Acta Scientiarum

http://www.uem.br/acta
ISSN printed: 1806-2563
ISSN on-line: 1807-8664
Doi: 10.4025/actascitechnol.v38i3.28438

Evaluation of heuristics for a branch and bound algorithm to minimize the makespan in a flowshop with blocking

Felipe Borreiro Sanches[1], Mauricio Iwama Takano[1] and Marcelo Seido Nagano[2*]

*[1]Universidade Tecnológica Federal do Paraná, Cornélio Procópio, Paraná, Brazil. [2]Universidade de São Paulo, Avenida do Trabalhador São-Carlense, 400, 13566-590, São Carlos, São Paulo, Brazil. *Author for correspondence. E-mail: drnagano@usp.br*

ABSTRACT. This paper has the objective to evaluate the use of different 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 long computational time to find the best solution. The use of an initial solution may reduce the computational time, by providing an initial upper bound. In this work, the efficiency of the use of an initial solution to the Branch and Bound algorithm was evaluated by comparison of the algorithms. The branch and bound algorithm used, as well as the lower bound, was proposed by Ronconi (2005). Four heuristic methods (MM, PF, wPF, and PW) were tested using a 180 problems data. Results show that the use of an initial solution does considerably reduce the computational time.

Keywords: Branch and Bound, heuristics, flowshop, block, makespan, scheduling.

Avaliação de heurísticas para um algoritmo branch e bound para minimizar o makespan em um flowshop com bloqueio

RESUMO. Este trabalho tem o objetivo de avaliar o uso de diferentes métodos heurísticos para obtenção de uma solução inicial para um algoritmo branch e bound, tendo como objetivo minimizar o makespan em um ambiente flowshop com buffer zero. Como o problema é conhecido por ser NP-Hard, o algoritmo branch e bound pode necessitar de elevado tempo computacional para encontrar a melhor solução. A utilização de uma solução inicial pode reduzir o tempo de processamento, proporcionando um limite superior inicial. Neste trabalho, a eficiência da utilização de uma solução inicial para o algoritmo branch e bound foi avaliada com a comparação dos algoritmos. O algoritmo branch e bound utilizado, bem como o limitante inferior foram propostos por Ronconi (2005). Cinco métodos heurísticos foram avaliados (MM, PF, wPF e PW) utilizando um conjunto de problemas teste composto de 180 problemas. Os resultados mostraram que a utilização da heurística para obter uma solução inicial para o algoritmo branch e bound reduz significativamente o tempo computacional.

Palavras-chave: Branch e Bound, heurísticas, flowshop, bloqueio, makespan, programação.

Introduction

Scheduling is a decision-making process that is used on a daily basis in many services and manufacturing in industry with the goal of optimizing one or more objectives, such as the makespan, number of delayed jobs, among others. It handles the allocation of resource operations (Pinedo, 2008; Nagano & Januário, 2012; Nagano, Silva, & Lorena, 2014; Sobreiro, Mariano, & Nagano, 2014; Nagano, Miyata, & Araújo, 2015; Sagawa & Nagano, 2015a; 2015b).

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 zero buffer constraint, there are no 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 $j+1$.

Branch and Bound algorithms can be used to solve scheduling problems in flowshop environment with blocking. However, the computational time required to solve big sized problems may be too long. So, an initial solution may be used to try to reduce the number of nodes explored, thereby reducing the total computational time.

The initial solution may be given by heuristics methods. When solving permutational flowshop

problems, the heuristic methods may consist of three phases, they are (Framinan; Gupta, & Leisten, 2004):

- Phase I –Index Development: in this phase, jobs are arranged according to a certain property based on the data of the problem;
- Phase II – Solution Construction: in this phase, a solution is constructed in a recursive manner, trying one or more unscheduled jobs to be inserted in one or more positions of a partial schedule until the schedule is complete;
- Phase III – Solution improvement: In this phase, an initial solution is improved by some procedure, usually a descending local search or a metaheuristic. This phase has two main characteristics: it requires an initial solution; the quality of the solution is always equal to or better than the initial solution.

A single heuristic method may consist of one or more of these phases. However, a heuristic consisting of more than one phase must perform them in the given order.

The use of an initial solution does not necessarily reduce the total computational time required by the Branch and Bound algorithm for some reasons: 1. The initial solution may not reduce the number of nodes explored. As the method used to provide an initial solution also requires some computational time, the total computational time required by the algorithm to solve the problem is increased; 2. The computational time required for the method used to provide an initial solution may be longer than, or equal to the computational time reduced by reducing the number of nodes explored.

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 & Mateo (2005); Ronconi (2005); Ronconi & Armentano (2001). 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.

McCormick, Pinedo, Shenker, and Wolf (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 sequence the one with the smallest sum of processing times in all machines. 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.

Ronconi & Armentano (2001) presented a Branch and Bound algorithm to minimize the total lateness in flowshop environments with blocking constraint. They proposed a lower bound for this criterion, and lower bounds for the makespan and the flow time. Ronconi & Armentano (2001) obtained satisfactory results, significantly reducing the number of nodes to be computed.

Ronconi (2004) analyzed the minimization of makespan of a problem in flowshop environment with blocking. An algorithm called MinMax (MM) was compared to the PF heuristic, considered the best algorithm proposed in the literature so far. The results showed that although the MM heuristic achieved good results, it was outperformed by the PF heuristic, which showed potential for major problems.

Companys & 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 in two steps: in the first step, a permutation is obtained, and in the second step, a local search procedure is applied. Companys & Mateo (2005) concluded that the computational time required to solve $Fm|block|Cmax$ problems is higher than for $Fm|prmu|Cmax$ problems. Companys & Mateo (2005) also presented a Lompen algorithm, which runs two algorithms simultaneously of Branch and Bound. They solved eight problems unsolved by other methods.

Ronconi (2005) developed an algorithm that exploits the occurrence of blocking in order to minimize the makespan time. Computational experiments proved that the proposed lower bound works better than the lower bound from Ronconi & Armentano (2001). As an extension of his work, Ronconi (2005) proposes the development of a dominance rule, some different node selection strategies, and the use of different methods to provide an initial solution for the Branch and Bound algorithm.

Pan & Wang (2012) adapted the PF algorithm (McCormick, Pinedo, Shenker, & Wolf, 1989), creating an improved form called 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. The results showed that both wPF and PW achieved better results than MM and PF. Also an insertion algorithm was applied, resulting in an improvement of 1.5% in CPU processing time.

Material and methods

Branch and Bound is a widely used method for solving difficult combinatorial optimization problems. Typically, the Branch and Bound method is characterized by two fundamental procedures (Rios-Mercado and Bard, 1999):

- Branching: 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 with 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.

For the scheduling problem, each node of the Branch and Bound corresponds to one sub-problem, which is defined by a subsequence of jobs. Each subsequence is called partial sequence (PS) and the set of jobs that are not in 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 (Ronconi, 2005). The lower bound is calculated for each node.

The Branch and Bound method can consume a high computational time when dealing with larger problems. One way to reduce the computational time is to implement an initial solution to the problem, which can result in a possible gain in terms of computational time by decreasing the number of explored nodes.

An initial solution provided by a phase I heuristic method was applied together with the Branch and Bound algorithm, as an initial upper bound for the problem, this may result in a reduced computational time. The initial upper bound may reduce the need to explore nodes, therefore reducing the number of nodes to be branched. However, the computational time required to generate an initial solution may be longer than the computational time required to solve the nodes that were not explored, thus resulting in an even longer computational time.

In this paper, we evaluated the efficiency of an initial solution to reduce computational time and which phase I heuristic method provides the best improvement to the computational time. The Branch and Bound algorithm as well as the lower bound were proposed by Ronconi (2005).

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 longer than or equal to LB2, according equations 1 and 2:

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

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

where:

m is the number of machines;

$D_{[PS],k}$ is the departure time of the last job in PS in machine *k*;

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

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

b_{k+1}^{g} is the *g*-th lowest 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*.

Initial Solution

The best phase I heuristics methods for $Fm|block|Cmax$ were used to provide an initial solution for the Branch and Bound algorithm. Those are MinMax (MM), Profile Fitting (PF), Weighted Profile Fitting (wPF) and PW.

MinMax (MM) algorithm, proposed by Ronconi (2004), initially sets the job with the shortest processing time on the first machine as the first job in the sequence, and then sets the job that has the shortest processing time on the last machine as the last job in the sequence. For the remaining jobs, the next job in the sequence (*c*) after the already scheduled 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.

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

The Profile Fitting (PF) algorithm, created by McCormick, Pinedo, Shenker, and Wolf (1989), works in two phases: 1. Sets the first job in the

sequence as the one with the smallest sum of processing times on all machines; 2. Then, the next position in the sequence ($c+1$) belongs to the job with the smallest sum of idle and blocking times. Equation 4 is used to calculate the possible sum of idle and blocking times provided by the insertion of each job that has not yet been sequenced in position ($c+1$) 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 equation 4.

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

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 & Wang (2012), applies a weight factor (w_k) to $\delta_{j,c}$, which differentiates the effects of idle and blocking times on machines in different stages and jobs in different positions. As in the *PF* algorithm, 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.

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

In which, w_k is defined by equation 6:

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

where:
n is the number of jobs;
c is the position of the last scheduled job.

The job j selected may also affect the idle and blocking times of the remaining jobs in |NPS|. Therefore, Pan & Wang (2012) proposed the PW algorithm, which seeks 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|:

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

where:

$P_{v,k}$ is the average processing time of all the remaining jobs in |NPS|, and is calculated by equation 8;
$D_{[c+2],k}$ is the departure time of $P_{v,k}$.

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

Combining the idle and blocking times caused by the insertion of jobs j in position ($c+1$) and the estimated idle and blocking times caused by the insertion of the artificial job v in position ($c+2$), it is obtained $f_{j,c}$, calculated by equation 9.

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

where:
($n - c - 2$) 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 scheduled as the next job in the sequence (c+1).

Results and discussion

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 an initial solution to the Branch and Bound algorithm. The heuristic methods used were: MM (Ronconi, 2004); PF (Mccormick, Pinedo, Shenker, and Wolf, 1989); PW and wPF (Pan & Wang, 2012). For the MM heuristic, the value of α is 0.6, which, according to Ronconi (2004), is the value in which the heuristic presents its best performance.

The tests were executed in an Intel® Core™ i5-4460 processor with 3.2 GHz, 8 GB RAM and Windows 7 operating system. The Branch and Bound algorithm was applied to 180 problems proposed by Ronconi (2005), divided into 9 groups, all computed by MatLab© (R2014b) software. The groups vary in number of jobs and machines, represented by matrices n x m: 10 x 2; 10 x 5; 10 x 10; 12 x 2; 12 x 5; 12 x 10; 14 x 2; 14 x 5; and 14 x 10. In this paper, the full database provided by Ronconi (2005) was not used due to the time required. However, as an extension of this, it is suggested using the complete database with the 540 suggested problems.

In order to evaluate the efficiency of using an initial solution to reduce computational time, it was compared the algorithms to find out which had the lowest computational time and the lowest number of nodes explored. To determine the best algorithm, it was used as a parameter the relative deviation, equation 10, which measures the relative distance of the solution found by this algorithm (DM) to the best result obtained (DM*) by all the algorithms.

$$DR = \frac{DM - DM^*}{DM^*} * 100 \quad (10)$$

In this paper, the mean relative deviations were used to compare computational times and number of nodes explored to find the best algorithm for each class. First, it was compared the mean relative deviation of the computational time (CPU time) of each class of problems, as shown in Table 1. Table 2 lists the mean relative deviation of the number of nodes explored for each class of problems.

Table 1. Mean relative deviation of CPU times of each problem class.

	Mean relative deviation of CPU time (%)				
	wPF	PF	PW	MM	Classic
10x2	52340.36	50471.35	50582.37	4.53	50739.78
10x5	2.91	1.91	1.16	1.63	2.18
10x10	4.43	3.29	0.47	2.80	3.68
12x2	6.92	3.78	3.10	1.49	2.52
12x5	1.15	0.37	0.43	0.19	0.26
12x10	1.80	2.34	1.32	1.80	2.15
14x2	3.78	2.43	3.27	1.98	2.74
14x5	1.26	0.13	0.25	0.47	0.40
14x10	7.72	9.09	0.16	9.32	9.35

Table 2. Mean relative deviation of number of nodes explored of each problem class.

	Mean relative deviation of Number of nodes (%)				
	wPF	PF	PW	MM	Classic
10x2	58607.5	58607.5	58607.5	0.00	58607.5
10x5	0.87	1.04	0.09	0.97	1.14
10x10	2.50	2.81	0.38	2.41	2.98
12x2	0.00	0.00	0.00	0.00	0.00
12x5	0.04	0.06	0.01	0.07	0.07
12x10	0.90	1.83	1.23	1.51	1.86
14x2	0.001	0.001	0	0.001	0.001
14x5	0.00	0.02	0.01	0.02	0.02
14x10	6.58	8.36	0.09	8.48	8.50

Table 3 presents the mean relative deviation of the computational time (CPU time) and number of nodes explored of all nine classes of problems to determine which, among the compared methods, had the shortest computational time and the minimum number of nodes explored.

Analyzing 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 greatly outperformed all other methods. 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 4.

Table 3. Mean relative deviation of CPU times and number of nodes explored for all problem classes.

Algorithm	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

Table 4. Mean relative deviation of CPU times and number of nodes explored for all problem classes 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

With data in Table 4, it is possible to observe that PW was the heuristics that reduced most the total computational time and the number of nodes explored by the Branch and Bound algorithm.

Analyzing Table 1, it can be verified that the PW method associated with Branch and Bound tends to provide better results as the number of machines increases. 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 that provided by MM algorithm (Pan & Wang, 2012), however requiring a longer computational time. In some cases, even when the number of nodes explored in smaller problems for the MM method associated with Branch and Bound algorithm is greater than or equal to the number of nodes explored for the PW method associated with Branch and Bound algorithm, the computational time of the MM method associated with Branch and Bound algorithm was shorter. This is due to the computational time required to calculate the lower bound for each node, which is shorter for fewer machines. In this way, for problems with fewer machines, the computational time required to solve the heuristics (which provides the initial solution) seems to affect more the total computational time than the quality of the initial solution itself.

The experiments showed that the use of the Branch and Bound algorithm achieves a good performance for problems considered NP-Hard. In terms of computational time and number of nodes

explored, the use of an initial solution obtained better results than the classic application of Branch and Bound in eight out of the nine classes of the analyzed problems.

Conclusion

The best heuristic methods in the literature were applied to get an initial solution to the Branch and Bound algorithm. Computational tests suggest that the use of an initial solution improves the results in terms of computational time to execute the algorithm. It is recommended to run the tests in a more diversified database. Other heuristics can be used with the Branch and Bound algorithm to compare the results. Some other lower bounds for the makespan can also be tested for comparison. Also, a dominance rule may be applied to reduce the number of nodes that need to be explored.

Acknowledgements

To the National Council for Scientific and Technological Development (CNPq): Processes: 308047/2014-1 and 448161/2014-1.

The authors are indebted to the University of São Paulo, campus São Carlos (USP) and the Federal University of Technological – Paraná State, campus Cornélio Procópio (UTFPR-CP) for their support for this paper.

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*(4), 938-953.

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*(12), 1243-1255.

Mccormick, S. T., Pinedo, M. L., Shenker, S., & Wolf, B. (1989). Sequencing in an assembly line with blocking to minimize cy. *Operations Research, 37*(6), 925-935.

Nagano, M. S., & Januário, J. C. S. S. (2012). Evolutionary heuristic for makespan minimization in no-idle flow shop production systems. *Acta Scientiarum. Technology, 35*(2), 271-278.

Nagano, M. S., Miyata, H. H., & Araújo, D. C. (2015). A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. *Journal of Manufacturing Systems, 36*(1), 224-230.

Nagano, M. S., Silva, A. A., & Lorena, L. A. N. (2014). An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times. *Expert Systems with Applications, 41*(8), 3628-3633.

Pan, Q., & Wang, L. (2012). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega, 40*(2), 218-229.

Pinedo, M. L. (2008). *Scheduling theory, algorithms, and systems* (3rd ed.). New York City, NK: Springer Science.

Rios-Mercado, R. Z., & Bard, J. F. (1999). A Branch-and-Bound Algorithm for Flowshop Scheduling with Setup Times. *IIE Transactions on Scheduling & Logistics*, *31*(8), 721-731.

Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking. *International Journal of Production Economics, 87*(1), 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*(1), 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*(11), 1289-1297.

Sagawa, J. K., & Nagano, M. S. (2015a). Modeling the dynamics of a multi-product manufacturing system: A real case application. *European Journal of Operational Research, 244*(2), 624-636.

Sagawa, J. K., & Nagano, M. S. (2015b). Applying bond graphs for modelling the manufacturing dynamics. *IFAC-Papers OnLine, 48*(3), 2047-2052.

Sobreiro, V. A., Mariano, E. B., & Nagano, M. S. (2014). Product mix: the approach of throughput per day. *Production Planning & Control, 25*(12), 1015-1027.

Received on July 2, 2015.
Accepted on October 6, 2015.