Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/110225

Multi-Objective Criteria in Hybrid Flow Shop Scheduling Using Improved Genetic Algorithm

M. L. Brabin Nivas[1,*] *and T. Prabaharan* [2]

[1] Ponjesly College of Engineering, Nagercoil, Tamil Nadu, India
[2] Mepco Schlenk Engineering College, Virudhunagar, Tamil Nadu, India

Received: 27 Nov. 2016, Revised: 25 Jan. 2017, Accepted: 27 Jan. 2017
Published online: 1 Mar. 2017

Abstract: Flow shop scheduling problem consists of scheduling n jobs on m machines. As an attempt for meeting this objective, all jobs are allotted with the same sequence of operations, where the problem is analysed in terms of make-span, total tardiness and flow time. In order to solve this scheduling problem, an approach based on Improved Genetic Algorithm (IGA) is developed. In this regard, job data Four Drawer Furniture Component (4dfc) has been collected from the company, from where the time sequence for each operation has been calculated manually. By using genetic algorithm, various sequences have been generated and the make-span time has also been calculated. Two factors namely, crossover and mutation are employed, thus to improve the genetic algorithm used, which in turn reduces the make-span time. Various sequences have been developed by using C where both the manual and program results are correlated. At the end of process, best sequence is found using IGA and the results are validated.

Keywords: Improved genetic algorithm, job data four drawer furniture component, make-span, crossover, mutation

1 Introduction

Scheduling is an activity by which any work can be done in an orderly manner without restriction or disturbance during the planned work process. Generally scheduling is the basic idea of accomplishing a work. The work may be a job or a project, and they are composed of constraints known as operations, availability, delay etc., in a general manufacturing system. Jobs or work are parts or assemblies and activities like machining, assembly and availability and delay are tardiness and machine availability. A good schedule has a series of objectives:

1. To maximize shop through put over a certain time period.
2. To minimize steady costs.
3. To improve quality for customer satisfaction.

In general, a flow shop schedule consists of a sequence of n machines aligned in series, where each machine cell is a group and each machine may or may not be identical and each group consists of only one machine performing a single operation at a time, whereas in a hybrid flow shop system, every group has one or more machines performing different operations in the group. Thus, by reducing make span, lead time of production also can be minimized.

Scheduling being an essential tool in the engineering process influences more on its outcome with the aim of reducing the overall production time, cost and enhancing the operational efficiency. It conveys the production department, the details of the manufacturing date, the device in which the experiment is going to be processed and by whom the action is going to be performed. Results of production scheduling confirm its efficiency over all other conventional techniques. Real time workloads in different production levels get optimised by graphical interfaces that serve as its integral part, where the software creates scheduling mechanism by the effective usage of pattern recognition. For instance, if an airline plans to reduce the gate count of aircraft, scheduling software determines time tables, number of aircraft or the passengers to be arrived by allowing the respective board to fix with the plan.

In order to assign human and machine resources along with other process and purchase materials, forward and backward scheduling mechanisms are employed. In forward scheduling, tasks are allocated from the data resources by which the due date can be calculated, where in case of backward scheduling, tasks are planned from due date to assess the start date or any other alterations.

* Corresponding author e-mail: brabinroyal@gmail.com

There are several advantages on applying the production scheduling. Some of them can be, process change-over, scheduling levels and inventory reduction on real time basis, apart from labour load levelling and enhancement in productivity.

1.1 Metaheuristics

Results are proved in relation with high performance behavior of local search and hybrid genetic algorithms that is intended to solve any flow shop scheduling problems. Commonly, genetic algorithm, tabu search and simulated annealing are the meta-heuristics strategies used for the sake of scheduling.

1.1.1 Genetic Algorithm (GA)

Genetic algorithm works on the basis of natural selection and genetics and has been used extensively in the field of scheduling and sequencing. Benefits of genetic algorithm are infinite in the field of job shop scheduling problems. Classical issues or deviations of them are also resolved by using GA, besides its ability to solve hard combinatorial complexities.

1.1.2 Tabu Search

In solving typical JSSPs, tabu search algorithms are broadly used in accordance with the disjunctive graph concept. The critical path arcs are reversed or the longest path precedence formulations are varied to acquire the neighborhood moves. The difference between tabu search methods and other local search methods lies in their searching criteria, where the results of Tabu search mechanisms have been proven to be better by making use of lesser computational time.

1.1.3 Simulated Annealing

Typical JSSPs are well handled by simulated annealing. While other applications intend to reduce make-span, this particular strategy takes more computational time to yield quality results.

2 GA Scheduling

2.1 Shop Scheduling by GA

Genetic algorithm is one of the attributes of schedule optimization methods, where the good solutions formed are forwarded to produce the better ones which in turn lead to the formation of ideal solutions. It is the feature of this algorithm that the characteristics of good solution make optimal new solutions. This proves that genetic algorithm can be really guaranteed to perform scheduling.

GA is comparatively better than other optimization approaches in correspondence with its nature of traversing large search spaces in a shorter interval of time. Beyond this, its mutation process goes ahead from local minima that results to be more familiar since there is rise in search space size. When n jobs and m machine scheme is taken into consideration, the upper bound for the solutions tends to be $(n!)^m$. The resultant reading would be high, if both n and m are not seemed to be large enough and so if n and m equal to 20 and 5 respectively, the result would be $8.52 * 10^9 1$. Hence in relation to this enlarged search space, conventional scheduling methods like mathematical programming get delayed in their processing.

2.2 Flow Shop Scheduling using Genetic Algorithm

The fitness of each solution among the population is assessed using problem specific objective relation of the genetic algorithm. Computing fitness is the integral part taking place soon after the completion of crossover and mutation. These operations result in optimal solution that even a better solution will not be removed, so that it remains as the near optimal solution, which might be used based upon the requirements. In this application, the appropriate characteristics of problems are analyzed and the factors such as fitness and genetic functions and setting of initial population are determined.

2.3 Scope of GA to Scheduling Problems

Generally genetic algorithm consists of a list of procedures that yields solution for the occurred problems. Unless appropriate results are obtained, genetic algorithm would not end in generating successive populations with possible solutions. In GA, all the alternative solutions are not expected to get analyzed for achieving ideal results. The advantages of GA scheduling can be listed as,

- ◇ Certain typical algorithms are limited with their usage as they rely on problem size, so that these might not able to deal for larger number of operations.
- ◇ Only reasonable solutions could be anticipated using traditional scheduling methods.
- ◇ Heuristic approaches most often come under the risk of problems as they are problem dependent.
- ◇ In certain case, global shop floor goals might get missed due to the use of priority dispatching rules.
- ◇ Genetic algorithm is better in concern with the simulation process as it is assured in providing near optimal solutions.

- ◇ Solutions are resulted for other objectives that make the process still more flexible since evaluation factors of the genetic search mechanism can be changed.
- ◇ For the purpose of obtaining operational schemes, genetic algorithm is extended with that of neural networks as well.

2.4 Objective

The prime objective is to initiate a scheduling process in order to attain multiple targets in minimizing make-span and minimizing total tardiness and flow time.

2.5 Need for Study

On taking into account, n jobs and m machines under scheduling, flow shop problem is defined with the allotted processing time. Basically, it is assumed that m machines processes n jobs of same order. Each machine takes care of a single job just once. A single job gets processed by one machine at a time and handling more than a single job by a single machine is practically impossible most of the time.

3 Literature Review

[1] introduced Hybrid Flow Shop Scheduling (HFSS) problem. Gupta [1988] proved that simple two stage HFSS issue with each stage of two identical machines was NP hard. [2] developed a heuristic algorithm for scheduling in a two-stage parallel-processor flowshop complexity in order to reduce total flow-time.

Initially, [3] took the initiative to solve HFSS complexity of multiple stages by developing a heuristic to minimize makespan for multi-level HFSS problems.

Several algorithms were suggested by [4] to reduce tardiness or finishing time in case of multi stage HFSS issues. They developed the algorithms based on various constructive heuristics and dispatching rules. Lower bound was also developed by them. [4] described many heuristic methods for a two-stage HFSS complexities, thus to reduce make-span. [5] proposed a hybrid heuristic algorithm by employing branch and bound and genetic algorithm to deal with the multi stage HFSS complexities. [6] used simulated annealing algorithm to reduce flow shop make-span with multiple processors. They generated initial solutions by using some heuristics in the first stage and the solutions were improved by simulated annealing algorithm in the second stage. In case of batch scheduling, a mixed integer process was employed in flexible flow lines in order to lower the make-span, where the used immediate buffers are restricted.

[7] presented a combined genetic and tabu search algorithms to lessen the make-span of HFSS complications. [8] used different heuristic algorithms to minimize make-span for a 2-stage FSS problem that is of multiprocessing functions.

[9] applied a heuristic and mixed integer program occurring in a 2-stage hybrid flow shop in which the first and last stage consists of a batch processor and a single machine respectively. The goal is to limit the make-span along with limited waiting time at the second stage.

[10] suggested a new heuristic in the view point of minimizing make-span in a 3-stage no idle flow shops. [11] addressed a branch and bound algorithm to limit jobs total tardiness considering a 2-stage hybrid flow shop. In HFSS problem, a single machine was employed at the first level where in the next level, multiple identical parallel machines were assessed.

[12] developed an efficient and effective approximation technique on the basis of tabu search algorithm with the aim of minimizing make-span for HFSS issues. [13] presented scheduling heuristic to minimize the total tardiness for a hybrid flow-shop. The bottleneck stage was identified and the schedule was constructed first. They used the random problems for testing the algorithm.

[14] developed a branch and bound approach for a 2-stage HFSS problem to minimize make-span considering a single and m machines at the first and second stage accordingly. They also considered the unavailability of machines for their research. [15] proposed a heuristic to minimize make-span for multistage flexible flow-shop problem using uniform parallel machines in both the stages.

[16] developed a new heuristic to minimize make-span for 2-stage assembly flow-shop scheduling issues. Concurrent and assembly tasks were performed in the first and the second levels. A self-adaptive differential evolution approach was developed by [14] to reduce lateness for a 2-stage assembly flow shop scheduling complexity. A hybrid constructive genetic strategy was addressed by [17] to deal with flexible flow shop problems thereby lessening the weighted completion time. In this regard, a couple of fitness function and a local search trained population were employed with that of the strategy used.

[9, 10] illustrated branch and bound method to minimize makespan for a 2-stage assembly scheduling problem. In this work, a job-dependent component across time constraints has been considered. Approaches are focused to handle flexible flow line issues in [5]. They considered independent parallel machines at each stage to minimize total tardiness in their research stage assembly flow shop scheduling problems.

[16] developed four heuristic algorithms in terms of linear programming that minimizes total span for a 2-stage flow shop. It incorporated parallel unrelated machines and renewable supply at each stage. [1] suggested an ant colony optimization method for HFSS issues to minimize the total earliness and tardiness as a

whole. The results are compared with various constructive heuristic algorithms for randomly generated instances.

Recently, [12] solved the HFSS problems to minimize the total cost of resource allotment make-span using a genetic algorithm hybridized with variable neighborhood search mechanism. Both the machine and the resource relied job computation time were analyzed. They developed some random problems and compareD their results with other conventional methods.

4 Methodology

4.1 Problem Description

A four drawer furniture component (4DFC) includes nineteen parts. They are as follows,

- ➔Back Plate (BP)
- ➔Side Plate (Right side & Left side) (SP)
- ➔Top Plate (TP)
- ➔Drawer side (Left side & Right side) (DS - LH & RH)
- ➔Drawer Back (DB)
- ➔DBT
- ➔DFT T.HAN
- ➔Kick Plate (KP)
- ➔Top Plate off - BK
- ➔Top Plate Off - Side
- ➔Slide Side (Right side & Left side) (SS - LH & RH)
- ➔BB
- ➔KPIC
- ➔ATC
- ➔LCH
- ➔LLR
- ➔AT-LB
- ➔LP
- ➔BCP

The above said parts have to be manufactured in batches for a set of 4DFC (100 Numbers).

4.2 Terms and Definitions

The basic terms in the GA are chromosomes, gene, allele, population and fitness.

- Chromosome: Individuals entire genetic details are comprised in a chromosome, which is the totality of genes. A chromosome is divided into two, each contributing to children during reproduction.
- Gene : It lies within a chromosome as a single feature.
- Allele : Genes take upon a certain value, where each gene might be varied in considering such values.
- Population : Several chromosomes are joined together to form a population.
- Objective : It is the function considered for minimization or maximization of a criterion.
- Fitness : Fitness assessment is inverse to objective criterion and is based on the performance of the parameter set. Smaller the fitness value, greater would be the fitness.

Fig. 1 depicts the methodology flowchart.

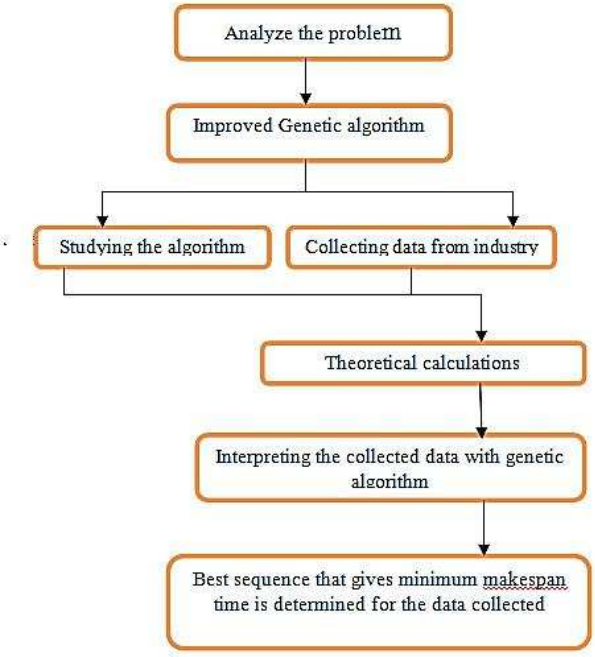

Fig. 1: Flow chart for methodology

4.3 Improved Genetic Algorithm Steps

4.3.1 Generation Of Initial Population

First step is the generation of initial population randomly. Each processing sequence is represented by chromosomes that is corresponding to one job. The chromosomes length equals number of jobs N. Chromosome elements are formed random-wise, where a single chromosome is responsible for the generation of one sequence. The population size varies with respect to problem, where it is assumed to be twice the number of jobs that are yet to be evaluated.

i.e. Population size $= 2*$number of jobs
Population size $= 2*$N

4.4 Evaluation

Evaluation of population is carried out for fitness along with the determination of chromosome selection

probability. Assessing chromosome fitness is according to the flow diagram stated below in Fig. 2,

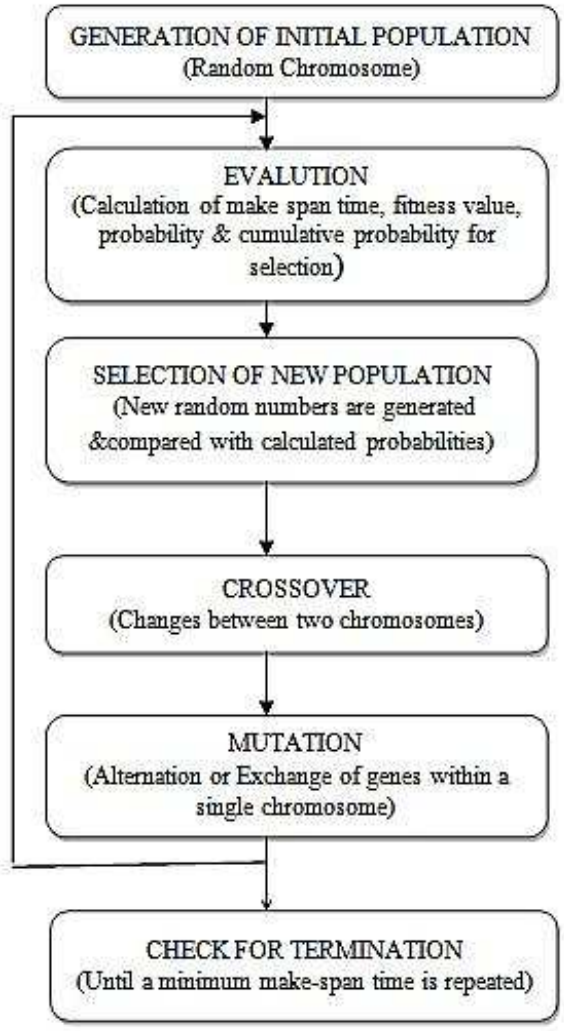

Fig. 2: Flow chart in improved genetic algorithm steps

4.4.1 Evaluation of Objective Function

Objective function f(c) is evaluated by loading jobs in relation to the chromosomes(c) sequence.

$$f(c) = make\ span\ of\ the\ corresponding\ chromosomes$$

4.4.2 Fitness Value

Converting the objective function to a fitness value fit(c) is the next step. The minimization factor is ranked high in correspondence with that of the optimal components and is chosen as parents several times. In this case exponential parameter would make sense, i.e.

$$f(v) = d \wedge k$$

where k refers to a negative number. By the above relation, fit(c) value can be obtained.

$$fit(c) = e^{\wedge} - k * f(c)$$

where, k equals 0.05 (assumption). Around 50% of the good chromosomes would get placed in the new set of population and the fitness value is determined using,

$$fit(c) = e^{\wedge} - 0.05 * f(c)$$

4.4.3 Probability

The fitness function is converted to the chromosome selection probability p(c) and the totality of fitness value is calculated by,

$$p(c) = fit(c)/\Sigma p(c)$$

Then the sum of probabilities of survival (cp(c)) of the entire chromosomes can be obtained by the formula,

$$cp(c) = \Sigma p(c)$$

4.5 Selection of New Population

A random selection method develops the next population of the same size which results in a random number r between 0 and 1. By doing so, a chromosome (c) is chosen that agrees with the following relation,

$$cp(c-1) \leq r \leq cp(c)$$

The selection method gets continued up to its population size. In this relation, the most-fit individuals are chosen and so its reliability factor is high, where its predicted frequency, i.e., the total number of selection time, equals 1. This process is the deciding factor to enable the fittest chromosomes to get multiplied and the least fit to get removed.

4.5.1 Cross Over

In crossover, offspring is produced when parents get combined. By considering the ideal attributes, the offspring chromosomes would be better than the parent chromosomes. Crossover arises in the course of evolution with respect to the user specific cross over probability. Unlike basic GA, here the assumption value(r) for crossover 0.5 has been categorized into two types

- $r < 0.5$ (use I method of crossover(single point crossover))
- $r > 0.5$ (use II method of crossover(two point crossover))

4.5.1.1 Single point crossover

Example:

$$P1 = 3\ 2\ 4\ 5\ 1$$
$$P2 = 4\ 5\ 1\ 2\ 3$$

New Sequence

$$O1 = 3\ 2\ 1\ 5\ 4$$
$$O2 = 1\ 5\ 4\ 2\ 3$$

4.5.1.2 Two point crossover

Example:

$$P1 = 3\ 2\ 4\ 5\ 1$$
$$P2 = 4\ 5\ 1\ 2\ 3$$

New Sequence

$$O1 = 3\ 5\ 1\ 2\ 4$$
$$O2 = 1\ 2\ 4\ 5\ 3$$

4.5.2 Mutation

Mutation must not be the factor inducing for the production of new genetic structure, rather arising of newer ones must be reduced and again, the lost good genes due to improper selection of parents are regenerated. Hence mutation has a great role where the gene pool should be secured from any undesirable changes. Here the assumption value (r) for mutation is chosen as 0.33 and it is divided into three categories as

- ◇ If generate random no value < 0.33(use I method of mutation(flip bit mutation))
- ◇ If generate random no value $0.33 < r > 0.66$ (use II method of mutation(uniform mutation))
- ◇ If generate random no value > 0.66(use III method of mutation(non-uniform mutation))

4.6 Genetic Algorithm Illustration

4.7 Improved Genetic Algorithm Illustration

4.7.1 Crossover

- $R < 0.5$ (using I method of crossover(single point crossover))
- $R > 0.5$ (using II method of crossover(two point crossover))

Table 1: Generation of initial population with corresponding make-span

C	Priority Sequence					F(c)
1	1	2	3	4	5	168
2	2	3	4	5	1	135

Table 2: Evaluations of parameters for the generated initial population

C	F(c)	Fit(c)	P(c)	Cp(c)
1	168	0.000224	0.16115	0.16115
2	135	0.00117	0.84172	1

Table 3: New population selection

C'	R	C	New chromosomes					
			Priority Sequence					F(c)
1	0.15	1	1	2	3	4	5	168
2	0.85	2	2	3	4	5	1	135

Table 4: Crossover

C"	R	S/N	chromosomes									
		S	Parents					Crossover				
1"	0.25	S	1	2	3	4	5	1	3	4	2	5
2"	0.38	S	2	3	4	5	1	4	2	3	5	1

4.7.2 Mutation

1. If Generate random no < 0.33 (using I method (flip bit mutation))
2. $0.33 > r < 0.66$ (using II method (uniform mutation))
3. $R > 0.66$ (using III method(nonuniform mutation))

5 Results and Discussions

Tables 1 to 12 represent the calculated results for using single machine and multi machine. Time is calculated manually for various sequences of operations. It is found that for single machine, sequence drilling, power pressing, welding, bending, punching are controlled in order to produce less make-span time and for multi-machine, sequence bending, welding, power pressing, drilling, punching are the factors to be considered to produce less make-span time.

By using genetic algorithm, the best sequence are 1 4 2 3 5 and 2 5 3 4 1 and the ideal corresponding times are 147, 146 sec and the make-span is reduced by using improved genetic algorithm from 147 to 126 and 166 to 125 sec.

5.1 Program Results

Table 5: Obtaining new population after mutation

C”’	R	Sequence									
		Before mutation					After mutation				
1”’	0.02	1	3	4	2	5	1	2	4	3	5
2”’	0.04	4	2	3	5	1	4	5	3	2	1

Table 6: Crossover

C”	R	I method or II method	chromosomes									
			Parents					Crossover				
1”	0.25	I method	1	2	3	4	5	1	2	4	3	5
2”	0.38	I method	2	3	4	5	1	2	4	3	5	1
3”	0.60	II method	1	3	4	2	5	1	4	3	5	2
4”	0.72	II method	4	2	3	5	1	3	5	4	2	1

Table 7: Generation of new population after mutation

C”’	R	Sequence									
		Before mutation					After mutation				
1”’	0.25	1	2	3	4	5	1	4	2	3	5
2”’	0.40	2	4	3	5	1	2	5	3	4	1
3”’	0.52	1	4	3	5	2	1	5	3	4	2
4”’	0.70	3	5	4	2	1	3	4	5	2	1

Table 8: Single machine calculation

Sequence of operations	Time (Sec)
1-19	763
19-1	824
Random sequence	666

Table 9: Multi-machine calculation results

Sequence of operations	Time (Sec)
1-19	168
19-1	156
Random sequence	135

Table 10: Basic genetic algorithm

After Mutation (Sequence)	Total Make-span(Sec)
1 2 4 3 5	147
4 5 3 2 1	156

Table 11: Improved genetic algorithm

After Mutation (Sequence)	Total Make-span(Sec)
1 4 2 3 5	126
2 5 3 4 1	125

Table 12: Program results

Simulated annealing algorithm	Improved genetic algorithm
Sequence 13 20 18 6 16 2 8 9 14 19 10 7 12 3 17 1 15 11 5 4	Sequence 7 15 14 19 16 6 20 1 9 8 13 18 3 11 5 2 17 10 12 4
7183 Secs	7016 Secs

6 Conclusion

In this flow shop scheduling with n job on m machine, all jobs have the same sequence of operation and scheduling can be enhanced by using the improved genetic algorithm. Also the make span can be calculated through this and it can be reduced drastically compared to the previous genetic algorithm. It is shown from the illustrations that the efficiency of genetic algorithm in solving a flow shop scheduling can be improved significantly by tailoring the genetic algorithm operations to suit the structure of the problem. The computation results based on crossover and flow shop scheduling bench mark problems show that genetic algorithm gives a better solution when compared with the earlier reported results.

```
//HEADER FILES
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
//GLOBAL DECLARATION
int count=0,count1=0,count2=0,count3=0,count4=0,
count5=0,total_avg_sec=0;
int punching[100];
int bending[100];
int welding[100];
intpower_pressing[100];
int drilling[100];
int a[10],job,n;
//STRUCTURE INITIALIZATION
struct FDFC
{
char model[15],component_name[15];
char operation[15],machine[5];
intquantity,avg_time,batch_size;
};
//CALCULATE FUNCTION
int calculate(char operation1[15])
{
int j=0;
if(strcmp(operation1,”PUNCHING”)==0)
{
for(j=count-1;j>=0;j–)
if(punching[j]!=0) break;
if(count1==0) punching[count1++]=total_avg_sec;
else
punching[count1++]=punching[count1-1]+total_avg_sec;
}
```

```
else if((strcmp(operation1,"BENDING"))==0)
{
for(j=count-1;j>=0;j–)
if(bending[j]!=0) break;
if(count2==0)
bending[count2++]=punching[count1-1]+total_avg_sec;
else
{
if(punching[count1-1]>ending[j])
bending[count2++]=punching[count1-1]+total_avg_sec;
else
bending[count2++]=bending[j]+total_avg_sec;
total_avg_sec=bending[count2-1];
}
}
else if(strcmp(operation1,"WELDING")==0)
{
for(j=count-1;j>=0;j–)
if(welding[j]!=0) break;
if(count3==0) welding[count3++]=total_avg_sec;
else
{
if(bending[count2-1]>welding[j])
welding[count3++]=bending[count2-1]+total_avg_sec;
else
welding[count3++]=welding[j]+total_avg_sec;
total_avg_sec=welding[count3-1];
}
}
else if(strcmp(operation1,"PRESSING")==0)
{
for(j=count-1;j>=0;j–)
if(power_pressing[j]!=0) break;
if(count4==0) power_pressing[count4++]=total_avg_sec;
else
{
if(welding[count3-1]>power_pressing[j])
power_pressing[count4++]=welding[count3-
1]+total_avg_sec;
else
power_pressing[count4++]=power_pressing[j]+total_avg_
```

References

[1] H.C Chang, Y.P Chen, T.K Liu and J.H Chou, IEEE Access **3**, 1740-1754 (2015).

[2] J. fernandogoncalves, J.J Magalhaesmendes and M.G.C Resende, European Journal of Operational Research **167**, 77-95 (2005).

[3] R. Ruiz and Concepcionmarato, European Journal of Operational Research **169**, 781-800 (2006).

[4] T.K Varadharajan and C. Rajendran, European Journal of Operational Research **167**, 772-795 (2005).

[5] A. Fink and S. Vob, European Journal of Operational Research **151**, 400-414 (2003).

[6] B. Yagmahan and M.M Yenisey, Expert Systems with Applications **37**, 13611368 (2010).

[7] A. Janiak, E. Kozan, M. Lichtenstein and C. Oguz, International Journal of Production Economics **105**, 407424 (2007).

[8] V.S Jorapur, V.S Puranik, A.S Deshpande and M. Sharma,Journal of Software Engineering and Applications **9**, 208-214 (2016).

[9] A.K Parvathi, Journal of Excellence in Computer Science and Engineering **1**, 25-33 (2015).

[10] B. Park, H.R Choi and H.S Kim, Computers & Industrial Engineering **45**, 597613 (2003).

[11] E.G Negenman, European Journal of Operational Research **128**, 147-158 (2001).

[12] Y. Li and Y. Chen, Journal of Software **5**, 269-274 (2010).

[13] E. Demirkol, S. Mehta and R. Uzsoy, European Journal of Operational Research **109**, 137-141 (1998).

[14] V.G Gopika and N. Alex, Journal of Excellence in Computer Science and Engineering **1**, 11-22 (2015).

[15] S.R Hejazi and S. Saghafian, International Journal of Production Research **43**, 28952929 (2005).

[16] R. Ruiz and J.A.V Rodriguez, European Journal of Operational Research **205**, 118 (2010).

[17] J.S Chen, J.C.H Pan and C.M. Lin, Expert Systems with Applications **34**, 570577, (2008).

M. L. Brabin Nivas is working as an Assistant Professor in the Department of Mechanical Engineering, Ponjesly College of Engineering, Nagercoil. He is specialised in CAD/CAM.

T. Prabaharan is working as Professor in the Department of Mechanical Engineering, Mepco Schlenk College of Engineering, Sivakasi. He is specialised in Industrial Engineering. He has published his papers in 16 international journals and one national journal. He has participated in 16 international conferences and 22 national conferences and have 27 years of teaching experience.