Best-Case Lower Bounds in a Group Sequence for the Job Shop Problem

Guillaume Pinot [*] **Nasser Mebarki** [*]

[*] *IRCCyN, 1 rue de la Noé, BP 92101, 44321 Nantes Cedex 3, France (e-mail: surname.name@irccyn.ec-nantes.fr)*

Abstract:
Group sequencing is a well-studied scheduling method for the job shop problem. The goal of this method is to have a sequential flexibility during the execution of the schedule and to guarantee a minimal quality corresponding to the worst case. But the best case quality of a group sequence should also be interesting. This article presents new methods to evaluate the best case quality for any regular objective function. More particularly, three new makespan lower bounds are presented. The experiments performed with these lower bounds exhibit very good performances.

Keywords: Job and activity scheduling; Discrete event systems in manufacturing.

1. INTRODUCTION

The job shop problem with multiple precedence constraints ($J|r_i,\text{prec}|f$ from the classification described in Graham et al. [1979]) is an optimization problem composed of resources, operations, and constraints. Operations (O_i) are executed on resources (M_ℓ, also named as machines) during a processing time p_i with precedence constraints (predecessors (resp. successors) of O_i are given by $\Gamma^-(i)$ (resp. $\Gamma^+(i)$)). A resource can execute only one operation at a time. An operation O_i has a release date r_i, its starting time is denoted by t_i and its completion time is denoted by C_i.

Generally, the job shop problem uses a regular objective function f that is a monotonous function of the C_i. The goal is to minimize this objective function.

The makespan, denoted by $C_{\max}$, calculated $\max C_i$, that corresponds to the total time of execution of the schedule, is a classical regular objective.

But, in reality, manufacturing problems are not so deterministic. This is why group sequencing was created by Erschler and Roubellat [1989]. This method describes a set of feasible schedules in order to delay decisions to take into account uncertainties. Group sequencing evaluates a group sequence according to the worst case quality in the set of feasible schedules.

But the best case quality of a group sequence can also be interesting for different reasons. It gives information on the schedule before the execution: for a given group sequence, with an evaluation of the best case quality and the worst case quality, it would be possible to know the range of all possible qualities of the final schedule. It could also be helpful to evaluate a decision during the execution of the schedule: it would allow to know, during the execution of the schedule, if there is at least one schedule in the group sequence that has no delay.

In this article, we present new methods to evaluate the best case quality of a group sequence.

2. GROUPS OF PERMUTABLE OPERATIONS

Group of permutable operations was first introduced in Erschler and Roubellat [1989]. The goal of this method is to have a sequential flexibility during the execution of the schedule and to guarantee a minimal quality corresponding to the worst case. This method has been widely studied in the last twenty years, in particular in Erschler and Roubellat [1989], Billaut and Roubellat [1996], Wu et al. [1999], Artigues et al. [2005]. For a theoretical description of the method, see Artigues et al. [2005].

A group of permutable operations is a set of operations to be performed on a given resource M_ℓ in an arbitrary order. It is named $g_{\ell,k}$. The group containing the operation O_i is denoted by $g(i)$.

A group sequence is defined as a sequence of groups (of permutable operations) on each machine M_ℓ: $g_{\ell,1},\dots,g_{\ell,v_\ell}$, performed in this particular order. On a given machine, the group after (resp. before) $g(i)$ is denoted by $g^+(i)$ (resp. $g^-(i)$).

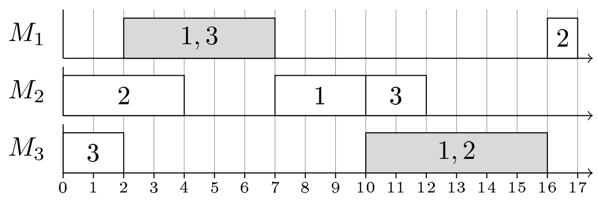
A group sequence is feasible if for each group, all the permutation among all the operations of the same group gives a feasible schedule (*i.e.* a schedule which satisfies all the constraints of the problem). As a matter of fact, a group sequence describes a set of valid schedules, without enumerating them.

The quality of a group sequence is expressed in the same way as of a classical schedule. However, it is measured as

[⋆] This paper was published in the *Proceedings of the 17th World Congress, The International Federation of Automatic Control*, Seoul, Korea, July 6-11, 2008.
doi://10.3182/20080706-5-KR-1001.1295

i	j	$M_{i,j}$	$p_{i,j}$
1	1	1	3
1	2	2	3
1	3	3	3
2	1	2	4
2	2	3	3
2	3	1	1
3	1	3	2
3	2	1	2
3	3	2	2

(a) A job shop problem



(b) A group sequence solving the problem described in Fig. 1a

Fig. 1. A Job Shop Problem Solved by a Group Sequence

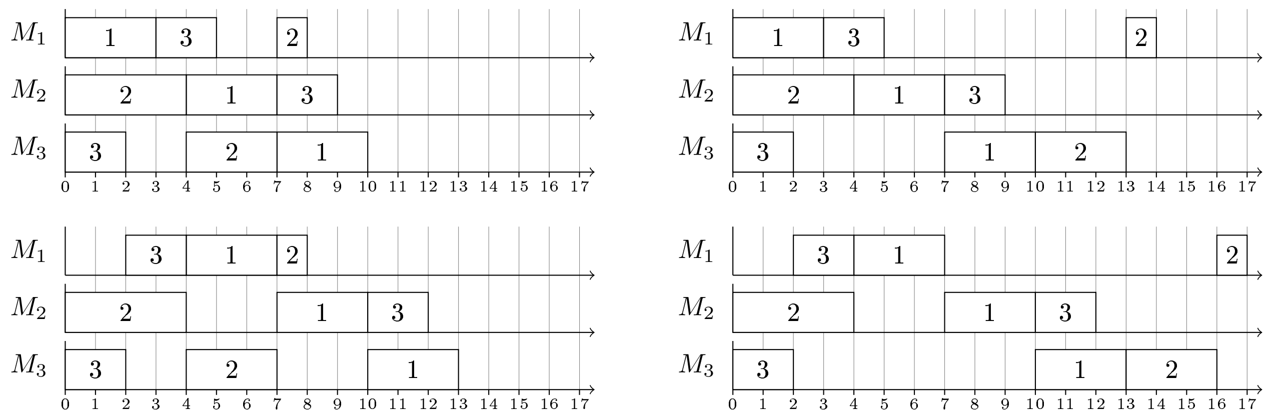


Fig. 2. Semi-active Schedules Described by Fig. 1b

the quality of the worst semi-active schedule[1] found in the group sequence, as defined in Artigues et al. [2005].

To illustrate these definitions, let us study an example. To simplify, the job-shop problem without multiple precedence constraints is used ($J||f$ from the classification described in Graham et al. [1979]). Operations are indexed by a couple ($O_{i,j}$) and the operations need to be executed in second index order ($C_{i,j} \leq t_{i,j+1}$). In the figures, only the first index is written in order to have more readable graphics. Fig. 1a presents a job shop problem with three machines and three jobs, while Fig. 1b presents a feasible group sequence solving this problem. This group sequence is made of seven groups: two groups of two operations and five groups of one operation. This group sequence describes four different semi-active schedules shown in Fig. 2. Note that these schedules do not always have the same makespan: the best case quality is with $C_{\max} = 10$ and the worst case quality is with $C_{\max} = 17$.

Group sequencing has an interesting property: the quality of a group sequence in the worst case can be computed in polynomial time for minmax regular objective functions like makespan and maximum lateness (see Artigues et al. [2005] for the description of the algorithm). Thus, it is possible to compute the worst case quality for large scheduling problems. Consequently, this method may be used to compute the worst case quality in real time during the execution of the schedule. This real-time property makes it possible to dynamically use it in a decision support system.

[1] A semi-active schedule is a feasible schedule in which no local left-shift of an operation leads to another feasible schedule.

This method enables to describe a set of schedules in an implicit manner (*i.e.* without enumerating the schedules) which guarantees a minimal performance. Indeed, as it proposes a group of permutable operations, one can choose inside a group the operation that best fits the real state of the system.

Furthermore, the flexibility added to the schedule should be able to absorb uncertainties. Only three studies have tried to verify this property. Wu et al. [1999] studies the impact of disturbed processing times on the objective of weighted sum of tardiness in comparison with static and dynamic heuristics. When processing times are not greatly disturbed, they observe that group sequencing obtain better performances. Esswein [2003] studies the impact of disturbed processing times, due dates and release dates on a one machine problem and compares its results with a static heuristic method. On average, performances are better with group sequencing than with the static method. Pinot et al. [2007] studies the impact of non-modeled transportation time between two operations. The method exhibits good performances, even when transportation times and processing times are comparable.

3. FINDING THE BEST CASE COMPLETION TIME OF AN OPERATION

3.1 The algorithm

In Artigues et al. [2005], the worst case completion time of each operation is computed in polynomial time using dynamic programming. Our main goal in this section is to compute the best case completion time, that corresponds to the smallest value of C_i in every semi-active schedule described by a group sequence. As this problem is NP-hard,[2] it would be very useful to compute a lower bound in polynomial time for the best case completion times.

We can easily compute such a lower bound for our problem using a relaxation on the resources by making the assumption that each resource has an infinite capacity. In this case, the best case lower bound for starting time of an operation (θ_i) is computed as the maximum of the best case (lower bound for) completion time (χ_j) of all of its predecessors: for an operation O_i, they include the predecessors given by the problem $(\Gamma^-(i))$ as well as the operations on the predecessor group (each operations in $g^-(i)$). For example, in the example described in Fig. 1, the predecessors of operation $O_{2,3}$ (executed on M_1) are operation $O_{2,2}$ (executed on M_3) because of the precedence constraint, and the operations $O_{1,1}$ and $O_{3,2}$ (executed on the same machine M_1) because they are on the predecessor group $(g^-(2,3))$. So, we have:

$$\begin{cases} \theta_i = \max\left(r_i, \max\limits_{j \in g^-(i)} \chi_j, \max\limits_{j \in \Gamma^-(i)} \chi_j\right) \\ \chi_i = \theta_i + p_i \end{cases} \tag{1}$$

Calculating θ_i using (1) is equivalent of the head computation of operation O_i as explained in Carlier and Pinson [1989].

However this bound can be improved using the property of group-sequencing: an operation in a given group cannot be executed before all the operations of its predecessor group have been executed. As a consequence, an operation can only begin after the optimal makespan of the predecessor group.

Thus, it necessitates the computation of the optimal makespan of a group. We have previously computed θ_i as release date, so we can generate a $1|r_i|C_{\max}$ problem instance that corresponds to our problem, with $r_i = \theta_i$. This problem is polynomially solvable by ordering the operations in ascending release date (Brucker and Knust [2007], Lawler [1973]).

Thus, we compute a lower bound of the best case starting time of an operation (θ_i), the best case completion time of an operation (χ_i), and the best case completion time of a group $(\gamma_{g_{\ell,k}})$:

$$\begin{cases} \theta_i = \max\left(r_i, \gamma_{g^-(i)}, \max\limits_{j \in \Gamma^-(i)} \chi_j\right) \\ \chi_i = \theta_i + p_i \\ \gamma_{g_{\ell,k}} = C_{\max} \text{ of } 1|r_i|C_{\max}, \forall O_i \in g_{\ell,k}, r_i = \theta_i \end{cases} \tag{2}$$

3.2 Complexity

The complexity of (2) corresponds to the complexity of the longest path in a directed acyclic graph, which is $O(M)$, M being the number of arcs, using Bellman's algorithm.

For each node χ_i, there is one arc that comes from the node θ_i, so there are n such arcs, n being the number of operations.

[2] Because the reduction between $F||C_{\max}$ and the best $C_{\max}$ in a group sequence is trivial, and that $F||C_{\max}$ is NP-hard, then the best case in a group sequence is NP-hard.

For each node θ_i, considering k as the maximum number of predecessors of an operation, then there are $k+1$ arcs, *i.e.* k that come from the predecessors, and one that come from the predecessor group. So, we have $O((k+1)n)$ arcs that come to every θ_i node, n being the number of operations.

Because a group sequence is a partition of the set of operations, the number of arcs incident to a group node correspond to the number of operations n.

So, the number of arcs in the graph is

$$O(n) + O((k+1)n) + O(n) = O(kn)$$

On the group's nodes, we use an algorithm with the complexity $O(h \log h)$, h being the number of operations on the group, so the complexity of all the computation on group's nodes is:

$$\begin{aligned} O(h_1 \log h_1 \times \cdots \times h_n \log h_n) &= O(n \log(\max h_i)) \\ &= O(n \log n) \end{aligned}$$

Thus, the global complexity of the algorithm is:

$$O = O(kn) + O(n \log n) = O(kn + n \log n).$$

In most cases, $k = 1$ for classical job shop, and the complexity is $O(n \log n)$. On the worst case, $k = n$, and the complexity is $O(n^2)$.

3.3 Example of the algorithm's execution

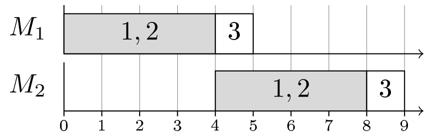


Fig. 3. The group sequence described in Tab. 1

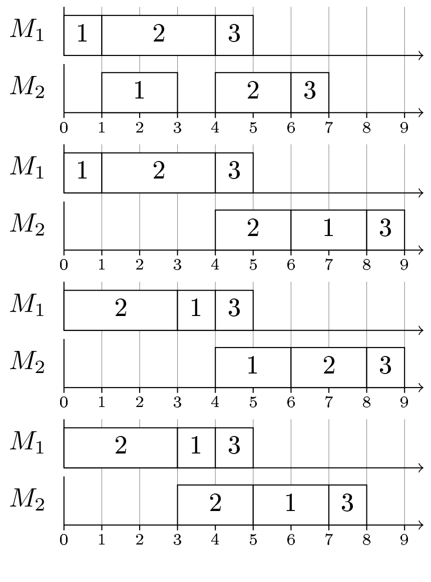


Fig. 4. Semi-active Schedules Described by Fig. 3

Let us illustrate these algorithms with a small example. To keep this example easily understandable, a small flow shop problem is considered. The problem, the group sequence and the different values computed by the algorithms are presented in Tab. 1. A worst-case representation of the

Problem description					Using (1)		Using (2)			Optimal val.	
i	j	$M_{i,j}$	$p_{i,j}$	$g_{\ell,k}$	$\theta_{i,j}$	$\chi_{i,j}$	$\theta_{i,j}$	$\chi_{i,j}$	$\gamma_{g(i,j)}$	$\theta_{i,j}$	$\chi_{i,j}$
1	1	1	1	$g_{1,1}$	0	1	0	1	4	0	1
1	2	2	2	$g_{2,1}$	1	3	1	3	5	1	3
2	1	1	3	$g_{1,1}$	0	3	0	3	4	0	3
2	2	2	2	$g_{2,1}$	3	5	3	5	5	3	5
3	1	1	1	$g_{1,2}$	3	4	**4**	**5**	5	4	5
3	2	2	1	$g_{2,2}$	5	6	5	6	6	**6**	**7**

Table 1. Example of a flow shop

group sequence can be viewed in Fig. 3. The four semi-active schedules described by the group sequence are represented in Fig. 4. These are used to compute the optimal value presented in Tab. 1.

We can see the improvement between (1) and (2) for the operation $O_{3,1}$. The computation of the makespan of the group $g_{1,1}$ improves the lower bound of $\chi_{3,1}$ up to its optimal value.

Eq. (2) does not give the optimal value of $\chi_{3,2}$: the optimal value is 7, but (2) finds 6. This is due to the fact that any two best case starting times for any two operations may not appear in the same semi-active schedule. In the example, there is no semi-active schedule for which $t_{1,2} = \theta_{1,2} = 1$ (first schedule on Fig. 4) and $t_{2,2} = \theta_{2,2} = 3$ (fourth schedule on Fig. 4): if $t_{1,2} = 1$, then $t_{2,2}$ cannot be 3 but only 4 because of the precedence constraints. So, $\gamma_{g_{2,1}}$ is not equal to the best case completion time of the group $g_{g_{2,1}}$ (6, as on the first schedule on Fig. 4), but only a lower bound (5), and then the error is propagated to the successor groups.

4. LOWER BOUNDS FOR REGULAR OBJECTIVES

Because regular functions increase monotonously with increasing C_i, a lower bound of C_i enable to compute a lower bound for any regular objective function. For this, we can use χ_i as a lower bound of C_i.

Let us study minsum regular objective functions, denoted by $\bar{f}$. Minsum regular objective functions are the functions that can be modeled as a sum of functions on the completion times C_i that need to be minimized. If we call f_i the function for the operation O_i, the objective function can be computed as:

$$\bar{f} = \sum_{\forall O_i} f_i(C_i)$$

For example, a well known minsum objective is the mean tardiness denoted by $\bar{T}$ with $f_i(C_i) = \max(0, C_i - d_i)$.

We can directly use the fact that χ_i is a lower bound of C_i to create a generic minsum lower bound as:

$$\mathrm{LB}(\bar{f}) = \sum_{\forall O_i} f_i(\chi_i)$$

But, due to the fact that our schedule are made of group sequences, this formulation can easily be improved: there is at least one operation O_i per group which has $\gamma_{g(i)}$ as a lower bound. The idea is to use the minimal $f_i(\gamma_{g(i)})$ on each group as the lower bound of $f_i(C_i)$:

$$\mathrm{LB}(\bar{f}) = \sum_{\forall g_{\ell,k}} \left(f_j(\gamma_{g_{\ell,k}}) + \sum_{O_i \in g_{\ell,k}, i \neq j} f_i(\chi_i), \right.$$
$$\left. j \text{ such as } f_j(\gamma_{g_{\ell,k}}) = \min_{O_i \in g_{\ell,k}} f_i(\gamma_{g_{\ell,k}}) \right)$$

There is yet another kind of generic regular objective that is often used: minmax regular objectives, denoted by $f_{\max}$. These are functions that can be modeled as the maximum of functions on the completion times C_i that need to be minimized. If we call f_i the function for a given operation O_i, the objective function can be computed as:

$$f_{\max} = \max_{\forall O_i} f_i(C_i)$$

Using the same technique as for minsum objectives, we can calculate a lower bound as:

$$\mathrm{LB}(f_{\max}) = \max_{\forall g_{\ell,k}} \left(\max\left(f_j(\gamma_{g_{\ell,k}}), \max_{O_i \in g_{\ell,k}} f_i(\chi_i) \right), \right.$$
$$\left. j \text{ such as } f_j(\gamma_{g_{\ell,k}}) = \min_{O_i \in g_{\ell,k}} f_i(\gamma_{g_{\ell,k}}) \right)$$

5. MAKESPAN LOWER BOUND

5.1 *Using regular objective formulation*

The makespan, denoted by $C_{\max}$, is a regular minmax objective. It represents the total time span of the schedule.

A lower bound can be computed using the formula for regular minmax objective:

$$\mathrm{LB}(C_{\max}) = \max_{\forall g_{\ell,k}} \left(\max\left(\gamma_{g_{\ell,k}}, \max_{O_i \in g_{\ell,k}} \chi_i \right) \right)$$
$$= \max_{\forall g_{\ell,k}} \gamma_{g_{\ell,k}}$$

This lower bound is referred as the "natural lower bound," denoted by "Natural LB."

To improve this lower bound, let us study the classical lower bound for the job shop problem.

5.2 *The classical lower bound for the job shop problem*

The classical lower bound for the job shop problem, described in Carlier and Pinson [1989], consists of the relaxation of the problem in different one machine problems: for each machine, we take the operations that are executed by this machine. For each operation, we have three values:

- The head r_i, that is a lower bound of the earliest starting time of the operation in the job shop problem (with possible partial schedule).

- The tail q_i, that is a lower bound of the time between the end of the operation and the end of the schedule (with possible partial schedule).
- The processing time p_i, the same as the processing time of the operation in the job shop problem.

Thus, we have a one machine problem as defined by Carlier [1982] for each machine: each operation O_i cannot begin before r_i, executes for a duration p_i on the machine, and the operation will end q_i after the end of its execution. The objective is to minimize the makespan of the schedule. This problem is equivalent to $1|r_i|L_{\max}$ with $d_i = -q_i$. It is NP-hard (Brucker and Knust [2007], Lenstra et al. [1977]). Carlier [1982] provides a good optimal algorithm to solve this problem. The classical lower bound of this problem is called Jackson's preemptive schedule and its complexity is $O(n \log n)$, n being the number of operations.

The optimal value of such a one machine problem, is a lower bound of the job shop problem. So, in practice, we take the maximal optimal makespan (resp. a lower bound) of the generated one machine problem as a lower bound of the job shop problem. The efficiency of this technique relies on the quality of the heads and tails as shown by Carlier and Pinson [1990, 1994].

5.3 An improved makespan lower bound

The first step of the classical lower bound for the job shop problem is a one-machine-problem relaxation. For group sequencing, a group relaxation is adapted: each group is converted into a one machine problem.

Then, efficient heads and tails have to be found for these one machine problems. θ_i using (2) is a valid head and must be quite effective. Because of the symmetry of heads and tails, tails can be computed as θ_i using a reversed (2): rather than starting the computation at the beginning of the scheduling problem, the computation begins at the end. So, replacing predecessor by successor, the new formulation is

$$\begin{cases} \theta'_i = \max\left(\gamma'_{g^+(i)}, \max_{j \in \Gamma^+(i)} \chi'_j\right) \\ \chi'_i = \theta'_i + p_i \\ \gamma'_{g_{\ell,k}} = C_{\max} \text{ of } 1|r_i|C_{\max}, \forall O_i \in g_{\ell,k}, r_i = \theta'_i \end{cases} \quad (3)$$

with θ'_i being a valid tail for the group relaxation. The one machine problems are then generated.

Now, as for the classical lower bound for the job shop problem, the one machine problems have to be evaluated using Jackson's preemptive schedule or Carlier's algorithm. The maximal evaluation will be the lower bound of the group sequence.

The lower bound based on the optimal resolution of the one-machine-problem relaxation is denoted by "Optimal OMP LB" and the lower bound based on the Jackson preemptive schedule of the one-machine-problem relaxation is denoted by "JPS OMP LB."

Next section will present an evaluation of the makespan lower bounds described above.

6. EXPERIMENTS

6.1 Protocol

The goal of these experiments is to compare three makespan lower bounds: Natural LB presented in section 5.1, Optimal OMP LB and JPS OMP LB presented in section 5.3.

We took a well known set of benchmark instance called `la01` to `la40` from Lawrence [1984]. These instances are widely used in the job shop literature. These are classical job shop instances, with m operations on each job (m as the number of machine), each operation of a job is executed on a different machine. It is composed of 40 instances of different sizes (5 instances for each size).

For each instance, we generated group sequences with known optimal value and very high flexibility. To generate these group sequences, we used a greedy algorithm that merges two successive groups according to different criteria until no group merging is possible. This algorithm begins with a one-operation-per-group sequence computed by the optimal algorithm described in Brucker et al. [1994] (so, by construction, the optimal makespan of these group schedules is the makespan of the one-operation-per-group sequence). The greedy algorithm is described in Esswein [2003]. The source code of the program used to makes these experiments can be downloaded at `http://www.irccyn.ec-nantes.fr/~pinot/`.

Then, for each generated group sequences, we compute the gap between each lower bound and the optimal makespan of the group sequence. The results are represented on a boxplot in Fig. 5 and the results corresponding to the instance sizes are presented in Tab. 2.

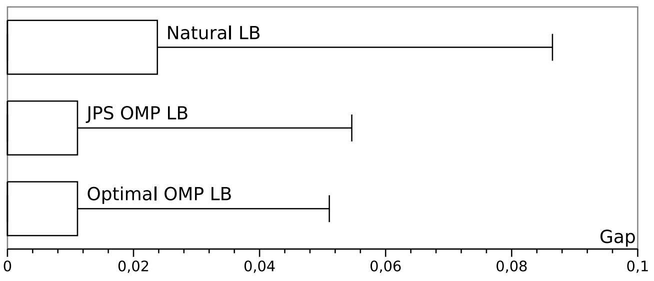


Fig. 5. Gap of the different lower bound

Optimal OMP LB and JPS OMP LB have similar execution times. Execution times of Natural LB is about two times shorter than JPS OMP LB.

6.2 Discussion

Execution times give important information on the critical part of the lower bounds. Difference between Natural LB and JPS OMP LB can be explained by the fact that Natural LB uses only head computation whereas JPS OMP uses head and tail computation. The fact that Optimal OMP LB and JPS OMP LB have almost the same execution time reveals that the optimal one-machine-problem resolution is not significant relative to head and tail computation.

Size	Natural LB		JPS OMP LB		Optimal OMP LB	
	Gap	Time (ms)	Gap	Time (ms)	Gap	Time (ms)
10×5	0.00369	0.1840	0.00369	0.3712	0.00369	0.3840
15×5	0.00472	0.3120	0.00000	0.6128	0.00000	0.6400
20×5	0.00000	0.4640	0.00000	0.8648	0.00000	0.9040
10×10	0.05535	0.5280	0.03652	0.9440	0.03270	0.9920
15×10	0.01208	0.8480	0.00324	1.5416	0.00305	1.6401
20×10	0.01132	1.2000	0.00708	2.1857	0.00708	2.3281
30×10	0.00000	1.9241	0.00000	3.5450	0.00000	3.7682
15×15	0.02877	1.4480	0.01864	2.6225	0.01713	2.8401
Mean	0.01477	0.8635	0.00865	1.5859	0.00796	1.6871

The size is noted as $n \times m$ with n the number of jobs and m the number of machines.

Table 2. Mean gap of the different lower bounds according to the size of the instance

In Fig. 5, we can see that the median is 0 for each lower bound. Thus, for more than half of our instances, the lower bound is optimal (21/40 for Natural LB, 23/40 for the others). This is a very encouraging result, which means that the computation of χ_i is quite effective on these instances.

Natural LB is the weakest lower bound, but is quite effective with a mean gap of 1.5% (Tab. 2) and the worst gap at 8.6% (Fig. 5).

JPS OMP LB and Optimal OMP LB have performances of the same scale, with a little advantage for Optimal OMP LB. Because Optimal OMP LB have greater or equal performance than JPS OMP LB (by construction) and that the execution times are quite the same, Optimal OMP LB seems to be the best lower bound.

In these experiments, the difference of performances between these two lower bounds does not seem to be so high, but, used in an exact branch and bound method, small improvement can have big repercutions. In an exact method using these lower bounds, the method using Optimal OMP LB is about 10 times faster than using JPS OMP LB.

It seems that wider instances exhibit small gaps. It is consistent with the fact that wide job shop instances seem simpler instances in Brucker et al. [1994].

7. CONCLUSION

In this article, a first step on best case evaluation of group sequencing is done. Eq. (2) seems to be a good method to generate different lower bounds. For the makespan, Optimal OMP LB gives very good results.

Other tools can also be helpful for best case evaluation of group sequencing. Heuristics and exact methods would be very useful to complete the resolution of this problem.

REFERENCES

Christian Artigues, Jean-Charles Billaut, and Carl Esswein. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2):314–328, September 2005.

Jean-Charles Billaut and François Roubellat. A new method for workshop real-time scheduling. *International Journal of Production Research*, 34(6):1555–1579, 1996.

Peter Brucker and Sigrid Knust. Complexity results for scheduling problems. `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`, 2007. [online; retrieved on 2008-06-18].

Peter Brucker, Bernd Jurisch, and Bernd Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49(1-3):107–127, 1994.

Jacques Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42–47, September 1982.

Jacques Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.

Jacques Carlier and E. Pinson. A practical use of jackson's preemptive schedule for solving the job shop problem. *Annals of Operations Research*, 26(1-4):269–287, 1990.

Jacques Carlier and E. Pinson. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2):142–147, October 1994.

Jacques Erschler and François Roubellat. An approach for real time scheduling for activities with time and resource constraints. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*. Elsevier, 1989.

Carl Esswein. *Un apport de flexibilité séquentielle pour l'ordonnancement robuste*. Thèse de doctorat, Université François Rabelais Tours, 2003.

Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and A. G. H. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

Eugene L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5):544–546, January 1973.

S. Lawrence. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

Jan Karel Lenstra, A. H. G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

Guillaume Pinot, Olivier Cardin, and Nasser Mebarki. A study on the group sequencing method in regards with transportation in an industrial FMS. In *Proceedings of the IEEE SMC 2007 International Conference*, 2007.

S. David Wu, Eui-Seok Byeon, and Robert H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124, 1999.