TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 66, Issue Special I, September, 2023

AN UPPER BOUND FOR THE BRANCH-AND-BOUND ALGORITHM FOR THE BLOCKING PERMUTATION FLOW SHOP WITH TOTAL TARDINESS CRITERION

Thiago KATO, Mauricio TAKANO, Marcelo NAGANO

Abstract: *The use of an upper bound for the branch and bound algorithm proposed by [1] is proposed in this paper. The problem explored is a blocking-in-process permutation flow shop problem with total tardiness criterion, which is known to be NP-Hard for m≥2. The literature for this theme is scarce, therefore this article aims to fill this gap. To improve the algorithm, it is proposed the use of an initial solution that will be used as an upper bound for the problem. A database that contains 27 different classes of problems was used for the computational experiments. Each class of problems varies in number of jobs (n) and in number of machines (m). To generate the initial solution, different constructive heuristics will be analyzed and compared to each other.*

Key words: *Scheduling, Permutation flow shop, Blocking, Total tardiness, Branch and bound, Upper Bound.*

1. INTRODUCTION

In the blocking permutation flow shop problem with total tardiness criterion, which is known to be NP-Hard for m≥2, a set of jobs must be processed on a sequence of machines in a specific order, subject to constraints on blocking and tardiness. According to [2] in the permutation flow shop problem each job must pass through all machines in the same order, with no overlap in processing times. Blocking occurs when a machine is occupied by a job and cannot process any other jobs that are waiting to be processed [3].

There are papers that considered unlimited buffers, [4], but this approach may not accurately reflect the reality of the industrial environment. When buffers are limited, a machine may become blocked. This article considers a zero-buffer constraint, meaning that if machine k completes the processing of job j and machine k+1 is not ready to receive the next job (because it is still processing task j-1 or has not yet been set up), the job will remain on machine k, blocking it. In this situation, machine k cannot receive the next job in the sequence.

[3] were one of the first to study the block in process problem, they developed an algorithm, known as Profile Fitting (PF), which is a constructive heuristic that refers to a job insertion technique in which an unscheduled job is added to a partial sequence, with the objective of minimizing both machine idle and blocking times. This method performed satisfactorily, because in the five tests conducted with two PF have good results for the developed tests.

[5] proposed a heuristic called MaxMin (MM), based on the properties of the makespan presented by [4] where the minimization of makespan for a flow shop with blocking was analyzed. The MM heuristic showed satisfactory results, but outperformed PF, demonstrating great potential to solve large sized problems.[6], proposed a constructive heuristic named an algorithm named Weighted Profile Fitting (wPF) and PW (an improved form of algorithmic PF), which follows the same rules as the PF method, however considering a relative weight to the sum of idle and blocking times, which differentiates the effect of machines in different stages and jobs in different positions.

Both wPF and PW produced much better results than existing PF heuristics.

The blocking flow shop for total tardiness problem is a relatively unexplored area of research, [7] proposed a constructive heuristic for the total tardiness minimization in a flowshop with blocking problem, named Fitting Processing times and Due dates (FPD). This Heuristic generates a list of priorities among the jobs, with the objective to minimize overall delays. In a comparison of the FPD and LB, the proposed FPD algorithm presented an average improvement of 11.11%.

[1], proposed a lower bound for the branch and bound algorithm to solve the blocking permutation flow shop scheduling problem with total tardiness criterion. The proposed *LB* was compared to two other lower bounds proposed by [8] and [9].

However, the utilization of an upper bound for the branch and bound problem, have an intention to improve the efficiency of the algorithm.

This paper aims to fill a gap in the literature on the topic, as few studies have addressed this issue. This will allow us to identify which method is most suitable for generating an initial solution that can serve as an upper bound for the problem. This initial solution will then be used to improve the performance of the branch-and-bound algorithm.

2. PROBLEM DEFINITION

The proposed method aims to minimize the total tardiness, which is the sum of the cumulative tardiness of a set of tasks. To calculate the total tardiness, it is needed to find the tardiness time for each job. For this, the departure time for each job on the last machine must be calculated. Let σ = {1, 2, ..., i, j, ..., n} be any sequence of jobs, k = {1, 2, ..., m} be a sequence of available machines, $P_{jk}$is the processing time of job j on machine k, and D_{jk} is the time when job j departs from machine k. Equations 1-5 show how the departure time is calculated.

$$D_{10} = 0 \quad (1)$$

$$D_{1k} = \sum_{q=1}^{K} P_{1k} \;\; \forall 1 \leq k \leq m-1 \quad (2)$$

$$D_{j0} = D_{j-1,1} \;\; \forall 2 \leq j \leq n \quad (3)$$

$$D_{jk} = \left(D_{j,k-1} + P_{jk}, D_{j-1,k+1}\right) \;\; \forall 2 \leq j \leq n; 1 \leq k \leq m-1 \quad (4)$$

$$D_{jm} = D_{j,m-1} + P_{jm} \;\; \forall 1 \leq j \leq n \quad (5)$$

Initially, Expression 1 is used to define the start time of the first job in the sequence (D10). Then, Equation 2 is applied to calculate the departure time of the first job on all machines. The start time for the next job in the sequence is calculated using Equation 3, and the departure time for all machines but the last one is calculated using Equation 4. For the last machine, the departure time of the job is determined by Equation 5. Equations 3, 4, and 5 are used again to calculate the start and departure times of the next job in the sequence. This process is repeated until all departure times are calculated.

Tardiness is a function of the due dates, and the total tardiness is the sum of the tardiness of all jobs.

Therefore, to calculate the total tardiness (TT), Equation 6 is used.

$$TT = \sum_{j=1}^{n} \max\left(D_{jm} - d_j, 0\right) \quad (6)$$

3. THE BRANCH-AND-BOUND ALGORITHM

The branch-and-bound algorithm consists of branching high complexity problems into simplified problems, or sub-problems, called nodes. In the algorithm proposed by [10] each of these nodes are represented by a Partial Sequence |PS|. The set of jobs that are not part of the |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 choice of the node to be branched was made using the depth-first method proposed by [11]. This means that the node with the highest number of jobs in the Partial Sequence |PS| will be branched. In case of ties the algorithm selects the node with the smallest lower bound.

3.1 Lower bound

The lower bound used for the branch-and-bound algorithm was proposed by [1]. The proposed lower bound assumes a relaxation in which one

machine can process one job at a time and the others can handle the whole set of n jobs at a time. Since the SPT rule is optimal for total flow time criterion on a single machine, it is possible to obtain a lower bound for the flow shop problem. The lower bound considering a partial sequence σ of size s is computed as follows.

3.2 Upper bound

To improve the Branch-and-Bound algorithm it was proposed the use of an upper bound. This can reduce the number of nodes that must be branched, as it will prune every node whose lower bound is greater than or equal to the upper bound value.

This upper bound was calculated based on the best available heuristics in the literature. Used the heuristic that showed good results for Fm|prm,block|Tj problems and other constructive heuristics for Fm|prm,block|Cmax problems. Those are the Fitting Processing times and Due date (FPD), MinMax (MM), Profile Fitting (PF), Weighted Profile Fitting (wPF) and PW.

The Fitting Processing times and Due date (FPD) heuristic proposed by [7] generates a dynamic list of priorities among the jobs. When a job is selected for a position in the sequence, the list is reorganised with the jobs that have not yet been added to the sequence. Initially choose the task with the lowest sum of due dates and processing time. For the next positions, it uses a priority measure F_k (equation 7) composed of two rules: 1. (fit_k) for the jobs that have processing times near the "windows" generated by the last fixed task in the sequence on each machine; 2. ($dynslack_k$) aims at increasing the priority of jobs that have a short remaining period of time to be processed before their due date

$$F_k = \rho\, fit'_k + (1-\rho) dynslack'_k \quad (7)$$

$$fit_k = \sum_{j=1}^{m-1} \left| b_j - P_{kj} \right| \quad (8)$$

$$dynslack_k = LB_k - D_{ti,1} \quad (9)$$

The MinMax (MM) heuristic proposed by [5], uses three criteria: 1. it defines the first job of the sequence as the one with the shortest processing time on the first machine; 2. it defines the last job of the sequence as the one with the shortest processing time on the last machine; 3. for the remaining jobs, the next job of the sequence is the one that gets the smallest result in equation 10.

$$\alpha \sum_{l=1}^{m-1} \left| P_{[j],l} - P_{[i],l+1} \right| + (1-\alpha) \sum_{k=1}^{m} P_{[j],k} \quad (10)$$

Where:

α - constant used to weight the two terms of the expression.

The Profile Fitting (PF) proposed by [3] tries to identify and sequence jobs to reduce machine blocking times. To use the Profile Fitting (PF) method, two rules are used: 1. Define as the first job the one that presents the lowest sum of the processing times on all machines; 2. Then, equation 11 is used to calculate the possible idle and blocking times provided by the insertion of each job not yet sequenced in position c of the sequence. The job (j) that obtains the smallest value for δj,c is determined as the next job (c+1) in the sequence.

$$\delta_{j,c} = \sum_{k=1}^{m} \left(D_{[c+1],k} - D_{[c],k} - P_{j,k} \right) \quad (11)$$

The Weighted Profile Fitting (wPF) proposed by [6] works similarly to PF, however it defines a different weight (w_k) for each machine and position. 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 12

$$\delta_{j,c} = \sum_{k=1}^{m} w_k \left(D_{[c+1],k} - D_{[c],k} - P_{j,k} \right) \quad (12)$$

Where wk is defined by equation 13:

$$w_k = \frac{m}{\left(\frac{k + c(m-k)}{(n-2)} \right)} \quad (13)$$

The PW heuristic was proposed by [6], and it tries to minimize the idle and blocking times, in addition to the effects on the start and completion times of subsequent jobs. The selected job j also impacts the idle and blocking times of other jobs in |NPS|. Equation 12 is used to estimate the idle and blocking times caused by indexing job j at position (c+1) of the sequence.

Then, Equation 14 is used to estimate the idle and blocking times of the other jobs in |NPS|.

$$x_{j,c} = \sum_{k=1}^{m} w_k\left(D_{[c+2],k} - D_{[c+1],k} - P_{v,k}\right) \quad (14)$$

Where:

$P_{v,k}$ is the average processing time of all the remaining jobs in |NPS|.

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

$$P_{v,k} = \sum_{\substack{q\in|NPS| \\ q\neq j}} P_{q,k} \Big/ (n - c - 1) \quad (15)$$

By combining the estimated idle and blocking times caused by jobs j and v, it can be obtained $f_{j,c}$, which is calculated using equation 16. The term (n-c-2) is used to balance the idle and blocking times caused by job j and its effects on subsequent jobs. The initial job in the sequence corresponds to the job j with the smallest value for $f_{j,0}$. For the remaining positions in the sequence, the job j that achieves the minimum value for $f_{j,c}$ becomes the subsequent job in the sequence (c+1).

$$f_{j,c} = (n - c - 2)\delta_{j,c} + x_{j,c} \quad (16)$$

4. COMPUTACIONAL RESULTS

This paper proposes programming the branch-and-bound algorithm using the lower bound proposed by [1]. Then, the only heuristic found in the literature for the Fm|prm,block|somaTj and the best heuristic methods for the Fm|block|Cmax problems were used in order to obtain an initial solution for the branch-and-bound algorithm. The heuristic methods used were: FPD [7]; MM [5]; PF [3]; PW and wPF [6].

For the processing times, the database proposed by [9] was used. In this database the processing times were uniformly distributed between 1 and 99. The database consists of 27 different problem sizes, with 20 unique problems each,totaling 540 instances.

The due dates were generated as proposed by [12]. i.e., the due dates of each job were uniformly distributed between P(1-T-R/2) and P(1-T+R/2), where T is the tardiness factor of the jobs, R is the dispersion range of the due date, and P is a lower bound of the flow shop with unlimited buffer.

$$P = \max(\, max_{1\le k\le m} \sum_{j=1}^{n} P_{jk} + min_{1\le j\le n} \sum_{q=1}^{k-1} P_{jq} + min_{1\le j\le n} \sum_{q=k+1}^{m} P_{jq}; max_{1\le j\le n} \sum_{k}^{m} P_{jk} \,) \quad (17)$$

These scenarios represent different configurations by varying T and R, as follows:

- Scenario 1: low Tardiness factor (T = 0.2) and small due date Range (R =0.6);
- Scenario 2: low Tardiness factor (T = 0.2) and wide due date Range (R =1.2);
- Scenario 3: high Tardiness factor (T = 0.4) and small due date Range (R =0.6);
- Scenario 4: high Tardiness factor (T = 0.4) and wide due date Range (R =1.2).

The tests were performed on an Intel® Core TM i5-9300H processor with 2.40 GHz, 8 GB of RAM DDR 4 and Windows 11 operating system and coded in MatLab© software (R2023a). The computational times were obtained through the function tic and toc. A time limit of 3600 seconds was established for the execution of the algorithm.

Only the first 12 were tested due to the high computational time and limited time to execute them.

A comparison was made between the Branch-and-Bound algorithm without the use of an initial solution (B&B) and the same algorithm using different heuristics to provide an initial solution ($B\&B_{FPD}$, $B\&B_{MM}$, $B\&B_{PF}$, $B\&B_{wPF}$ and $B\&B_{PW}$).

To compare the performance of the algorithms, the relative percent deviation (RPD) was calculated for the average number of nodes created and the average computational times, using equation 18:

$$RPD_{Variation} = \frac{\sigma_{\,variation} - \sigma^{*}}{\sigma^{*}} * 100 \quad (18)$$

Where:

$\sigma_{\,variation}$ is the average value of computational time or the average number of nodes created obtained with the analyzed B&B variation;

σ^{*} is the best value obtained among all B&B variations.

All algorithms were run in the same computational conditions. The RPD was calculated for all databases, and the results are shown in table 1 the average number of nodes created and table 2 the average time spent.

Table 1

RPD of the average node count for all scenarios

Scenarios	B&B	B&B$_{FPD}$	B&B$_{MM}$
Scenarios 1	0.0164448	**0**	0.0163875
Scenarios 2	2.0398042	**0**	1.9980851
Scenarios 3	0.0243709	0.0234177	0.0220784
Scenarios 4	0.0133636	0.01123577	0.0128441
Average	0.52349592	**0.00866337**	0.51234881
Scenarios	**B&B$_{PF}$**	**B&B$_{wPF}$**	**B&B$_{PW}$**
Scenarios 1	0.0146737	0.0146940	0.0134706
Scenarios 2	2.0388168	2.0391286	1.9592651
Scenarios 3	0.0007495	0.0084349	**0**
Scenarios 4	0.0083201	0.0080275	**0**
Average	0.51564007	0.51757127	0.49318395

Table 2

RPD of the average CPU time for all scenarios

Scenarios	B&B	B&B$_{FPD}$	B&B$_{MM}$
Scenarios 1	**0**	0.0233367	0.0083939
Scenarios 2	1.5625408	**0**	1.5498333
Scenarios 3	0.0149708	0.0461302	0.0142837
Scenarios 4	**0**	0.0292762	0.0042258
Average	0.39437793	**0.02468579**	0.39418423
Scenarios	**B&B$_{PF}$**	**B&B$_{wPF}$**	**B&B$_{PW}$**
Scenarios 1	0.0061358	0.0305527	0.0301620
Scenarios 2	1.5934854	1.6849868	1.5928504
Scenarios 3	**0**	0.0306932	0.0229659
Scenarios 4	0.0051602	0.0318383	0.0267223
Average	0.40119539	0.44451778	0.41817517

From table 1 it is possible to analyze that the B&B$_{FPD}$ heuristic got better results for scenarios 1 and 2, and the B&B$_{PW}$ heuristic for scenarios 3 and 4. However the B&B$_{FPD}$ in these scenarios got very close results to B&B$_{PW}$, this contributed to B&B$_{FPD}$ having a better average node count.

From table 2 it is possible to notice that the B&B$_{FPD}$ method presented the best average computational time compared to the others.

Comparing tables 1 and 2 it is possible to notice that the number of nodes does not always mean that the computational time spent to solve the problem is lower. It can be true due to the computational time spent to solve the heuristic that provides the initial solution.

5. CONCLUSION

This paper considers a blocking permutation flow shop problem with the total tardiness criterion, which is known to be NP-hard for m≥2. The use of efficient heuristics as initial upper bounds for the branch and bound algorithm is proposed. Four due date databases were generated, each with different values ranges and tardiness factors.

For the initial solution the best heuristics found in the literature were used, they are: FPD [7]; MM [5]; PF [3]; PW and wPF [6]. The FPD, which is the only heuristic found in the literature that was designed for the blocking permutation flow shop problem with total tardiness criterion. The B&B$_{FPD}$ outperformed all other algorithms only in the second scenario.

Although the FPD heuristic requires more computational time to find a solution, it is able to reduce the number of nodes significantly. Therefore, for larger problems it will probably reduce the computational time required to solve each problem, as the time demanded to solve the FPD method will be compensated by the reduction in the number of nodes.

For future works, it is recommended to run the entire database to evaluate the performance of the FPD heuristic as an initial solution for larger problems. Also, it is suggested to develop a dominance rule for pruning a higher number of nodes. These suggestions aim to confirm the efficiency and improve the algorithm. Furthermore, other exploration rules may be tested, such as the best bound rule, or a hybrid technique.

6. ACKNOWLEDGMENTS

This work was partially supported by Pró-Reitoria de Graduação UTFPR (PROGRAD), Directorate of Research and Postgraduate Studies UTFPR-CP (DIRPPG), Directorate of Graduate and Professional Education UTFPR-CP (DIRGRAD) and Mechanical Engineering Department UTFPR-CP (DAMEC).

7. REFERENCES

[1] Nagano, M. S., Takano, M. I., & Robazzi, J. V. S. *A branch-and-bound for the blocking*

permutation flowshop with total tardiness criterion, International Journal of Industrial Engineering Computations, 2022.

[2] Garey, M. R., Johnson, D. S., & Sethi, R. *The complexity of flowshop and jobshop scheduling*, Mathematics of Operations Research, 1976.

[3] Mccormick, S., Pinedo, M., J. Shenker, S., & Wolf, B. *Sequencing in an assembly line with blocking to minimize cycle time*, Operations Research 37(6), 925-935 ,1989.

[4] Nawaz, M., Enscore, E. E., Ham, I. *Heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, Omega, 1983.

[5] Ronconi, D. P. *A note on constructive heuristics for the flowshop problem with blocking*, International Journal of Production Economics, 2004.

[5] Quan-Ke Pan,Ling Wang. *Effective heuristics for the blocking flowshop scheduling problem with makespan minimization*, Omega, 2012.

[6] Ronconi, D. P. e Henriques, L. R. S. *Some heuristic algorithms for total tardiness minimization in a flowshop with blocking*, Omega, 2009.

[7] Ronconi, D. P. e Armentano, V. A. *Lower bounding schemes for flowshops with blocking in-process*, The Journal of the Operational Research Society, 2001.

[8] Said, T. et al. *Branch and bound algorithm for solv-ing blocking flowshop scheduling problem with total tardiness and total weighted tardiness criteria*, International Journal of Operational Research, 2017.

[9] Kim,Y.D. Minimizing mean tardiness inpermutation flowshops, European Journal of Operational Research, v. 85, n. 3, p. 541-555, 1995.

[10] Ronconi, D. P. *A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking*, Annals of Operations Research, 2005.

[11] Taillard, E. *Benchmarks for Basic Scheduling Problems*, European Journal of Operational Research, 1993.

O LIMITĂ SUPERIOARĂ PENTRU ALGORITMUL BRANCH-AND-BOUND PENTRU FLUXUL DE PRODUCȚIE CU PERMUTĂRI BLOCAT CU CRITERIU DE ÎNTÂRZIERE TOTALĂ

Utilizarea unei limite superioare pentru algoritmul branch-and-bound propus de [1] este propusă în această lucrare. Problema explorată este o problemă de blocare-în-proces a fluxului de producție cu permutare cu criteriu de întârziere totală, despre care se știe că este NP-Hard pentru m≥2. Literatura pentru această temă este limitată, prin urmare acest articol își propune să umple acest gol. Pentru a îmbunătăți algoritmul, se propune utilizarea unei soluții inițiale care va fi folosită ca limită superioară a problemei. Pentru experimentele de calcul a fost utilizată o bază de date care conține 27 de clase diferite de probleme. Fiecare clasă de probleme variază ca număr de locuri de muncă (n) și ca număr de mașini (m). Pentru a genera soluția inițială, se vor analiza diferite euristici constructive și se vor compara între ele.

Thiago Keizo KATO, graduating in electrical engineering, Student, Federal University of Technology-Paraná-Brazil, Electrical Engineering Department, thiago.kato@alunos.utfpr.edu.br, +55 14 991117706, Rua Dos Andradas, 421, apto 104 - Centro, Cornélio Procópio – PR – Brazil., +55 14 991117706.

Mauricio Iwama TAKANO, PhD, Professor, Federal University of Technology-Paraná-Brazil, Mechanical Engineering Department, takano@utfpr.edu.br, +55 43 3133-3912, Av. Alberto Carazzai, 1640 - Centro, Cornélio Procópio – PR – Brazil., +55 43 996536741.

Marcelo Seido NAGANO, PhD in Mechanical Engineering, Professor, São Carlos School of Engineering - University of São Paulo-Brazil, Production Engineering, drnagano@usp.br, +55 16 33739428, 400 Trabalhador Sãocarlense Avenue, São Carlos/SP, Zip Code: 13566-590, Brazil, +55 16 991717439.