Application of a Heuristic for solving the 15-20 jobs 3-5 Stages Flexible Flowshop Scheduling Problem

Vanessa Patricia Manotas Niño
Universidad de La Sabana, Chía (Cundinamarca), Colombia, vanessa.manotas@unisabana.edu.co

ABSTRACT

Hybrid flowshop, or more commonly named Flexible flowshop, is a frequently problem in the industries in where all proccesed jobs have the same production routing, and therefore uses the stages in the same order. This paper shows the applicability of a heuristic algorithm to determine and optimize the average completion time when there are several jobs to process (15 to 20 jobs) in multiple stages (3 to 5 stages), and each job is associated to one specific weighted. The results showed by the algorithm are obtained from the interaction between the process, release and setup times; main constrains of this scheduling problems. The main contribution of this paper is to schedule a set of n jobs so as to minimize the makespan, developing a model that allows to identify the logical sequence of jobs to carry out for each stage.

Keywords: Flexible flowshop, Average Completion Time, Process Time, Release Time, Setup Time

1. INTRODUCTION

In the practice, all industries take into consideration basic several objectives in order to get optimal results like for example the production secheduling. One of these objectives is minimizing the Average Completion Time, denoted by w_jC_j and defined as the time the job j leaves the system contemplating an associated weighted wj. In flexible flowshops problems, like in other scheduling problems is essential to reflect the optimization of the Average completion time as one of the main objectives to consider with the intention of satisfying the clients requirements. To satisfy the demand in a suitable time and having control of this, allows companies to work under a discreet scheduling of the times of production, organization and distribution of products. By doing this, indisturies will ensure to have the capacity to fulfill the dates and hours "timing" that are decided with the clients. However, for the companies it is very important of planning and making this shceduling do to the cost benefits (cost reduction) it can bring and also taking into account the storage capacitiy.

A production line structured as a flexible flowshop requires all jobs (products) to visit the workstations in series. In the case there exists different (speed condition, setup, ect) parallel machines, each machine should be in the capacity of processing the job in file and they have to be processed without interruption once it starts. Minimizing properly the Completion Time, balancing the productivity, and minimizing the sequence of setup times, maximizes the throughput rate. It is for this reason that is so important to develop mechanisms that allow programmers to make more agile the process of scheduling assuring it more effective.

For solving this type of problems, there are a lot of references that propose heuristic and other methods to work it out. Kis and Pesch) in its work "A review of exact solution methods for the non-preemptive multiprocessor flowshop problem", make a point saying flexible flowshop problem is a generalization of the flowshop in such a way that every job can be processed by one among several machines on each machine stage. In this paper we provide the first comprehensive and uniform overview on exact solution methods for flexible flowshops with branching, bounding and propagation of constraints under two different objective functions: minimizing the makespan of a schedule and the mean flow time.

On the other hand (Babayan and David, 2004) proposed a general methodology of agent-based manufacturing systems scheduling, incorporating theoritic game analysis of agent cooperation is presented to solve the n-job 3-

stage flexible flowshop scheduling problem. Their objective is to schedule a set of n jobs with the intention to minimize the makespan. They performed error bound analysis using the lower bound estimates developed in the literature as a datum for comparing the agent-based scheduling solutions with other heuristic solutions.

(Logendran, Carson and Hanson, 2005) of Department of Industrial and Manufacturing Engineering, Oregon State University, present a methodology for solving this important problem, namely group scheduling, within the context of cellular manufacturing systems in order to minimize the total completion time of all groups of jobs considered in the planning horizon.

(Yamada and Reeves, 2002) present in their investigation Permutation Flowshop Scheduling by Genetic Local Search the landscape for the permutation flowshop scheduling problem (PFSP) with stochastic local search and a critical block-based neighborhood structure has been investigated. An approximation method for PFSP that would make use of this big valley structure is proposed by using a critical block-based neighborhood structure, and a genetic local search method called MSXF-GA, previously developed for the job shop scheduling problem. Computational experiments using more challenging benchmark problems demonstrate the effectiveness of the proposed method.

This article is structured as follows: section 1 describes the problem to solve; section 2 shows the base for develop the heuristic; section 3 illustrates the algorithm; section 4 contains the results of the heuristic; and section 5 concludes the derived solutions.

2. Problem Definition

The model of flexible flowshop consists in a set of n jobs (j_1, j_2, ..., j_j , ..., j_n) which have to be sequenced on m stations (st_1, st_2, ..., st_i, ..., st_m), arranged in series. Each job has m sets of operations and requires its first set of operations on station 1; it's second on station 2, and so on. The set of sequences describes the order in which the jobs are processed on the m stations.

The processing time P_j of job j on station m is known and constant. The time which the job j enters the system is called release time r_j and the time which the job j exits the system is called completion time C_j. In this case in that the setup time is concerned, an additional time S_{jk} may occur, necessary to change the setup of station i, in order to be able to process job k.

The problem treated in this paper is the programming of 15 to 20 jobs in 3 to 5 Stages, each one with 1 to 3 different machines in parallel (different rate production).

The jobs can be processed in anyone of the machines of the Stages. All the works are available in time zero and each machine must finish completely the processing of a work to initiate the processing of the following work, it means a non-preemptive job.

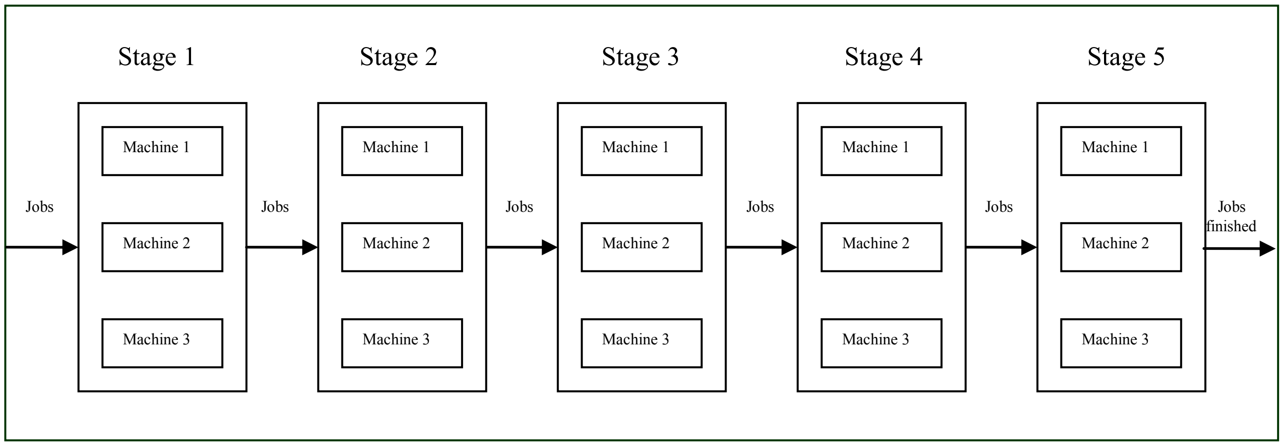

Figure 1. Structure of manufacturing system

The problem can be formulated using the following annotation and defining the following variables and restrictions:

P_j is the process time of job j (j = 1,…, n).
r_j is the release time of job j (j = 1,…, n).
S_{jk} is the setup time (j, k = 1,..., n).

As the objective is minimizing the average completion time, each job j has associated a weighted w_j, in order to complete the formula:

$$\sum_{j=1}^{n} w_j C_j \tag{1}$$

3. DEVELOPMENT OF THE HEURISTIC

To minimize the criteria of average completion time in problems of flexible flowshops, the heuristic was developed having in consideration the process time of each job j, P_j, the release times (the times the job arrives at the system) of each job j, r_j, and the setup times (the machine preparation time depending of the sequence), S_{jk}, resolving the problem: $\mathbf{FF_c \mid r_j, S_{jk} \mid w_jC_j}$

The tests of the algorithm took control of: 3 to 5 Stages, each one with 1 to 3 machine in parallel; j, k = {1, 2, ..., N}, N ∈ [15, 20].

The heuristic algorithm developed in Visual Basic programming language, contains a programmer interface which asks for the data like shown in the next example: Jobs number = 18 and Stages number = 5.

Table 1. Number of machine in each Stage

Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
M1	M1	M1	M1	M1
M2	M2	M2		M2
	M3			M3

Table 2. Velocity

Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
3	3	1	1	2
1	4	2		5
	1			1

Table 3. P_j in each standard machine

Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	6	3	28	6	16	19	1	6	11	22	28	12	30	28	30	18	14	29		
P2	27	27	6	3	24	21	6	4	15	9	30	7	30	12	30	17	18	3		
P3	10	24	24	1	6	14	1	20	5	8	21	16	1	24	20	3	19	25		
P4	23	17	25	2	13	25	1	9	13	16	2	6	26	4	27	17	12	30		
P5	26	14	6	7	4	22	8	2	21	11	8	16	5	3	17	21	3	9		
rj	23	22	21	23	0	23	17	9	1	7	2	21	25	8	8	25	3	9		
Wj	2	8	1	4	1	7	1	5	6	3	4	4	4	8	3	6	3	3		

Table 4. S_{jk} Stage 1

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1		10	8	14	4	13	8	6	9	6	12	2	10	9	14	7	9	1		
2	10		9	5	8	12	4	6	11	2	8	3	4	0	3	2	3	12		
3	8	12		6	14	2	9	10	2	8	14	11	4	2	11	9	14	2		
4	8	13	8		6	9	12	1	2	1	14	7	5	13	6	11	7	1		
5	2	11	4	1		11	1	6	3	13	0	12	2	12	10	4	12	8		
6	13	0	10	6	8		7	3	4	3	12	8	2	8	6	13	0	5		
7	14	8	6	9	13	10		2	14	2	10	3	9	6	6	1	5	4		
8	14	2	14	13	3	0	6		5	13	6	3	2	8	3	12	4	0		
9	4	0	4	6	5	13	8	6		3	3	6	1	8	13	5	8	5		
10	0	11	3	6	7	9	0	4	1		10	9	7	3	0	4	2	13		
11	3	11	13	3	8	7	13	8	5	10		4	4	6	12	2	5	5		
12	7	4	4	7	11	12	3	5	2	3	13		7	1	10	2	4	2		
13	9	2	0	11	11	0	10	3	6	11	7	13		10	11	3	4	0		
14	0	2	0	4	4	3	9	11	14	10	1	2	9		9	14	9	5		
15	9	3	14	10	4	11	1	13	0	14	8	1	3	8		9	4	13		
16	0	12	11	6	9	3	7	14	1	12	3	9	4	4	3		0	5		
17	13	13	2	0	9	0	3	12	0	1	0	11	7	4	1	1		6		
18	10	1	10	14	0	4	10	2	13	0	4	8	9	14	14	5	5			
19																				
20																				

S_{jk} Stage 2… Stage 5

Once the programmer indexes all the data, comes itself to execute the algorithm.

4. ALGORITHM

Algorithm Minimizing $\sum w_j C_j \quad FF_C \left| r_j, s_{jk} \right| w_j C_j$

Input

The input the entrances of the algorithm they are:

- The number of jobs to process between 15 and 20, the number of stations between 3 and 5 and the number and velocity of the machines in each station which are 1 and 3 machines.
- A matrix where is the P_j and r_j of each job.
- A matrix where is the S_{jk} of each Job to the different stations.

Step 1

Sort considering r_j in ascending order and broken ties P_j longest processing time (LPT). Prioritize the job: Select the job with the smallest r_j.

Step 2

Begin Schedule the jobs in the first station with the sequence previously mentioned (highest priority job on the machine capable of finishing it first).

In the case of a tie between two or more machines, the quickest machine is selected (the P_j of the job to assign is divided by the velocity of the machines on that stage; in the machine that result the minor P_j the job is processed).

To select the machine and assign the next work it is computed the C_j for each one considering the following restrictions:

If $r_j > C_{j\ before}$

$$C_{jnew} = P_J + r_j + s_{jk} \qquad (2)$$

Else

$$C_{jnew} = P_J + s_{jk} \qquad (3)$$

End

And looking for the minimum value

$$M_j \middle| C_j = \min(C_i)\forall i \qquad (4)$$

Step 4

The algorithm continues with the same restriction to other stages, considering that job that finished is programming immediately.

Step 5

When all the programs are assigned in completes stage the algorithm finishes.

Step 6

To calculate the $\sum W_j C_j$, the C_j for each work is related to the corresponding W_j.

5. COMPUTATIONAL COMPLEXITY

Due to the variables were vectors and the algorithm of ordering used only had two cycles "for" for doing their function, the complexity of the heuristic is delimited by the complexity of the algorithm of ordering used, that in our case, had a complexity of n^2 (2 "for" indicates n^2). The rest of the heuristic consisted in a series of comparisons that did not increase the complexity significantly and therefore were not representative at the time of calculating it. In short, we have:

$$T(n) = O(n^2), \qquad (5)$$

The computational complexity of our algorithm of ordering was finding of this way:

$$T(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{n} K \qquad (6)$$

Where T(n) is the execution time, K is the number of instructions that the cycle execute and n is the vector size.

$$T(n) = \sum_{i=1}^{n-1} K * n = K\sum_{i=1}^{n-1} n \rightarrow K(n-1)(n) = k(n^2 - n) = (kn^2 - kn)^{n\rightarrow\infty} \approx O(n^2) \quad (7)$$

6. ANALYSIS OF RESULTS

This algorithm was proved with 22 instances generated in random form. The results obtained were the following:

Table 5. Results obtained from the implementation of the algorithm CMV,JLVW and GMEC

	Groups		
Instance	CMV	JLVW	GMEC
JAL1	13397,00	9618,12	3586,66
JAL2	4197,50	3409,72	2619,98
LRP1	12286,77	12639,01	3646,31
LRP2	15096,67	14890,58	17175,61
VRN1	9136,50	8368,45	7815,91
VRN2	8015,00	5926,00	7839,00
WMLLP1	19380,33	16885,71	7261,00
WMLLP2	3707,00	3813,83	2831,00
GBC1	2832,18	8279,11	21185,71
GBC2	6647,95	7104,88	3767,47
MCS1	2110,84	2318,34	3246,46
MCS2	4818,86	6773,52	3431,09
ELS1	4680,59	9385,06	3881,00
ELS2	4063,16	4054,01	1577,08
CTB1	4346,47	6202,60	3808,00
CTB2	2102,50	2898,52	2267,97
CAZ1	4238,00	5015,50	5891,00
CAZ2	2783,50	2216,50	2652,00
MCOA1	2754,00	2587,42	1276,96
MCOA2	2905,08	1972,70	1811,81
VMN1	2902,92	2015,46	1175,95
VMN2	2505,80	2278,17	1787,52

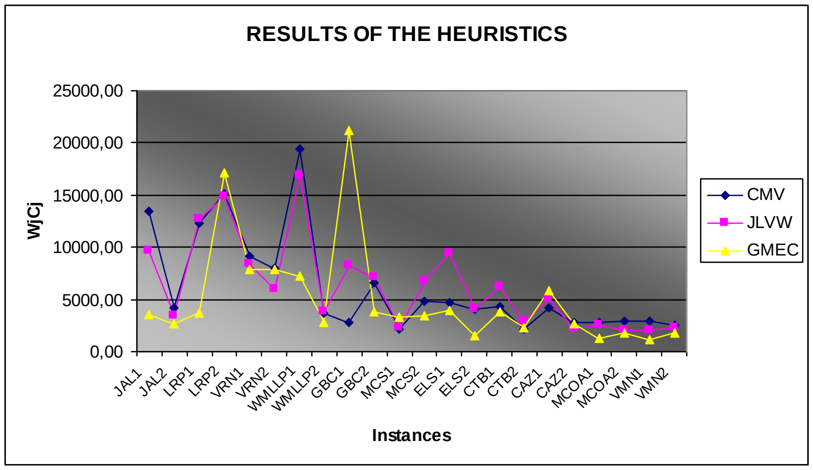

Figure 2. Interaction plot between instances and W_jC_j

The results obtained starting from the implementation of the algorithms proposed indicate that on average the solutions to carry out by the GMEC algorithm are more optimal in comparing with the others. In second place, we find to the solutions of CMV algorithm and finally in third, the solutions of the JLVW.

7. Conclusions

This paper develops a heuristic and compares some constructive and iterative heuristics for flexible flowshop scheduling problems with unrelated parallel machines, where a sequence-dependent setup time is necessary before starting the processing of a job. As objective function, this paper considers the minimization of the average weighted completion time or total weighted completion time of jobs. The best result obtained with the heuristic one was in the instance of GBC1 and the worst result obtained was in the instance of WMLL P1.

In design and implementation of a group of algorithms, there exist important decisions that for a same problem can cause that the results vary in significant ranges. The range of these goes from the type of parallel platform in where the implementation will be carried out, to the determination of diverse parameters, depending on the heuristic one.

It is necessary a bigger study on the different alternatives of combination of the existing algorithms and the determination of the appropriate heuristics.

Concluding, we can affirm that GMEC presents more robust solutions in the appearance of a group of problems with different characteristics and difficulties, transforming into an option been worth for the resolution of scheduling problems. Nevertheless, we also find feasible the use of CMV like an option now that it also minimizes the outlined objective.

REFERENCES

Babayany, A. and Hey, D. (2004). Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. Taylor & Francis Group, Vol. 42, No. 4, pp 777–799.

Kis, T. and Pesch, E. A review of exact solution methods for the non-preemptive multiprocessor owshop problem. Computer and Automation Research Institute, Hungarian Academy of Sciences, 1111 Budapest, Kende utca 13-17, Hungary. Faculty of Economics and Business Administration, University of Siegen Holderlinstrasse 3, 57068 Siegen, Germany.

Logendran, R., Carson, S., and Hanson, E. (2005). Group Scheduling Problems In Flexible Flow Shops. Department of Industrial and Manufacturing Engineering, Oregon State University, Corvallis, Oregon 97331-2407.

Yamadaand, T. and Reeves, C**.** (2002). Permutation Flowshop Scheduling by Genetic Local Search**.** NTT Communication Science Labs, Kyoto, Japan, Coventry University, UK.

Authorization and Disclaimer