23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

COMPARISON AMONG HEURISTIC METHODS TO PROVIDE AN INITIAL SOLUTION FOR A BRANCH-AND-BOUND ALGORITHM TO MINIMIZE THE MAKESPAN IN A FLOWSHOP WITH BLOCKING ENVIRONMENT

Felipe Borreiro Sanches, felipebsanches@outlook.com[1]
Mauricio Iwama Takano, takano@utfpr.edu.br[1]
Marcelo Seido Nagano, drnagano@sc.usp.br[2]

[1]Universidade Tecnológica Federal do Paraná câmpus Cornélio Procópio, Av. Alberto Carazzai, 1640 – Cornélio Procópio – PR. CEP 86300-000,
[1]Universidade de São Paulo câmpus São Carlos, Av. Trabalhador São-carlense, 400 – São Carlos – SP. CEP 13566-590,

Abstract: *This paper has the objective to compare the use of diferent methods to obtain an initial solution for the Branch-and-Bound algorithm with the objective of minimizing the makespan in a flowshop with zero buffer environment. As the problem is known to be NP-Hard, the Branch-and-Bound algorithm may take great computational time to find the best solution. The use of an initial solution may reduce the computational time, by providing an initial upper bound. The efficiency of the use of an initial solution for the Branch-and-Bound algorithm is measured by comparing the traditional algorithm (without an initial solution) and four other algorithms which use different initial solutions provided by four different I phase constructive heuristics (according to Framinan et al., 2004 classification) each. The heuristic methods used to provide the initial solutions are: MM (Ronconi,2004); PF (McCormick et al., 1989); PW and wPF (Pan and Wang, 2012). The Branch-and-Bound algorithm used as well as the lower bound were proposed by Ronconi (2005). The five methods were tested using a 180 problems database. Results show that the use of an initial solution does reduce considerable computational time.*
Key-words: *Branch-and-Bound, heuristics, flowshop, block, makespan,scheduling.*

1. INTRODUCTION

Scheduling is a decision making process that is daily used in many services and manufacturing in industry with the objective of optimizing one or more objectives, such as the makespan, number of delayed jobs, among others. It handles the allocation of resources operations (Pinedo, 2008).

Especially in production, scheduling is the decision process of the processing order, that is, which job should be processed first and which should be processed next. According to (Pinedo, 2008) a good programming can lead to a minimization of costs, waste, and manpower of a company, allowing much greater projection of growth.

In flowshop environments with blocking constraint and zero buffer represents the lack of intermediate queues between machines, hence, after a job j is processed in machine k and machine k+1 has not finished processing job j-1, job j remains in machine k blocking it from starting the process of job j+1.

The use of an initial solution for the branch-and-bound will always result in a number of nodes lower than or, at worst, equal to the number of nodes obtained by the classical application of the branch-and-bound algorithm, however it may not always result in lower computational time. This is because the computational time required to solve the heuristic method to obtain an initial solution may be greater than the computational time required to solve the additional nodes of the branch-and-bound algorithm. This is the same reason why it is not certain that the best heuristic method will provide the lowest computational time.

During the past 40 years, the scheduling problem with the objective of minimizing the makespan in flowshop environment with blocking constraint has been studied by many researchers such as Companys e Mateo (2005); Ronconi (2005); Ronconi and Armentano (2001); Kharbeche (2011). This problem, which is described as Fm|block|Cmax according to Pinedo (2008), consists of scheduling n jobs that must be processed by m machines always with the same flow.

Companys and Mateo (2005) proposed an improved branch-and-bound algorithm to solve these problems and auxiliary heuristics, to get a good initial solution in Fm|prmu|Cmax and Fm|block|Cmax problems, in which, prmu refers to the permutational environment. The auxiliary heuristics are built by two steps: the first step a permutation is obtained, and in the second step a local search procedure is applied. Companys and Mateo (2005) concluded that the difference in behavior between Fm|prmu|Cmax and Fm|block|Cmax, when both are subjected to heuristics, shows that

the computational time required by the type problem Fm|block|Cmax is higher than in the permutation. Companys and Mateo (2005) also proposed a LOMPEN algorithm, which main idea is the ability to simultaneously run two algorithms of branch-and-bound. They obtained good results, solving eight problems unsolved by other methods.

Ronconi and Armentano (2001) presented a branch-and-bound algorithm to minimize the total lateness in flowshops environments with block. They proposed a lower bound for this criterion, and lower bounds for the makespan and the flow time. Ronconi and Armentano (2001) concluded that the tests by the branch-and-bound method performed by them, obtained satisfactory results, significantly reducing the numbers of nodes to be computed.

Ronconi (2005) proposed an algorithm that exploits the occurrence of blocking in order to minimize the makespan time. His conclusion in numerical experiments showed that the limitation scheme works better than the lower bound proposed by Ronconi and Armentano (2001). As an extension of his work Ronconi (2005) proposes the development of a dominance rule.

Kharbeche (2011) proposes the study an accurate method for the problem he named as Celular Robotic Problem (CRP), which is a case of generic flowshop. Kharbeche (2011) explains his new mathematical programming method, which is used to computationally solve small problems, and then explains and presents its branch-and-bound algorithm and said that the model of branch-and-bound algorithms can solve major problems because it requires less processing time. Kharbeche (2011) has resulted in his mathematical method failed to resolve 55% of the proposed issues, while the branch-and-bound method failed to resolve 35% of them.

The heuristic methods consist of three phases, they are (Framinan; Gupta and Leisten, 2004):

- Phase I: Development Index;
- Phase II: Construction of the solution;
- Phase III: Solution improvement.

Phase I

The Development index according Framinan, Gupta and Leisten (2004) is the phase in which the jobs are usually arranged according to the processing time of each job on each machine. The result of this phase is a sequence of jobs that can be used as an initial solution for the next phases, or used as the solution of the problem.

Phase II

At this phase, a solution is constructed in a manner that is inserted into a partial sequence in one or more positions the jobs not yet sequenced, to create a full sequencing (Framinan, 2004). In such a manner, this phase consists of several repetitions of this action.

This phase is divided into two phases:

- Selection of the Jobs, that is, the selection of the jobs that will be inserted in a partial sequence;
- Insertion of Jobs, that is, where to insert the selected jobs in the partial sequence.

The selection of the jobs may be according to any selected index from Phase I, or by trying to insert it to get the best result.

After selecting the job, the next step is the insertion of this job in the partial sequence in order to decide its best position and get the best result as a result of this phase.

Phase III

In this last phase an existing solution is improved by some meta-heuristic, or from some initial solution process that aims to change the sequence of jobs in reason to improve the solution (Framinan; Leisten and Gupta, 2004). Thus, the solution will always be found at this stage better than or equal to the initial solution.

McCormick et al. (1989) developed an algorithm named Profile Fitting (PF), which is an algorithm for environments with finite size buffer, blocking occurring when these buffers are full. This algorithm takes as the first job in the processing sequence, one in which the sum of the processing time of all the machinery is smaller. The result obtained from 5 tests between two variations of PF reveals that the two algorithms have good results for the developed tests, and qualified as one of the best heuristic methods already programmed.

Pan and Wang (2012) adapted the PF algorithm (McCormick et al., 1989) creating an improved form of the same named PW and the algorithm Weighted Profile Fitting (wPF) that follows the same rules as the PF method, except that there is a weight factor w_i weighing up the terms of the sequencing order. The results showed that both wPF and PW result better than many heuristic methods existing MM and PF, the insert sequencing techniques in which they introduced, resulted in an improvement of 1.5% in CPU processing time.

Ronconi (2004) analyzed the minimization of makespan of a problem in flowshop environment with block. An algorithm called MinMax (MM) and compared to PF considered the best algorithm proposed in the literature. The results showed that although the MM achieved good results, it has been surpassed by the PF, which showed potential for major problems.

2. BRANCH-AND-BOUND METHOD

In the branch-and-bound algorithm each node of the tree corresponds to one sub-problem, which is defined by a sub-sequence of jobs. Each subsequence is called partial sequence (PS) and the sequence of jobs out of PS is called non partial sequence (NPS). When a node is branched one or more nodes can be generated by adding one or more jobs to the partial sequence associated with the node that was branched. The next node to be branched is the one with more jobs in the partial sequence. In case of ties the algorithm selects a node with the smallest lower bound. The lower bound is calculated for each node (Ronconi, 2005).

Branch-and-bound, according to Kharbeche (2011), is a widely used method for solving difficult combinatorial optimization problems. Typically the branch-and-bound method is characterized by two fundamental procedures (Kharbeche, 2011):

- Branching: Process of a problem division into one or more smaller sub-problems.
- Bounding: Process of calculating a bound (lower and/ or higher) to the evaluation criteria used.

The branching procedure replaces the initial problem by a new smaller set of problems than the original. This method is used to find the most accurate solution, systematically analyzing the subsequences of possible solutions, or through a solution of tree nodes that represent these sub-sequences.

The branch-and-bound method although viable be used to calculate the makespan, can consume a high computational time when it comes to larger problems. One way out for these cases is to implement an initial solution to the problem, which can generate a possible gain in computational time decreasing the number of nodes necessary.

The initial solution provided by heuristic methods was applied together with the branch-and-bound algorithm as a way of achieving the result faster, it means decreasing the makespan and the number of nodes by a prior result provided by the heuristic method. This result reduces the possibilities of the branch-and-bound algorithm tree, hence, reducing the number of nodes to be branched, through the adoption of an upper bound equal to the initial solution that makes the algorithm reach its result faster.

In this paper, in addition to evaluate whether the use of an initial solution as an efficient method of reducing the makespan and the number of nodes, which of the evaluated heuristic method provides the most improvement to the computational time. The branch-and-bound algorithm as well as the lower bound was proposed by Ronconi (2005).

2.1. Lower bound

The lower bound for the makespan is obtained by calculating the lower bound for the completion time of the last job in NPS on machine *m*. The lower bound used for the branch-and-bound algorithm was proposed by Ronconi (2005). The completion time of the last job on machine *m* is greater than or equal to *LB*2, where:

$$L2(k) = D_{[\mathrm{PS}],\mathrm{k}} + \sum_{g=1}^{|\mathrm{NPS}|} \max\left(\mathrm{a}_{\mathrm{k}}^{\mathrm{g}}, b_{k+1}^{g}\right) + \sum_{q=k+1}^{m} p_{\min_q} \tag{1}$$

$$LB2 = \max_{1 \le k \le m}\{L2(k)\} \tag{2}$$

$p_{\min_q}$ is the smallest processing time on machine *k* among all jobs in NPS;

$\mathrm{a}_{\mathrm{k}}^{\mathrm{g}}$ is the *g*-th smallest processing time on machine *k* among the jobs in NPS;

b_{k+1}^{g} is the *g*-th smallest value between $D_{|PS|,k+1}$ - $D_{|\mathrm{PS}|,\mathrm{k}}$ and the processing times of jobs in NPS on machine *k* among the jobs in NPS on machine *k*+1 without $p_{\min_{k+1}}$ and $b_{m+1}^{g} = 0$ for all *g*.

2.2. Initial Solution

The best heuristics methods for Fm|block|Cmax were used to provide and initial solution for the branch-and-bound algorithm. Those are MinMax (*MM*), Profile Fitting (*PF*), Weighted Profile Fitting (*wPF*) and *PW*.

In MinMax (*MM*) algorithm, proposed by Ronconi (2004), are used three rules: 1. Set the first job the one with the smallest processing time on the first machine; 2. Set the last job of the sequence the one that has the smallest processing time on the last machine; 3. For the remaining jobs, the consecutive job (*c*) of the already sequenced job (*i*) is the one that gets the smallest result in equation 3. Where α is a constant used to weight the two terms of the expression.

$$\alpha\sum_{l=1}^{\text{m-1}}\left|P_{[c],l}-P_{[i],l+1}\right|+(1-\alpha)\sum_{k=1}^{m}P_{[c],k} \tag{3}$$

For the Profile Fitting algorithm (*PF*), created by McCormick et al. (1989), two rules are used: 1. Set the first job in the sequence the job with the smallest sum of processing times on all machines; 2. Then equation 4 is used to calculate the possible idle and blocking times provided by the insertion of each job that has not yet been sequenced in position *c* of the sequence. The job (*j*) that obtains the smallest value for $\delta_{j,c}$ is determined as the next job (*c*+1) in the sequence.

$$\delta_{j,c}=\sum_{k=1}^{\text{m}}\left(D_{[c+1],k}-D_{[c],k}-P_{j,k}\right) \tag{4}$$

Idle and blocking times caused by earlier jobs and by earlier machines may have larger effects on the makespan value than those caused by later jobs and machines. Therefore, the Weighted Profile Fitting (*wPF*) algorithm, proposed by Pan and Wang (2012), applies a weight factor (w_k) to $\delta_{j,c}$, which differentiates the effect of machines in different stages and jobs in different positions. The first job of the sequence is the job with the smallest sum of processing times on all machines. Then the next job (*c*+1) in the sequence is the job *j* with the smallest value of $\delta_{j,c}$ calculated by equation 5.

$$\delta_{j,c}=\sum_{k=1}^{\text{m}}w_k\left(D_{[c+1],k}-D_{[c],k}-P_{j,k}\right) \tag{5}$$

Where w_k is defined by:

$$w_k=m/\left(k+c(m-k)/(n-2)\right) \tag{6}$$

The job *j* selected also affects the idle and blocking times of the remaining jobs in |NPS|. Therefore, Pan and Wang (2012) proposed the *PW* algorithm, which tries to minimize the idle and blocking times and the effects on the starting and completion times of later jobs. Equation 5 is used to estimate the idle and blocking times caused by the indexing of job *j* in position (*c*+1) of the sequence. Then equation 7 is used to estimate the idle and blocking times of the remaining jobs in |NPS|:

$$x_{j,c}=\sum_{k=1}^{m}w_k\left(D_{[c+2],k}-D_{[c+1],k}-P_{v,k}\right) \tag{7}$$

Where:

$P_{v,k}$ is the average processing time of all the remaining jobs in |NPS|, and is calculated by equation 10;

$D_{[c+2],k}$ is the departure time of $P_{v,k}$.

$$P_{v,k}=\sum_{\substack{q\in|NPS|\\ q\neq j}}P_{q,k}/\left(n-c-1\right) \tag{8}$$

Combining the estimated times of idle and blocking caused by jobs *j* and *v*, it is obtained $f_{j,c}$, calculated by equation 9. (*n*-*c*-2) is used to balance the idle and blocking times caused by job *j* and the effects of job *j* on later jobs. The first job in the sequence is the job *j* with the smallest value for $f_{j,0}$. Then, for the remaining positions of the sequence, the job *j* that obtains the smallest value for $f_{j,c}$ is the next job in the sequence (*c*+1).

$$f_{j,c} = (n - c - 2)\delta_{j,c} + x_{j,c} \tag{9}$$

3. COMPUTACIONAL RESULTS

This paper proposed programming the branch-and-bound algorithm using the lower bound proposed by (Ronconi, 2005). Then the best Phase I heuristic methods for Fm|block|Cmax problems were used in order to obtain and initial solution to the branch-and-bound algorithm. The heuristic methods used were: *MM* (Ronconi, 2004); *PF* (Mccormick *et al*., 1989); *PW* and *wPF* (Pan and Wang, 2012).

The tests were executed in an Intel® Core ™ i5-4460 processor with 3.2GHz, 8GB RAM and Windows 7 operational system. The branch-and-bound algorithm was applied to 180 problems proposed by Ronconi (2005), divided into 9 groups, all computed by MatLab software © (R2014b). The groups vary in number of jobs and machines, represented by matrices *nxm*: 10x2; 10x5; 10x10; 12x2; 12x5; 12x10; 14x2; 14x5; and 14x10. In this paper couldn't be analyzed the full database provided by Ronconi (2005) due to time requirement, however it is suggested as an extension of this work using the complete database with the 540 suggested problems by Ronconi (2005).

The results of the average relative deviation of the computational time and the number of nodes for all algorithms and problem classes were tabulated in Microsoft Excel software.

Table 1 - Average relative deviation of CPU time and number of nodes of all problem classes (continued).

Algorithms	10x2		10x5		10x10	
	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)
wPF	52340,36%	58607,50%	2,91%	0,87%	4,43%	2,50%
PF	50471,35%	58607,50%	1,91%	1,04%	3,29%	2,81%
PW	50582,37%	58607,50%	1,16%	0,09%	0,47%	0,38%
MM	4,53%	0,00%	1,63%	0,97%	2,80%	2,41%
Classic	50739,78%	58607,50%	2,18%	1,14%	3,68%	2,98%

Table 1 - Average relative deviation of CPU time and number of nodes of all problem classes (continued).

Algorithms	12x2		12x5		12x10	
	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)
wPF	6,92%	0,00%	1,15%	0,04%	1,80%	0,90%
PF	3,78%	0,00%	0,37%	0,06%	2,34%	1,83%
PW	3,10%	0,00%	0,43%	0,01%	1,32%	1,23%
MM	1,49%	0,00%	0,19%	0,07%	1,80%	1,51%
Classic	2,52%	0,00%	0,26%	0,07%	2,15%	1,86%

Table 1 - Average relative deviation of CPU time and number of nodes of all problem classes (continued).

Algorithms	14x2		14x5		14x10	
	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)	CPU Time (%)	Number of nodes (%)
wPF	3,78%	0,00%	1,26%	0,00%	7,72%	6,58%
PF	2,43%	0,00%	0,13%	0,02%	9,09%	8,36%
PW	3,27%	0%	0,25%	0,01%	0,16%	0,09%
MM	1,98%	0,00%	0,47%	0,02%	9,32%	8,48%
Classic	2,74%	0,00%	0,40%	0,02%	9,35%	8,50%

Table 2 - Average relative deviation of CPU time and number of nodes of the *Branch-and-Bound* algorithms.

Algorithms	CPU Time (%)	Number of nodes (%)
wPF	5818,93%	6513,15%
PF	5610,52%	6513,51%
PW	5621,39%	6512,15%
MM	2,69%	1,50%
Classic	5640,34%	6513,56%

Analyzing the data in Table 1 it can be noted that in the first problem class (with 10 jobs and 2 machines) the *MM* method associated with branch-and-bound outperformed all other methods greatly. The results obtained in this class

may distort the final result. Therefore another comparison was made without considering the first class of problems. The new comparison is presented in Table 3.

Table 3 – Mean relative deviation of CPU times and number of nodes of the Branch-and-Bound algorithms without considering the first class of problems

Algorithm	CPU time (%)	Number of nodes (%)
wPF	3,75%	1,36%
PF	2,92%	1,76%
PW	1,27%	0,23%
MM	2,46%	1,68%
Classic	2,91%	1,82%

4. CONCLUSIONS AND EXTENSIONS

This paper compared the use of different methods for obtaining an initial solution to the branch-and-bound algorithm in order to minimize the makespan in a flowshop environment with zero buffer.

The best heuristic methods in the literature were presented and these were applied in order to get an initial solution to the branch-and-bound algorithm. For comparison computer based tests were performed on a known database provided by Ronconi (2005). The experiments showed that the use of the branch-and-bound algorithm obtains a good performance for problems considered NP-Hard. In computational time term and number of nodes, the use of an initial solution obtained better results than the classic application of branch-and-bound in eight of the nine classes of the analyzed problems.

From the data presented in Table 2 it can be noted that the *MM* method associated with branch-and-bound had the best results among the compared methods. However, this result may have been influenced by the results obtained in the first class of problems (with 10 jobs and 2 machines), where the method resulted in really low computational time and number of nodes compared to the others. Hence another comparison was performed without considering the first class of problems, and, in this new scenario, *PW* associated with branch-and-bound provided the best results.

It can be noted from Table 1 that the *PW* method associated with branch-and-bound tends to provide better results as the number of machines increase. With fewer machines, the *MM* method associated with branch-and-bound seems to provide better results than *PW*. This can be explained by the quality of the results obtained by the *PW* algorithm, which is better than those provided by *MM* algorithm (Pan and Wang, 2012), however requiring more computational time. In some cases even when the number of nodes branched in smaller problems for the *MM* method associated with branch-and-bound is greater than or equal to the number of nodes branched for the *PW* method associated with branch-and-bound, the computational time of the *MM* method associated with branch-and-bound was smaller. This is due to the computational time required to calculate the lower bound for each node, which is smaller for fewer machines. So for problems with fewer machines the computational time required to solve the heuristic (which provides the initial solution) may affect more the computational time than the reduction of the number of nodes branched.

As the size of the problem seems to affect the results it is recommended that, the tests should be run in a more diversified database. It is recommended to use the entire database provided by Ronconi (2005) to analyze the results. As the quality and computational time of the initial solution affected the computational time of the branch-and-bound algorithm, it is recommended to use other heuristics (say phase II or III) associated with the branch-and-bound algorithm to compare the results.

The time required to calculate the lower bound on each node have a significant consequence to the computational time, therefore some other lower bound for the makespan can be tested for comparison. In addition, a dominance rule may be applied to reduce the number of nodes that need to be branched, analyzing the gains obtained in terms of computational time. The results obtained in this paper can be used in future works that study different lower bounds for flowshop with blocking problems, reducing the computational time of the branch-and-bound algorithms, hence reducing the computational time of the tests.

5. ACKNOWLEDGEMENTS

This work was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processes 308047/2014-1 and 448161/2014-1), University of São Paulo campus in São Carlos (USP) and Federal University of Technology - Paraná campus in Cornélio Procópio (UTFPR-CP).

6. REFERENCES

Companys, R.; Mateo, M. (2005). "Different Behavior of a Double Branch-and-Bound Algorithm on $F_m|prmu|C_{max}$ problems." *Computers & Operations Research* 34, 938-953.

Ronconi, D.P. (2004). "A note on Constructive Heuristics for the Flowshop Problem with Blocking."*International Jour-Nal of Production Economics* 87, 39-48.

Ronconi, D.P. (2005). "A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop with Blocking." *Annals of Operations Research* 138, 53-65.

Ronconi, D.P.; Armentano, V.A. (2001). "Lower Bounding Schemes for Flowshops with Blocking In-Process." *Journal of the Operational Research Society 52*, 1289-1297.

Framinan, J.M.; Gupta, J.N.D.; Leisten, R. (2004). "A Review and Classification of Heuristics for Permutation Flow-Shop Scheduling with Makespan Objective." *Journal of the Operational Research Society* 55,1243-1255.

Pinedo, M.L. (2008). "Scheduling Theory, Algorithms, and Systems." New York. Springer Science.3rd edition.

Kharbeche M. (2011). *Exact and Heuristic Methods for Variants of the Permutation Flow Shop Problems.*Ph D. thesis,University of Tunis, Tunis.

McCormick, S.T.; Pinedo, M.L.; Shenker, S.; Wolf, B. (1989). "Sequencing in an Assembly Line with Blocking to Minimize Cy." *Operations Research* 37, 925-935.

Pan, Q.; Wang, L. (2012). "Effective Heuristics for the Blocking Flowshop Scheduling Problem with Makespan Minimization." *Omega* 40, 218-229.

7. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.