CONSIDERATIONS RELATED TO THE LOWER BOUND, IN FLOWSHOP SCHEDULING PROBLEMS

Cristina-Elena DODU[1] and Mircea ANCĂU[2]
[1] *Universitatea Tehnică din Cluj-Napoca, B-dul Muncii 103-105, 400641 Cluj-Napoca, Romania*
E-mail: cristina.dodu@tcm.utcluj.ro
[2] *Universitatea Tehnică din Cluj-Napoca, B-dul Muncii 103-105, 400641 Cluj-Napoca, Romania*
E-mail:mircea.ancau@tcm.utcluj.ro

***ABSTRACT**: This paper presents some theoretical considerations necessary to determine the lower bound for permutation flowshop scheduling problems. There are analyzed two different situations corresponding at classical problem and respective to batch size production problem. It is shown, in pseudo-code, the programs to calculate the lower bound in both situations. The paper originality came from the fact that there are corrected some of the lower bound corresponding to the existing sets in the literature. There is also explained the cause of these errors.*

***KEYWORDS**: flowshop scheduling, benchmarks, lower bound, batch size.*

1 INTRODUCTION

The classical flowshop scheduling problem (FSP), refers to the total time optimization of the manufacturing succession of *n* jobs that must be processed on *m* machines, all jobs having the same technological itinerary (i.e. each job route to the machines is the same). At this problem some initial assumptions must be respected: the jobs processing pursue the same succession on all machines, once a job manufacturing on a machine is started, it must be finished an so on. Because the manufacturing succession may start with any of *n* jobs, the problem is called permutation flowshop scheduling (PFSP). A complete list of initial assumptions of FSP or PFSP may be found in (Gupta & Stafford, 2006).

Having multiple practical applications, PFSP is an intense studied subject by several researchers. This aspect is illustrated by numerous papers that summarize the up to date results in this field (Framinan, Gupta & Leisten, 2004), (Hejazi & Saghafian, 2005), (Pan & Ruiz, 2013) and so on.

Each time in the literature, a new algorithm that solve FSP or PFSP is proposed, its performances are compared with the best already existing algorithms in the field. In the literature can be habitually found comparative studies of several algorithms, but only concerning the quality of results, or concerning the CPU time, but not concerning both criteria (i.e. results quality and CPU time). On the other hand, is problematic to compare the speed of some algorithms, while they are executed on different platforms, using different processors. Nevertheless, the quality of results is always easy to compare, relative to the same set of test problems.

Starting in 1993 when E. Taillard published in European Journal of Operational Research the paper Benchmarks for basic scheduling problems, countless researchers utilized these problems for testing their research results. In the paper there are proposed 260 test problems for flowshop, jobshop and openshop scheduling. Numerous researches linked to this field make use to these sets of test problems (Liu, 1999), (Abedinnia, Glock & Brill, 2016), (Fernandez-Viagas, Molina-Pariente & Framinan, 2019). The present research refers to the test set of flowshop scheduling problems (Taillard, 2020).

The paper is organized as follows: the chapter no.2 analyses the calculation of *lower bound* in two cases. First case refers to classical problem, where one must find optimal manufacturing sequence of *n* jobs on *m* machines. In this case, to each job corresponds a single part, different from any others. In the second case, the problem of obtaining optimal manufacturing arrangement, refers also to *n* jobs on *m* machines, this time to each job correspond a batch of *s* identical pieces. In both cases, the method to calculate the *lower bound* is investigated. The paper ends with a chapter of concluding remarks and the list of references.

2 THE LOWER BOUND CALCULATION

In (Taillard, 1993), the algorithm for random generation of values corresponding to elements of manufacturing time matrix, for each job on *m* machines, is detailed. All the manufacturing time values from interval [1, 99] have the same probability of being generated. There are proposed 12 groups of (nxm) problems (i.e. *n* jobs on *m* machines), as follows: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20 and 500x20, each group type including 10 problems. Furthermore, for each problem in a group the value of the *lower bound* (LB) and *upper bound* (UB) are indicated. LB represents an estimated value of the optimal solution, while UB is the best solution found for that problem until present.

To calculate the lower bound, all the awaiting times that might occur into a specific succession of jobs, are neglected. Basically, all machines are working without pause, so giving rise to an ideal arrangement of jobs. The only variation that might emerge from one succession of jobs to another are due to the *time before* and the *time after* manufacturing on each machine. These values of time are due to the technological itinerary, which is the same for each job. Therefore, at the first job of a specific sequence of jobs, the machine m_i cannot start work until the job is not finished on machine m_{i-1}. In the same way, at the last job in the manufacturing sequence, when the machine m_i finished the work, there is necessary an amount of time so that machines m_{i+1}, m_{i+2}, …, m_m must finish to work.

2.1 The case of classical problem

The case of classical PFSP refers to the arrangements of *n* jobs on *m* machines, that obey the initial assumption specified above. Each job within the *n* jobs corresponds to a single part, different from other.

To solve this kind of problem, means to find that arrangement of jobs that corresponds to the minimum manufacturing time. It is well known that such type of problem is NP-hard (Fernandez-Viagas, Dios & Framinan, 2016).

In this context, let us consider the matrix of manufacturing time:

$$T_{m,n} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ t_{m1} & t_{m2} & \dots & t_{mn} \end{bmatrix}; \qquad (1)$$

According to (Taillard, 1993), there are noted down as b_i, the work time done before machine mi work start at first job in a specific jobs sequence. There are also noted down by a_i, the work time remained to be completed, after machine m_i finalized the last job from the specific job sequence. In this case:

$$b_i = \min_{1 \le j \le n} \left(\sum_{k=1}^{i-1} t_{kj} \right); \qquad (2)$$

and

$$a_i = \min_{1 \le j \le n} \left(\sum_{k=i+1}^{m} t_{kj} \right); \qquad (3)$$

The sum of manufacturing time of all *n* jobs on machine m_i is:

$$T_i = \sum_{j=1}^{n} t_{ij}; \qquad (4)$$

In Fig.1, can be seen the Gantt diagram corresponding to the case of two jobs manufactured on six machines. There are also indicated the *time before* b_i and *time after* a_i, on each machine.

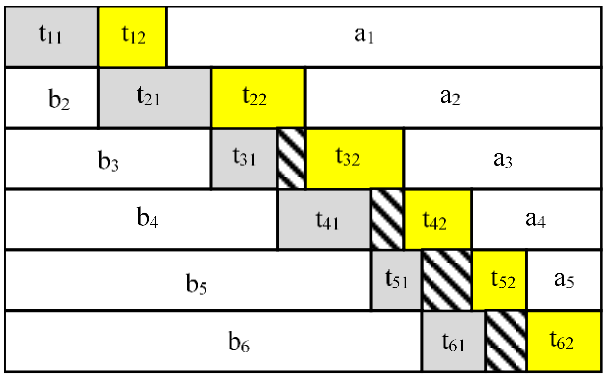

Fig. 1 The diagram for *time before* b_i and *time after* a_i.

Each one of those *n* jobs can be start job or final job in a specific manufacturing sequence. Depending on the start job and final job, the *time after* and the *time before* can take different values. To calculate the *lower bound* value, on machine m_i will be considered the lowest value of b_i and a_i. The sum of manufacturing time for all *n* jobs on machine m_i is a constant (see eq.4). Taking into

account the equations (2), (3) and (4), the *lower bound* can be calculated as:

$$LB = \max_{1 \le i \le n} (b_i + T_i + a_i); \quad (5)$$

In equations (2) and (3), $b_1 = 0$, $a_n = 0$. Let us examine the case of a scheduling problem with 5 jobs on 6 machines (see Fig. 2).

$$time = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} & t_{15} \\ t_{21} & t_{22} & t_{23} & t_{24} & t_{25} \\ t_{31} & t_{32} & t_{33} & t_{34} & t_{35} \\ t_{41} & t_{42} & t_{43} & t_{44} & t_{45} \\ t_{51} & t_{52} & t_{53} & t_{54} & t_{55} \\ t_{61} & t_{62} & t_{63} & t_{64} & t_{65} \end{bmatrix};$$

Fig. 2 Case $b_1 = t_{11}+t_{21}$; $a_5 = t_{45}+t_{55}+t_{65}$.

If the manufacturing sequence is ($j_1j_2j_3j_4j_5$), then $b_1 = (t_{11}+t_{21})$, and $a_5 = (t_{45}+t_{55}+t_{65})$. Every job can start the manufacturing sequence, while every job can be the final one. In Fig. 3, for instance, the manufacturing sequence starts with job j_2 and ends with job j_4. This is the reason why in eq. 2 and eq.3, we take the smallest possible value for b_i and a_i. The sum of manufacturing time of all jobs on each machine is constant (see eq.4).

$$time = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} & t_{15} \\ t_{21} & t_{22} & t_{23} & t_{24} & t_{25} \\ t_{31} & t_{32} & t_{33} & t_{34} & t_{35} \\ t_{41} & t_{42} & t_{43} & t_{44} & t_{45} \\ t_{51} & t_{52} & t_{53} & t_{54} & t_{55} \\ t_{61} & t_{62} & t_{63} & t_{64} & t_{65} \end{bmatrix};$$

Fig. 3 Case $b_2 = t_{12}+t_{22}$; $a_4 = t_{44}+t_{54}+t_{64}$.

When the lower bound is calculated using eq. 2 and eq. 3 from (Taillard, 1993), it must be remarked that *time before* and *time after* (i.e. b_i and a_i) cannot correspond to the same job. The *time before* corresponds to the first job in manufacturing sequence, while *time after* corresponds to the last job in the same sequence. For this reason, the values of bi and ai are depending of the order they are calculated. Therefore, it is possible to obtain two different values for *lower bound*, and in this case, we will take the smallest one.

In the absence of this remark, a_i and b_i does not depend on the calculation order and a single value for *lower bound* is obtained. But the remark cannot be ignored because a_i and b_i cannot correspond to the same job. So, the situation exemplified in Fig.4 isn't possible because a manufacturing sequence cannot be finalized with the start job.

$$time = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} & t_{15} \\ t_{21} & t_{22} & t_{23} & t_{24} & t_{25} \\ t_{31} & t_{32} & t_{33} & t_{34} & t_{35} \\ t_{41} & t_{42} & t_{43} & t_{44} & t_{45} \\ t_{51} & t_{52} & t_{53} & t_{54} & t_{55} \\ t_{61} & t_{62} & t_{63} & t_{64} & t_{65} \end{bmatrix};$$

Fig. 4 The impossible case: b_i and a_i correspond to the same job.

Table 1 shows the values of lower bound, time before and time after for the set of 20x5 (i.e. 20 jobs on 5 machines). We must remark that at 7th problem, the values for b_i and a_i correspond to the same job j_{10}, situation that is practically impossible. So, for this problem, if we calculate first time the value for *time before* and then the value for *time after*, will get LB = 1234. Contrary, if we calculate first time the value for *time after* and then the value *for time before*, will get LB = 1251. From these two values, we will take the smallest one, that is LB = 1234, which will replace the actual LB = 1226.

Table 1. The *lower bound* values for 20x5 set.

Nr.crt.	m_i	min b_i	min a_i	$\sum t_{ij}$	LB
1	m_1	b_1=0	a_3=111	1121	1232
2	m_5	b_{14}=83	a_1=0	1207	1290
3	m_5	b_3=106	a_1=0	967	1073
4	m_2	b_{13}=9	a_{18}=82	1177	1268
5	m_1	b_1=0	a_5=91	1107	1198
6	m_4	b_{11}=51	a_3=7	1122	1180
7	m_4	**b_{10}**=66	**a_{10}**=8	1152	1226
8	m_3	b_9=37	a_1=36	1097	1170
9	m_5	b_4=68	a_1=0	1138	1206
10	m_4	b_5=63	a_4=10	1009	1082

Table 2. The *lower bound* values for 20x10 set.

Nr.crt.	m_i	min b_i	min a_i	$\sum t_{ij}$	LB
1	m_9	b_2=267	a_{16}=3	1178	1448
2	m_7	b_9=220	a_{18}=82	1177	1479
3	m_5	b_4=68	a_8=122	1217	1407
4	m_7	b_{18}=187	a_{17}=50	1071	1308
5	m_8	b_{16}=134	a_5=37	1154	1325
6	m_7	b_{18}=168	a_1=47	1075	1290
7	m_9	b_{19}=186	a_6=8	1194	1388
8	m_3	**b_{10}**=37	**a_{10}**=218	1108	1363
9	m_9	b_{14}=218	a_9=3	1251	1472
10	m_4	b_{19}=95	a_{12}=103	1158	1356

In a similar manner, Table 2 shows the *lower bound*, *time before* and *time after* values

corresponding to the set of 20x10 problems (i.e. 20 jobs on 10 machines). In this set, we can see again that at problem 8th, the *time before* and *time after* takes minimum values at the same job j_{10}. While a manufacturing sequence cannot start and end with the same job, b_i and a_i must be calculated in two distinct ways. If the calculation starts with b_i, followed by a_i, and taking into consideration that these quantities cannot correspond to the same job, we get LB = 1415. Contrary, if the calculation starts with a_i, followed by b_i, we get LB = 1364. So, we take as *lower bound* the value LB = 1364, instead of 1363.

A program in Matlab was designed to find the true value of lower bound. This program is presented in pseudo-code in Fig. 5. In Matlab, assign refers to the memory allocation for a vector or a matrix.

```
%        Lower Bound
% Read the manufacturing time
time = load('m10j20_8_1363.txt');
% Find m and n
[m,n] = size(time);
% assign time before vector
tb = zeros(m,1);
% assign time after vector
ta = zeros(m,1);
% assign sum vector Tij
Tij = zeros(m,1);
% assign lower bound vector
lowerBound = zeros(m,1);
%
for i = 1:m
    tij = 0;
    for j = 1:n
        tij = tij + time(i,j);
    end
    Tij(i) = tij;
end
% sums (on columns) for time before
sumeCol_b = zeros(m,n);
for j = 1:n
    sumeCol_b(2,j) = time(1,j);
end
%
for i = 3:m
    for j = 1:n
        sumeCol_b(i,j) =
sumeCol_b(i-1,j) + time(i-1,j);
    end
end
% calculation of tb (time before)
for i = 2:m
    t = sumeCol_b(i,1);
    for j = 1:n
        if sumeCol_b(i,j) < t
            t = sumeCol_b(i,j);
            index_j = j;
            j_before = j;
        end
    end
    tb(i,1) = t;
end
% sums (on columns) for time after
sumeCol_a = zeros(m,n);
%
for j = 1:n
    sumeCol_a(m,j) = time(m,j);
end
for i = 1:m-2
    for j = 1:n
        sumeCol_a(m-i,j) =
sumeCol_a(m-i+1,j) + time(m-i,j);
    end
end
%
for j = 1:n
    for i = 1:m-1
        sumeCol_a(i,j) =
sumeCol_a(i+1,j);
    end
end
for j = 1:n
    sumeCol_a(m,j) = 0;
end
%
% calculation of ta (time after)
for i = 1:m-1
    t = sumeCol_a(i,1);
    for j = 1:n
        if sumeCol_a(i,j) < t && j
~= index_j
            t = sumeCol_a(i,j);
            j_after = j;
        end
    end
    ta(i,1) = t;
end
%
for i = 1:m
    lowerBound(i,1) = tb(i,1) +
Tij(i,1) + ta(i,1);
end
LB = max(lowerBound);
%
```

Fig. 5 The Matlab program for *lower bound* calculation (b_i before a_i).

```
%        Lower Bound
% Read the manufacturing time
time = load('m10j20_8_1363.txt');
% Find m and n
[m,n] = size(time);
% assign time before vector
tb = zeros(m,1);
```

```
% assign time after vector
ta = zeros(m,1);
% assign sum vector Tij
Tij = zeros(m,1);
% assign lower bound vector
lowerBound = zeros(m,1);
%
for i = 1:m
    tij = 0;
    for j = 1:n
        tij = tij + time(i,j);
    end
    Tij(i) = tij;
end
% sums (on columns) for time before
sumeCol_a = zeros(m,n);
for j = 1:n
    sumeCol_a(m,j) = time(m,j);
end
%
for i = 1:m-2
    for j = 1:n
        sumeCol_a(m-i,j) =
sumeCol_a(m-i+1,j) + time(m-i,j);
    end
end
% calculation of ta (time after)
for j = 1:n
    for i = 1:m-1
        sumeCol_a(i,j) =
sumeCol_a(i+1,j);
    end
end
for j = 1:n
    sumeCol_a(m,j) = 0;
end
for i = 1:m-1
    t = sumeCol_a(i,1);
    for j = 1:n
        if sumeCol_a(i,j) < t
            t = sumeCol_a(i,j);
            index_j = j;
            j_after = j;
        end
    end
    ta(i,1) = t;
end
% calculation of tb (time before)
sumeCol_b = zeros(m,n);
for j = 1:n
    sumeCol_b(2,j) = time(1,j);
end
%
for i = 3:m
    for j = 1:n
        sumeCol_b(i,j) =
sumeCol_b(i-1,j) + time(i-1,j);
    end
end
for i = 2:m
    t = sumeCol_b(i,1);
    for j = 1:n
        if sumeCol_b(i,j) < t && j
~= index_j
            t = sumeCol_b(i,j);
            j_before = j;
        end
    end
    tb(i,1) = t;
end
%
for i = 1:m
    lowerBound(i,1) = tb(i,1) +
Tij(i,1) + ta(i,1);
end
LB = max(lowerBound);
%
```

Fig. 6 The Matlab program for *lower bound* calculation (a_i before b_i).

2.2 The case of batch manufacturing

If each job is replaced by a batch of identical jobs, and batches may be equal in size or not, then it must be utilized a vector of batches as:

$$S = (s_1, s_2, \dots, s_n) \tag{6}$$

where s_j represents the batch size of job j, with $1 \leq j \leq n$. In such condition, the eq. 4 becomes:

$$T_i^S = \sum_{j=1}^{n} s_j \cdot t_{ij}\,; \tag{7}$$

and the *lower bound* can be written as:

$$LB = \max_{1 \leq i \leq n} \left(b_i + T_i^S + a_i\right); \tag{8}$$

This means that LB is closely dependent on batches size and for its calculation, the eq. 8 should be used.

At Matlab codes shown in Fig. 5 and Fig. 6, some lines of code should be added. For the beginning we must know the size of manufacturing batches according to eq. 6, then the sum of batches manufacturing on each machine must be determined, according to eq. 7. Thus, at lines code in Fig. 5 and Fig. 6, it must be added:

```
%
% minimum t before
minB =
      transpose(min(transpose(b)));
[~,ib] = min(transpose(b));
ib = transpose(ib);
% minimum time after
minA =
      transpose(min(transpose(a)));
[~,ia] = min(transpose(a));
ia = transpose(ia);
lb = minB + sumT + minA;
[lowerBound,ilb] = max(lb);
% LB for batches
tLot = zeros(m,n);
for i = 1:n
    for j = 1:m
        tLot(j,i) =
lot(1,i)*t(j,i);
    end
end
% sum of tij on each machine for
  batches
tLotTranspus =
      sum(transpose(tLot));
sumTLot = transpose(tLotTranspus);
lb_lot = minB + sumTLot + minA;
% the lower bound
[lowerBound_lot,ilb_lot] =
      max(lb_lot);
%
```

Fig.7 The code lines to calculate *lower bound* in the case of batches manufacturing (must be added to code lines in Fig. 5 and Fig. 6).

In the case of searching the optimal sequence of batches manufacturing, even if the batches size is the same for all jobs, the *lower bound* will not be equal with the LB in classic problem multiplied by the batches size. This aspect happens, no matter the value of the batches size, because the *time before* and *time after* does not change in eq. 8 (i.e. they remain the same as in eq.5), except the sums of batches manufacturing time according to eq. 7.

3 CONCLUDING REMARKS

This paper establishes an additional condition to the *lower bound* calculation method, presented in (Taillard, 1993). This condition is due to the fact that at creating an optimal manufacturing schedule, the schedule cannot start and end with the same job. In the floowshop scheduling problems sets introduced by (Taillard, 1993) the paper rectifies the value of the *lower bound* of jobs that does not fulfill the above condition. Also, the paper extends the *lower bound* calculation method from the classical case to the case of batches manufacturing. For both classical and batches cases, the paper shows the source code in Matlab for *lower bound* calculation, so being a valuable additional instrument for the already existent method in the literature.

4 REFERENCES

Abedinnia, H., Glock, C.H., Brill, A. (2016) *New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem*. Computers & Operations Research, Vol. 74, pp. 165–174.

Fernandez-Viagas, V., Dios, M., Framinan, J.M. (2016) *Efficient constructive and composite heuristics for the Permutation Flowshop to minimise total earliness and tardiness*. Computers &OperationsResearch, Vol. 75, pp.38–48.

Fernandez-Viagas, V., Molina-Pariente, J.M., Framinan, J.M. (2019) *Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling*, European Journal of Operational Research, doi: https://doi.org/10.1016/j.ejor.2019.10.017

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 Vol. 55, pp.1243–1255.

Gupta, J.N.D., Stafford Jr. E.F. (2006) *Flowshop scheduling research after five decades*, European Journal of Operational Research, Vol. 169, pp. 699–711.

Hejazi, S.R., Saghafian, S. (2005) *Flowshop-scheduling problems with makespan criterion: a review*. International Journal of Production Research, Vol. 43, No. 14, pp.2895–2929.

Liu, J. (1999) *The Impact Of Neighbourhood Size On The Process Of Simulated Annealing: Computational Experiments On The Flowshop Scheduling Problem*. Computers & Industrial Engineering Vol. 37, pp. 285-288.

Pan, Q.K., Ruiz, R. (2013) *A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime*, Computers & Operations Research, V0l. 40, pp. 117–128.

Taillard, E. (1993) *Benchmarks for basic scheduling problems*. European Journal of Operational Research, Vol. 64, No. 2, pp. 278-285, https://doi.org/10.1016/0377-2217(93)90182-M .

Taillard, E. (2020) *Scheduling instances*, available at: mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html Accessed: 2020-11-24.