespan is dispatched first. Likewise, pair-wise comparisons of outstanding jobs have to be done and a flat file has to be done. The total number of instances is 10 for the five job flowshop problem. The J_1P_{m1}, J_1P_{m2} etc., are named *predictor attributes,* and Job-1 first is named *class attribute*. The attribute J_iP_{mj} represents job j_i is processed into machine m_j. After constructing a flat file for a given dataset, the decision tree algorithm is applied. In other words, for each attribute, the information gained is calculated and the attribute that possesses the highest information gain is taken as an origin node. Table 7 shows the flat file for the given problem.

Table 7 Flat file for 5x2 flowshop problem

Instance No.	*Job-1*	J_1P_{m1}	J_1P_{m2}	Job-2	J_2P_{m1}	J_2P_{m2}	Job-1 scheduled First
1	1	3	6	2	5	2	yes
2	1	3	6	3	1	2	no
3	1	3	6	4	6	6	yes
4	1	3	6	5	7	5	yes
5	2	5	2	3	1	2	no
6	2	5	2	4	6	6	no
7	2	5	2	5	7	5	no
8	3	1	2	4	6	6	yes
9	3	1	2	5	7	5	yes
10	4	6	6	5	7	5	yes

Step 3: Decision tree construction with basic attributes

The information gain is defined as:

Entropy (P_1, P_2, and P_n) = $-P_1 \log_2^{P1} - P_2 \log_2^{P2} - \cdots P_n \log_2^{Pn}$

In general:

Info ($[C_1, C_2, \cdots, C_n]$) = entropy ($P_1, P_2, \cdots, P_n$)

Iteration 1:

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

No. of Yes – 6 & No. of No - 4

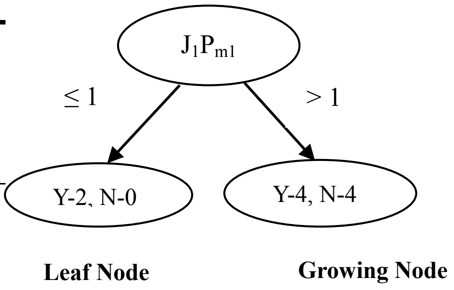
Information gain [6,4] = $-6/10 \log_2^{(6/10)} - 4/10 \log_2^{(4/10)}$ = 0.442179 + 0.528771 = 0.97095

Step 3(ii): Finding source node / root node

1) Estimation of information gain and tree construction for the attribute J_1P_{m1}

Split & Tree construction of attribute J_1P_{m1}

Processing time	*Yes*	*No*
3	3	1
5	0	3
1	2	0
6	1	0



1-indicates attributes split value

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

Info (4, 4) = $-4/8\log_2^{(4/8)} - 4/8\log_2^{(4/8)}$

= 0.5+0.5 = 1

Combined info{(2,0), (4,4)}

= $2/10 \times 0 + 8/10 \times 1$ = 0.8

A gain of attribute J_1P_{m1}

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

= 0.97095 – 0.8

= **0.1795**

To find a suitable attribute split value, the processing time values are set in ascending order to allow the information gain to be determined in both downward and upward directions by appending them one by one. The value that produces the highest information gain is chosen as a split value. In J_1P_{m1}, the processing unit 1 gives maximum information gain. Similarly, information gains are calculated for residual attributes and are shown in Table 8.

Table 8 Information gain of different attributes of the flat file shown in Table 7

Attribute	*Gain*
J_1P_{m1}	0.17951 (**max. Value**)
J_1P_{m2}	0.124511
J_2P_{m1}	0.046439
J_2P_{m2}	0.005802

The attribute J_1P_{m1} having highest information gain is selected as a source node. The decision rule derived from the J_1P_{m1} is as follows:

If $J_1P_{m1} \leq 1$ Job-1 scheduled first -- Rule (1)

The above rule correctly classifies instance numbers 8 & 9 in Table 7. Whereas the right branch of attribute J_1P_{m1} has a number of Yes-4 and No-4, it can be split further, so it is called a growing node. For the remaining eight instances, once again information gain is calculated and the attribute that possesses the maximum gain is selected as the sub-node. The process is repeated until all instances are classified from the flat file.

The resultant DT is shown in Fig. 3 and its size is 5 with 90% accuracy.

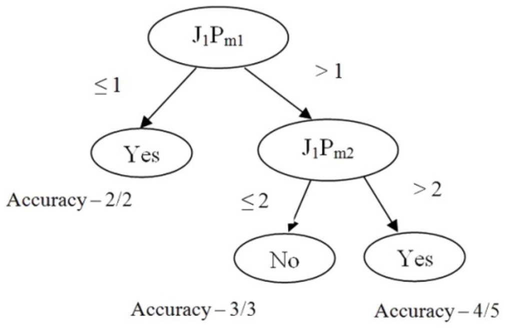


Figure 3 Decision tree for 5X2 flowshop problem with basic attributes

Table 9 Flat file for 5x2 problem proposed by Baker (1974)

Instance No.	*Job-1*	J_1P_{m1}	J_1P_{m2}	TP_1	*Job-2*	J_2P_{m1}	J_2P_{m2}	TP_2	P_{m1diff}	P_{m2diff}	T_{diff}	*Job-1 scheduled First*
1	1	3	6	9	2	5	2	7	−2	4	2	yes
2	1	3	6	9	3	1	2	3	2	4	6	no
3	1	3	6	9	4	6	6	12	−3	0	−3	yes
4	1	3	6	9	5	7	5	12	−4	1	−3	yes
5	2	5	2	7	3	1	2	3	4	0	4	no
6	2	5	2	7	4	6	6	12	−1	−4	−5	no
7	2	5	2	7	5	7	5	12	−2	−3	−5	no
8	3	1	2	3	4	6	6	12	−5	−4	−9	yes
9	3	1	2	3	5	7	5	12	−6	−3	−9	yes
10	4	6	6	12	5	7	5	12	−1	1	0	yes

Step 4: Decision tree construction with additional attributes

To obtain a more noteworthy DT, an improved data file is constructed. To demonstrate the potential benefit, various additional attributes are added that are helpful to reduce the tree size. The attributes such as

i) TP_1- total processing time of Job1 ($J_1P_{m1}+J_1P_{m2}$)

ii) TP_2-total processing time of Job2 ($J_2P_{m1}+J_2P_{m2}$)

iii) P_{m1diff} - processing time difference of $J_1P_{m1}-J_2P_{m1}$

iv) P_{m2diff} - processing time difference of $J_1P_{m2}-J_2P_{m2}$

v) $T_{diff} = TP_1 - TP_2$

are added to the flat file. Table 9 shows the resulting flat file for the given problem and Table 10 shows the information gain of all attributes.

Table 10 Information gain of various attributes for the flat file shown in Table 9

Attribute Name	*Information gain*
J_1P_{m1}	0.17951
J_1P_{m2}	0.124511
TP_1	0.124511
J_2P_{m1}	0.046439
J_2P_{m2}	0.005802
TP_2	0.321928 (**max.value**)
P_{m1diff}	0.144484
P_{m2diff}	0.000745
T_{diff}	0.019973

The attribute TP_2 demonstrates the maximum information gain and it is selected as a source node. Fig. 4 shows a tree structure of attribute TP_2.

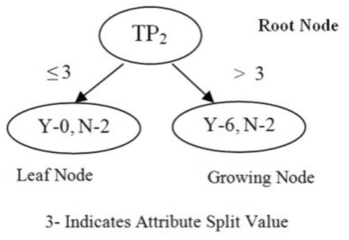


Figure 4 Decision tree constructed from the attribute TP_2

In the attribute TP_2, processing time 3 is selected as attribute split value, as it reflects the highest information gain as compared to the residual processing time. The decision rule derived from the above tree is

If $TP_2 \leq 3$, Job-2 scheduled first

The left branch of root node correctly classifies instance numbers 2 & 5 in Table-7, but the right branch can be split further. The remaining instance information gain is calculated and the resulting decision tree is shown in Fig. 5.

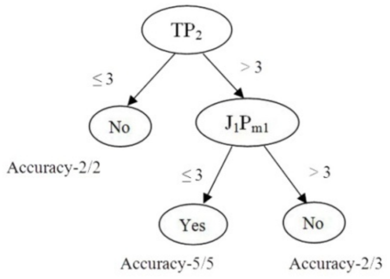


Figure 5 Decision tree for 5x2 flowshop problem with additional attributes

The size of the above tree is 5. The decision rules derived from the above tree are as follows:

If $TP_2 \leq 3$, Job-2 scheduled first -- Rule (1)

If $TP_2 > 3$, and $J_1P_{m1} \leq 3$ Job-1 scheduled first; otherwise, Job-2 -- Rule (2)

Except for instance number 10, the remaining instances are correctly classified by the rules. For this particular dataset, size and tree accuracy is the same before and after data engineering.

Step 5: Evaluation of rules and framing initial seed solution

To discover the dispatching sequence, the decision rules are applied to the dataset. Based on rules each job is checked for its priority. If the jobs satisfy all the conditions of the decision rules, they will be placed in pass-1 position: otherwise, they are placed in pass-2 position. Initial sequence is obtained by combining these two passes. Table 11 shows accuracy of each job of the sample problem.

Table 11 Accuracy of jobs for 5 × 2 flowshop problem

Job	*Accuracy (%)*
J_1	75
J_2	0
J_3	100
J_4	0
J_5	0

In the above example problem Job-3 is placed in pass-1 location and the residual jobs are placed in pass-2. The initial sequence is shown in Fig. 6.

Initial Sequence:

Pass - 1	Pass - 2
J_3	J_1, J_2, J_4, J_5

Figure 6 Framing of initial sequence

Step 6: Application of scatter search algorithm

The initial sequence produced by the DT is given as input to SS algorithm with population size of 10. The problem is tested with 5 trial runs. To compare the proposed methodology, Johnson's algorithm was applied to the above dataset. Gen & Cheng (1997) developed genetic algorithms for the above dataset with a *population size of 20* and the *number of iterations as 20*. The final results are shown in Table 12. For all trial runs, the proposed approach yielded optimal makespan value.

Table 12 Comparison of results of decision tree based method with literature

Sl.No.	*Method*	*Dispatching Sequence*	*Makespan*
1	Proposed Work	3-1-4-5-2	24
2	Genetic Algorithm (GA)	3-1-4-5-2 / 1-4-3-5-2 / 1-3-4-5-2	24
3	Johnson's Algorithm	3-1-4-5-2	24

3.1.1 Numerical Example: 2

Table 13 shows a 5 × 4 flowshop problem proposed by Ho & Chang (1991) which has been taken as a benchmark to test the proposed approach.

Table 13 A 5 × 4 flowshop problem developed by Ho & Chang (1991)

Machine & Job	J_1	J_2	J_3	J_4	J_5
M_1	31	19	23	13	33
M_2	41	55	42	22	5
M_3	25	3	27	14	57
M_4	30	34	6	13	19

The above dataset is given as input to the proposed approach and tested proposed method with population size of 10. Gen & Cheng (1997) developed genetic algorithms for the above dataset with *population size =20, maximum generation =150, Cross over probability =0.3, mutation probability =0.1* and compared it with various well-known heuristics. The results are reported in Table 14.

Table 14 Comparison of results of the proposed method with GA & various heuristics

Sl.No.	*Method*	*Dispatching Sequence*	*Makespan*
1	Proposed Method	4-2-5-1-3	213
2	Genetic Algorithm (GA)	4-2-5-1-3	213
3	Ho & Chang	4-2-5-1-3	213
4	CDs	4-2-1-5-3	246
5	Gupta	2-1-5-3-4	251
6	Palmer	5-2-4-1-3	245
7	Random	1-2-3-4-5	286

For all 5 trial runs, the proposed method yields an optimal makespan of 213 and offers evidence that the proposed hybrid method is better than other methods.

4. Experimental Results

To estimate the performance of the hybrid approach, it is compared to other algorithms using diverse datasets of the PFSP. In this work, sixty problems from various datasets are taken for experiments. The first eight problems are proposed by Carlier (1978), the next twelve problems are proposed by Reeves (1995), and the last 40 problems (Ta01- Ta40) are given by Taillard (1993). The proposed hybrid approach is coded in JAVA and run on Intel Core 2, 1.19-GHz with 952 MB RAM. The parameter settings are given in Table 15. All trials are carried out with 20 independent runs and completed after Jobs (n) × Machines (m) generations without enhancement in their value.

Table 15 Parameter settings of proposed algorithm

Population size	*100*
b_1	10 percentage of population size
b_2	10 percentage of population size
Local Search	5
The number of iterations	$n \times m$ (n-number of jobs, m-number of machines)
Elitism rate	0.01

The average Error Ratio (ER_i) is used to estimate the performance of the algorithm and it is determined by the following equation:

$$ER_i = \frac{C_{\max}(X) - U}{U} * 100\%,$$

where $C_{\max}(X)$ is the makespan of various algorithms and U is best known or optimal value available in the literature.

4.1 The Carlier datasets of PFSP

Eight well-known benchmark datasets proposed by Carlier (1978) are used for experiments. The performance of the proposed work for Carlier (1978) datasets are compared with p-ACGA (Chang et al. 2013), best hybridized version of SA (Rajkumar & Shahabudeen 2008), PSOMA (Liu et al. 2007) and HGA (Wang & Zheng 2003). Table 16 shows the test results of proposed method along with computational time, and Table 17 shows error ratio values for all eight datasets along with literature results.

From Table 17, except for the Car-6 problem, p-ACGA and proposed method yielded optimal value for 7 problems. For the Car-6 problem, the error ratio of the proposed approach is 2.5% higher than p-ACGA approach. The p-ACGA approach consists of GA, local search, heuristics and enhanced recombination operator. The proposed approach yielded much better solution quality compared to various hybrid algorithms, meta-heuristics and heuristics for Carlier (1978) datasets.

Table 16 Results of proposed method for Carlier (1978) datasets

Sl.No.	*Problem Name*	*Problem Size (Jobs × Machines)*	*Optimal Value available in the Literature*	*Results of the Proposed Method*	*Error Ratio*	*Number of Iterations*	*Computational Time (milli seconds)*
1	Car-1	11 × 5	7038	7038	0	56	323
2	Car-2	13 × 4	7166	7166	0	53	349
3	Car-3	12 × 5	7312	7312	0	79	474
4	Car-4	14 × 4	8003	8003	0	80	501
5	Car-5	10 × 6	7720	7720	0	67	343
6	Car-6	8 × 9	8505	8507	0.0235	78	390
7	Car -7	7 × 7	6590	6590	0	50	198
8	Car-8	8 × 8	8366	8366	0	88	388
				Average		55.1	296.6

Table 17 Average error ratio of algorithms for Carlier datasets

Problem	*n × m*	*PM*	*P-ACGA*	*SA+PS*	*PSOMA*	*HGA*
Car-1	11 × 5	0	0	0	0	0
Car-2	13 × 4	0	0	0	0	0
Car-3	12 × 5	0	0	0	0	0
Car-4	14 × 4	0	0	0	0	0
Car-5	10 × 6	0	0	0	0.018	0
Car-6	8 × 9	0.0235	0	0	0.114	0.04
Car-7	7 × 7	0	0	0	0	0
Car-8	8 × 8	0	0	0.65	0.02	0.01
Average		0.0029	0	0.0812	0.019	0.006

PM- Proposed Method

4.2 The Reeves Datasets of PFSP

Twelve benchmark datasets are taken for experiments proposed by Reeves (1995). The performance of proposed hybrid approach is compared with p-ACGA (Chang et al. 2013), HDBAC (Liu & Liu 2013), ACGA (Chen et al. 2009), best hybridized version of SA (Rajkumar & Shahabudeen 2008), PSOMA (Liu et al. 2007), PSO_{VNS} (Tasgetiren et al. 2004) and HGA (Wang

& Zheng 2003).Table 18 shows the test results of proposed method and Table 19 shows a comparison with literature on all 12 datasets.

Table 19 shows that the proposed methodology yields a lowest average error ratio more often than the other hybrid algorithms and meta-heuristics for these 12 datasets of *Reeves* (1995). Of the 12 problems, the proposed method yields optimal value for 7 problems and a lower error ratio for 4 problems. For Rec-05 problem, algorithm HDBAC yielded the lowest error ratio. The HDBAC includes NEH, GRASP, VNS, and ACO.

Table 18 Results of proposed method for Reeves (1995) datasets

Sl.No.	*Problem Name*	*Problem Size (Jobs × Machines)*	*Optimal Value available in the Literature*	*Results of the Proposed Method*	*Error Ratio*	*Number of Iterations*	*Computational Time (milli seconds)*
1	Rec-01	20× 5	1247	1247	0	362	5031
2	Rec-03	20× 5	1109	1109	0	304	4581
3	Rec-05	20× 5	1242	1245	0.2415	230	3761
4	Rec-07	20× 10	1566	1566	0	388	5476
5	Rec-09	20× 10	1537	1537	0	447	9839
6	Rec-11	20× 10	1431	1431	0	418	7321
7	Rec-13	20× 15	1930	1930	0	894	14129
8	Rec-15	20× 15	1950	1951	0.0512	675	13124
9	Rec-17	20× 15	1902	1907	0.2628	846	19640
10	Rec-19	30× 10	2093	2093	0	811	21931
11	Rec-21	30× 10	2017	2046	1.4377	1098	33943
12	Rec-23	30× 10	2011	2021	0.4972	1248	39229

Table 19 Average error ratio of algorithms for Reeves datasets

Problem	*$n \times m$*	*PM*	*P-ACGA*	*HDBAC*	*ACGA*	*SA+ CRIPS+PS*	*PSOMA*	*PSO_{VNS}*	*HGA*
Rec-01	20× 5	**0**	0.11	0.128	0.16	0.2	0.14	0.168	0.14
Rec-03	20× 5	**0**	0.07	0.041	0.09	0.04	0.19	0.158	0.09
Rec-05	20× 5	0.2415	0.28	**0.145**	0.24	0.33	0.25	0.249	0.29
Rec-07	20× 10	**0**	0.46	0.934	0.51	1.48	0.99	1.095	0.69
Rec-09	20× 10	**0**	0.07	0.059	1.11	0.28	0.62	0.651	0.64
Rec-11	20× 10	**0**	0.08	0.073	0.14	1.34	0.13	1.153	1.1
Rec-13	20× 15	**0**	0.37	0.474	0.83	1.04	0.89	1.79	1.68

Problem	$n \times m$	*PM*	*P-ACGA*	*HDBAC*	*ACGA*	*SA+ CRIPS+PS*	*PSOMA*	*PSO$_{VNS}$*	*HGA*
Rec-15	20×15	**0.0512**	0.53	0.956	0.72	0.79	0.63	1.487	1.12
Rec-17	20×15	**0.2628**	1.05	1.669	1.58	1.43	1.33	2.453	2.32
Rec-19	30×10	**0**	0.92	1.247	1.49	1.1	1.31	2.099	1.32
Rec-21	30×10	**1.4377**	1.49	1.448	1.64	1.6	1.6	1.671	1.57
Rec-23	30×10	**0.4972**	1.05	1.235	1.79	1.23	1.31	2.106	0.87
Average		**0.2075**	0.54	0.7007	0.858	0.905	0.782	1.256	0.985

PM- *Proposed Method***, Bold** *letter indicates the minimum value obtained from various methods*

4.2.1 Comparison of Computational Time

Liu & Liu (2013) indicated maximum elapsed time for the flowshop scheduling problem is $(n \times m)/10$ which results in practical computational time. The average computational time in seconds of the proposed method is compared with literature and depicted in the Table 20.

Table 20 Comparison of average CPU time in seconds

Problem Name	*Problem Size Jobs × Machines ($n \times m$)*	*Maximum Elapsed Time in Seconds*	*Proposed Method*	*HDBAC (Liu & Liu 2013)*
Rec-01	20×5	10	5.031	0.05
Rec-03	20×5	10	4.581	0.8
Rec-05	20×5	10	3.761	0.45
Rec-07	20×10	20	5.476	0.3
Rec-09	20×10	20	9.839	0.75
Rec-11	20×10	20	7.321	0.65
Rec-13	20×15	30	14.129	1.5
Rec-15	20×15	30	13.124	1.5
Rec-17	20×15	30	19.64	1.5
Rec-19	30×10	30	21.931	3.85
Rec-21	30×10	30	33.943	2.05
Rec-23	30×10	30	39.229	4.4

Although in the proposed method, computational time is higher than that of the HDBAC algorithm, it is well below the maximum elapsed time in most of the cases. Among the 12 problems, the proposed method yields an optimal value for 7 problems and a much lower error ratio except for the Rec-05 problem. The statistical test of significance between the proposed method and HDBAC proves that the proposed approach is appreciably better than the HDBAC algorithm.

4.3 The Taillard Datasets of PFSP

In this section, 40 well-known benchmark instances given by Taillard (1993) were experimented. For comparing algorithm efficiency, most literature resources used an upper bound and a few authors used a lower bound of Taillard (1993) to test dataset instances. Because most of the literature used upper bound, the proposed approach also uses upper bound as optimal value and compares it with the other algorithms from the literature such as the best hybridized version of Genetic Algorithm-MCGA (Chen et al. 2012), NEH+SA+C (Laha & Chakraborty 2009), NEH+SS (Saravanan et al. 2008), H-CPSO (Jabouri et al. 2008), DDE (Pan et al. 2008), PSO_{spv} (Tasgetiren et al. 2007), PACO (Rajendran & Ziegler 2004) and ACS (Ying & Liao 2004).

Table 21 shows the test results of the proposed method along with computational time and Table 22 shows average error ratio of various literatures on all 40 datasets. From Table 22, it is seen that for all 10 problems of 20×5 dataset, the proposed methodology yields an optimal value and much lower average error ratio for 20×10 and 20×20 datasets. In the 50×5 dataset hybrid algorithm of NEH, SA and composite heuristics produced lower values.

Table 21 Results of proposed method for Taillard (1993) datasets

Sl.No.	*Problem Name*	*Problem Size (Jobs × Machines)*	*Optimal Value available in the Literature*	*Results of the Proposed Method*	*Error Ratio*	*Number of Iterations*	*Computational Time (Milli Seconds)*
1	Ta01	20× 5	1278	1278	0	178	1862
2	Ta02		1359	1359	0	167	1725
3	Ta03		1081	1081	0	192	2609
4	Ta04		1293	1293	0	279	4664
5	Ta05		1235	1235	0	184	1909
6	Ta06		1195	1195	0	190	3362
7	Ta07		1234	1234	0	186	2903
8	Ta08		1206	1206	0	299	2737
9	Ta09		1230	1230	0	205	2893
10	Ta10		1108	1108	0	196	2476
		Average			**0**	207.6	2714
11	Ta11	20× 10	1582	1583	0.0632	454	5003
12	Ta12		1659	1664	0.3013	548	10964
13	Ta13		1496	1496	0	624	9094
14	Ta14		1377	1379	0.1452	584	6765
15	Ta15		1419	1419	0	605	9325
16	Ta16		1397	1397	0	496	6668
17	Ta17		1484	1484	0	514	10603
18	Ta18		1538	1545	0.4551	541	10221

19	Ta19		1593	1594	0.0627	536	10507
20	Ta20		1591	1591	0	615	12791
		Average			**0.1027**	551.7	9194.1
21	Ta21	20 × 20	2297	2307	0.4353	937	19886
22	Ta22		2099	2115	0.7622	1297	26180
23	Ta23		2326	2326	0	1074	22174
24	Ta24		2223	2232	0.4048	991	26528
25	Ta25		2291	2307	0.6983	778	12549
26	Ta26		2226	2233	0.3144	1328	21878
27	Ta27		2273	2282	0.3959	997	24384
28	Ta28		2200	2200	0	952	14291
29	Ta29		2237	2237	0	852	15777
30	Ta30		2178	2181	0.1377	1314	27636
		Average			**0.2791**	1052	21128.3
31	Ta31	50 × 5	2724	2724	0	421	24412
32	Ta32		2834	2838	0.1411	634	34074
33	Ta33		2621	2621	0	757	34149
34	Ta34		2751	2753	0.0727	668	27885
35	Ta35		2863	2864	0.0349	671	29973
36	Ta36		2829	2832	0.1060	561	28250
37	Ta37		2725	2725	0	646	37571
38	Ta38		2683	2683	0	686	34235
39	Ta39		2552	2555	0.1175	562	29891
40	Ta40		2782	2782	0	628	35981
		Average			**0.0472**	623.4	31642.1

Table 22 Average error ratio of algorithms for Taillard datasets

Problem	*n* × *m*	*PM*	*MCGA*	*NEH+SA+C*	*NEH+SS*	*H-CPSO*	*DDE*	*PSO (SPV)*	*PACO*	*ACS*
Ta01-Ta10	20 × 5	**0**	0.81	0.91	0.33	1.05	0.46	1.75	0.704	1.14
Ta11-Ta20	20 × 10	**0.103**	1.4	0.665	0.88	2.42	0.93	3.25	0.843	1.7
Ta21-Ta30	20 × 20	**0.279**	1.06	0.459	0.497	1.99	0.79	2.82	0.72	1.6
Ta31-Ta40	50 × 5	0.047	0.44	**0.025**	0.166	0.99	0.17	1.14	0.09	0.43
Average		**0.107**	0.927	0.335	0.468	1.61	0.58	2.24	0.589	1.23

PM-Proposed Method

4.4 Statistical Test

The Statistical tests were performed on all datasets. For Carlier (1978) and Reeves (1995) datasets, the error ratio for all the instances of the problem were specified in the literature and it is decided to conduct a *t-test* for these two datasets. In the Taillard (1995) dataset, most authors specified an error ratio of population means of dataset. For example Ta01-Ta10 consists of 10 problems, so most authors would utilize an average error ratio of 10 problems: they would not specify error ratio of individual problems. Accordingly, we decided to conduct a *paired t-test* for Taillard (1993) dataset.

The test results are reported in Tables 23-25. From Table 23, the Carlier datasets, p-ACGA is more statistically significant than the other methods: it produces an optimal value for all 8 problems, whereas the proposed method yields optimal value for 7 problems only.

For Carlier problem sets PSOMA & HGA gives statistically the same average error ratio. From Table 24 and 25, it is clear that for the *Reeves* and Taillard datasets, proposed method produces better solutions as compared to existing literature.

Table 23 Statistical test for Carlier datasets

	P-ACGA	*SA+PS*	*PSOMA*	*HGA*
Mean	−0.0029	0.07831	0.01606	0.00331
Std. deviation	0.00777	0.2162	0.02926	0.00596
t-test value	−1.0686	**1.0244**	**1.55249**	**1.5708**
Tested problem-8,				
The table value is1.895 at 0.05level.				

Table 24 Statistical test for Reeves datasets

	P-ACGA	*HDBAC*	*ACGA*	*SA+CRPIS+PS*	*PSOMA*	*PSOVNS*	*HGA*
Mean	0.3324	0.4932	0.6508	0.6974	0.5749	1.0491	0.7783
Std.deviation	0.2963	0.509	0.521	0.5033	0.4223	0.7587	0.6424
t-test value	**3.88667**	**3.356**	**4.3209**	**4.799**	**4.7162**	**4.789**	**4.1968**
Tested Problem-12							
The table value is 1.796 at 0.05level.							

Table 25 Statistical test for Taillard datasets

	MCGA	*NEH+SA+C*	*NEH +SS*	*H-CPSO*	*DDE*	*PSO (SPV)*	*PACO*	*ACS*
Mean	0.820	0.407	0.361	1.505	0.481	2.132	0.482	1.110
Std. deviation	0.320	0.358	0.251	0.553	0.249	0.777	0.278	0.450
t-test value	**5.113**	**2.276**	**2.869**	**5.439**	**3.846**	**5.480**	**3.460**	**4.931**
Number of mean values N-4								
The table value is 2.353 at 0.05level.								

Bold letters indicate the proposed method solutions are statistically significant

Table 26 Size & accuracy of the tree for Carlier and Reeves datasets

Sl.no.	*Data set Name*	*Dataset Size*	*Total number of Instances*	*Tree Size*		*Accuracy of Tree*	
				Before Data Engineering	*After Data Engineering*	*Before Data Engineering*	*After Data Engineering*
1	Car-1	11 × 5	55	13	13	92.727	94.545
2	Car-2	13 × 4	78	19	15	97.435	97.435
3	Car-3	12 × 5	66	17	17	95.454	98.484
4	Car-4	14 × 4	91	21	17	90.109	91.208
5	Car-5	10 × 6	45	11	11	100	100
6	Car-6	8 × 9	28	9	9	92.857	92.875
7	Car-7	7 × 7	21	7	7	95.238	95.238
8	Car-8	8 × 8	28	7	5	96.428	96.428
	Average		**51.5**	**13**	**11.75**	**95.031**	**95.776**
9	Rec-01	20 × 5		47	23	94.736	93.684
10	Rec-03	20 × 5		41	39	93.157	97.368
11	Rec-05	20 × 5		39	33	95.263	89.473
12	Rec-07	20 × 10		35	35	94.736	98.421
13	Rec-09	20 × 10	190	39	39	91.578	96.842
14	Rec-11	20 × 10		37	35	94.736	96.842
15	Rec-13	20 × 15		45	39	95.789	95.789
16	Rec-15	20 × 15		37	37	92.105	93.6842
17	Rec-17	20 × 15		47	41	91.578	96.842
		Average		**40.777**	**35.669**	**93.742**	**95.438**
18	Rec-19	30 × 10		79	43	92.873	86.666
19	Rec-21	30 × 10	435	87	39	86.896	80.229
20	Rec-23	30 × 10		81	77	92.183	95.402
		Average		**82.333**	**53**	**90.650**	**87.432**

Table 27 shows the average value of *Taillard* datasets, total number of instances, and the minimum, maximum and average size/accuracy.

Table 27 Tree size and accuracy for Taillard datasets

Data Set Size	*Total No. of Instances*	*Tree Size*						*Accuracy*					
		BDE			ADE			BDE			ADE		
		Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
20× 5	190	29	47	36.2	27	39	31.4	89.4	96.3	93.36	81.5	98.42	94.2
20× 10	190	33	51	43	33	49	41.4	90	96.8	93.47	91.0	97.36	95.1
20× 20	190	37	55	44	27	41	35	86.8	95.2	91.63	85.2	97.3	93.8
50× 5	1225	115	139	125	39	139	84.8	81.4	89.9	84.03	77.0	89.3	82.3

***BDE** -Before Date Engineering, **ADE-** After Data Engineering, **Min-**Minimum, **Max-**Maximum, **Avg-** Average*

4.5 Supplementary Attribute Construction:

The scheduling decision plays major role in manufacturing sector and important decisions are to be built by using data engineering. In most cases, it is done manually using an intuitive process. DTs are constructed both before and after data engineering. Table 26 shows the total number of instances, size, and accuracy of the datasets for Carlier and Reeves.

Tables 26 and 27 demonstrate that, for small size problems, there is a significant reduction in tree size with a slight increase in accuracy. But when job size increases to more than 30, there is a small reduction in accuracy with significant tree size reduction.

5. Conclusions and Future Work

The hybrid approach of combining DT and SS algorithms is presented to solve the permutation flowshop problem, a well-known combinatorial optimization problem. Unlike existing hybrid algorithms in literature, the proposed approach uses only two algorithms. To compare the proposed work, sixty problems are analyzed from various well-known standard datasets. The proposed method yields optimal value for 37 problems and produces the lowest error ratio for the most problems. Simulation results and statistical test comparisons demonstrate the advantages of the proposed algorithm in-terms of solutions. The *If-Then else* rules are used to form the Decision Tree and it is easily understandable by process planners. The SS has various unifying principles to find a new solution which avoids generating duplicate solutions. The Statistical results show that the proposed algorithm produces an enhanced solution superior to the existing methods. In real time situations, the better dispatching rules are framed from the conditions of shop floor system. The DT based approach constructs the tree from 1,225 instances for 50 job problems and the average tree size is 84.8 after data engineering. Various instance selection methods can be implemented in future to reduce the tree size with good solution accuracy.

Appendix 1: Paired *t*-test

A paired *t-test* compares two population means, where observations in one sample can be paired with observations in the other. Let us consider the population means of MCGA (Chen et al. 2012) and the proposed method. Table 28 shows the population means of various problem instances of Taillard.

Table 28 The comparison between MCGA and proposed method

Problem Name	$n \times m$	*Mean values of population*	
		MCGA	*Proposed Method*
Ta01 - Ta10	20×5	0.81	0
Ta11 - Ta20	20×10	1.4	0.1027
Ta21 - Ta30	20×20	1.06	0.2791
Ta31- Ta40	50×5	0.44	0.0472

The above mean values are called pre-module score and post module score.

Pre-module score	*Post-module score*	*Difference (d)*	$(d^2 - \bar{d}^2)$
0.81	0	0.81	−0.01679
1.4	0.1027	1.2973	1.01009
1.06	0.2791	0.7809	−0.063087
0.44	0.0472	0.3928	−0.518600
		($\bar{d}$)0.8203	Mean- 0.10294

i) Mean $\bar{d} = 0.8203$

ii) Standard deviation

$$S_d = \sqrt{Meanof(d^2 - \bar{d}^2)} = 0.3207$$

iii) $SE(\bar{d}) = \dfrac{S_d}{\sqrt{n}} = 0.3209/2 = 0.1604$

(n-number of mean sample-4)

iv) $t = \dfrac{\bar{d}}{SE(\bar{d})} = 0.8203/0.1604 = 5.113$

From the *t*-distribution table, c-value is 2.353 at 0.05level. The value of *t* is greater than 2.353. Hence the proposed method is statistically significant and better than the MCGA method.

Acknowledgements

The authors are thankful to the referees for their suggestions and comments to improve the earlier version of the paper.

References

[1] Agarwal, A., Colak, S. & Eryarsoy, E. (2006). Improvement heuristic for the flow-shop scheduling problem: an adaptive-learning approach. European Journal of Operational Research, 169(3): 801-815.

[2] Atif, S. & Nasser, M. (2012). Data mining based job dispatching using hybrid simulation-optimization approach for shop scheduling problem. Journal of Engineering Applications of Artificial Intelligence, 25(6): 1173-1181.

[3] Aytug, H., Bhattacharyya, S., Koehler, G.J. & Snowdon, J.L. (1994). A review of machine learning in scheduling. IEEE Transactions on Engineering Management, 41(2):165-171.

[4] Baker, K.R. (1974). Introduction to Sequencing and Scheduling. New York: John Wiley & Sons, Inc.

[5] Balasundaram, R., Valavan, D. & Baskar, N. (2014). Heuristic based approach for bi-criteria optimization of minimizing makespan and total flow time of flowshop scheduling. International Journal of Mechanical & Mechatronics Engineering, 14(02):1-11.

[6] Balasundaram, R., Kannan, G., Baskar, N. & Shiva, S.R. (2015). Rule based heuristic based approach for minimizing total flow time in permutation flowshop scheduling. Tehnički vjesnik, 22 (01) :25-32.

[7] Carlier, J. (1978). Ordonnancements a contraintes disjonctives. R.A.I.R.O. Recherche Operationelle/Operations Research, 12 (4): 333-350.

[8] Gen, M. & Cheng, R. (1997) Genetic Algorithms and Engineering Design, New York: John Wiley & Sons, Inc.

[9] Chen, S., Chang, P.C. & Zhang, Q. (2009). A self-guided genetic algorithm for flowshop scheduling problems. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, pp. 471-478.

[10] Chen, S., Chang, P.C., Cheng, TCE. & Zhang, Q. (2012). A self-guided genetic algorithm for permutation flowshop scheduling problems. Computers & Operations Research, 39(7): 1450-1457.

[11] Chang, P.C., Hunag, W.H., Wu, J.L. & Cheng, TCE. (2013). A block mining and re-combination enhanced genetic algorithm for the permutation flowshop scheduling problem. International Journal of Production Economics, 141(1): 45-55.

[12] Choi, H.S., Kim, J.S. & Lee, D.H. (2011). 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):3514-3521.

[13] Gupta, J.N.D. & Stafford, E.F. (2006). Flowshop scheduling research after five decades. European Journal of Operational Research, 169(3): 699-711.

[14] Han, J. & Kamber, M. (2006). Data Mining: Concepts and Techniques, San Francisco: Morgan Kaufmann.

[15] Ho, J.C. & Chang, Y.L. (1991). A new heuristic for the *n*-job, *m*-machine flow-shop problem. European Journal of Operational Research, 52 (2):194-202.

[16] Harding, J.A., Shahbaz, M. & Srinivas, K. A. (2006). Data mining in manufacturing: a review. Journal of Manufacturing Science and Engineering, 128(4): 969 -976.

[17] Jarboui, B., Ibrahim, S., Siarry, P. & Rebai, A. (2008). A combinatorial particle swarm optimization for solving permutation flowshop problems. Computers & Industrial Engineering, 54(3):526-538.

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

[19] Kalczynski, P.J. & Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. OMEGA, The International Journal of Management Science, 35(1): 53-60.

[20] Kumar, S. & Rao, CSP. (2009). Applications of ant colony, genetic algorithm and data mining based technique for scheduling. Robotics and Computer - Integrated Manufacturing, 25(6): 901-908.

[21] Laha, D. & Chakraborty, U.K. (2009). An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling. International Journal of Advanced Manufacturing Technology, 44(5):559-569.

[22] Li, X. & Olafsson, S. (2005). Discovering dispatching rules using data mining. Journal of Scheduling, 8(6):515-527.

[23] Li, X. & Olafsson, S. (2010). Learning effective new single machine dispatching rules from optimal scheduling data. International Journal of Production Economics, 128(1):118-126.

[24] Liu, B., Wang, L. & Jin, Y.H. (2007). An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Transactions on Systems, Man and Cybernetics, Part B, 37(1): 18-27.

[25] Liu, F.Y. & Liu, S.Y. (2013). A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. Applied Soft Computing, 13(3): 1459-1463

[26] Modrák, V. & Pandian, R.S. (2010). Flow shop scheduling algorithm to minimize completion time for *n*-jobs *m*-machines problem. Technical Gazette, 17(3): 273-278.

[27] Mircea, A. (2012). On solving flowshop scheduling problems. Proceeding of the Romanian Academy, Series A, 13(1):71-79.

[28] Nawaz, M., Enscore, Jr EE. & Ham, I. (1983) A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. OMEGA- The International Journal of Management Science, 11 (1): 91-95.

[29] Pan, Q., Tasgetiren, M. & Liang, Y. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers & Industrial Engineering, 55(4):795-816.

[30] Rad, S.F., Ruiz, R. & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permuation flow shops. OMEGA - The International Journal of Management Science, 37(2): 331-345.

[31] Rajendran, C. & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research, 155:426-438.

[32] Rajkumar, R. & Shahabudeen, P. (2008). Performance evaluation of simulated annealing algorithm for flowshop scheduling problems. The International Journal of Applied Management and Technology, 5(3):172-189.

[33] Reeves, C.R. (1995). A genetic algorithm for flow shop sequencing. Computers and Operations Research, 22 (1):5-13.

[34] Saravanan, M., Haq, N., Vivekraj, A.R. & Prasad, T. (2008). Performance evaluation of the scatter search method for permutation flowshop sequencing problems. International Journal of Advanced Manufacturing Technology, 37(11-12): 1200-1208.

[35] Shukla, S.K., Tiwari, M.K. & Son, Y.J. (2008). Bidding-based multi-agent system for integrated process planning and scheduling : a data-mining and hybrid tabu-SA algorithm oriented approach. International Journal of Advanced Manufacturing Technology, 38(1):163-175.

[36] Taillard, E. (1993). Benchmarks for basic scheduling instances. European Journal of Operational Research, 64(2): 278-285.

[37] Tasgetiren, M., Sevkli, M. & Liang, Y.C. (2004) Particle swarm optimization algorithm for permutation flowshop sequencing problem, in: M. Dorigo, et al. (eds.), Lecture Notes in Computer Science, vol. 3172, Springer-Verlag, New York, 2004, pp. 382-389.

[38] Tasgetiren, M., Liang, Y., Sevkli, M. & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flow shop sequencing problem. European Journal of Operational Research, 177(3):1930-1947.

[39] Wang, K. (2007). Applying data mining to manufacturing: the nature and implications. Journal of Intelligent Manufacturing, 18(4): 487-495.

[40] Wang, L. & Zheng, D.Z. (2003). An effective hybrid heuristic for flow shop scheduling. The International Journal of Advanced Manufacturing Technology, 21(1): 38-44.

[41] Ying, K.C & Liao, C.J. (2004). An ant colony system for permutation flow-shop sequencing. Computers & Operations Research, 31(5): 791-801.

Kannan Govindan works as a professor and Head of Center for Sustainable Engineering Operations Management, Department of Technology and Innovation, University of Southern Denmark. He received B.E. (mechanical engineering) from Madurai Kamarajar University, M.E. (industrial engineering) from Bharathiyar University and Ph.D (supply chain management) from National Institute of Technology, Tiruchirappalli, India. He has published more than 150 papers in international journals. Currently Editor-in-Chief for International Journal of Advanced Operations Management, International Journal of Business Performance and Supply Chain Modelling, Inderscience Publishers, Switzerland and subject editor – Sustainable Supply Chain Management in Journal of Cleaner Production. He is a visiting professor at UNESP, Dalian University of Technology and Dalian Maritime University. His current area of research includes logistics, supply chain management (SCM), green and sustainable SCM, CSR, EPR and e-waste.

R. Balasundaram works as a professor of mechanical engineering at K.Ramakrishnan College of Engineering, Tiruchirappalli, India. He received B.E (mechanical engineering) from Madras University, M.E. (engineering design) from Bharathiyar University and Ph.D (mechanical engineering) from Anna University. He is having more than 18 years of teaching experience and current area of research includes optimization techniques, production planning & control and data mining.

N. Baskar works as a professor of mechanical engineering at Saranathan College of Engineering, Tiruchirappalli, India. He received B.E in mechanical engineering from Bharathidasan University, M.E in manufacturing technology & Ph.D in production engineering from Regional Engineering College, Tiruchirappalli. He has published more than 100 papers in international & national journals and conferences. He has more than 22 years of experience. His current research includes non-traditional techniques, machining process optimization and data mining.

P. Asokan works as a professor of production engineering at National Institute of Technology, Tiruchirappalli, India. He received B.E (mechanical engineering) & M.E (manufacturing technology) from Regional Engineering College and Ph.D (production engineering) from Bharathidasan University. He has published more than 50 papers in international & national Journals. He is having more than 25 years of experience. His current area of research includes operation management and optimization in manufacturing, scheduling, non-conventional machining etc.