

HEURISTIC METHODS FOR PROBLEMS WITH BLOCKING CONSTRAINTS SOLVING JOBSHOP SCHEDULING

Wajdi TRABELSI, Christophe SAUVEY, Nathalie SAUER

LGIPM /University Paul Verlaine Metz
COSTEAM / INRIA Nancy Grand Est
Île du Saulcy
57045 Metz Cedex 01 - France
w.trabelsi@univ-metz.fr, sauvey@univ-metz.fr, sauer@univ-metz.fr

ABSTRACT: *Jobshop problem is one of the most difficult classical scheduling problems. Very simple special cases of jobshop problem are strongly NP-hard. This paper deals with makespan minimization in jobshop scheduling problems with a specific blocking constraint met in several industrial problems. We also propose and define another new type of blocking constraint.*

After a problem description and the definitions of different blocking constraints, possible conflicts situations are considered. A heuristic method is then proposed which avoids conflicting situations and numerical results with specific blocking constraints are presented and discussed.

KEYWORDS: *Scheduling, Jobshop, blocking constraints, heuristics, makespan.*

1 INTRODUCTION

The jobshop problem is one of most difficult classical scheduling problems met in professional world, especially in production systems. Several models of scheduling differ according to used technologies and applied constraints in these systems. In this work, we are interested in jobshop problem with particular blocking constraints and our goal is to minimize the completion time of last operation, also called makespan.

In the considered blocking constraint, a machine stays blocked until one job has finished its operation on the next machine and has left for this machine. This blocking, labelled "Release when Completing Blocking (RCb)", has been introduced for the first time by (Dauzère-Pérès *et al.*, 2000). In this paper we also define a variant of RCb blocking constraint, called RCb*, for which a machine will be immediately available to treat the next operation after his one on the following machine is finished without regard to whether it leaves or not the machine. RCb and RCb* constraints are practical in some industrial environments, such as waste treatment industry and aeronautics part fabrication (Martinez, 2005).

The classical jobshop scheduling problem was studied by many authors. As this problem is NP-hard (Garey and Johnson, 1979), use of exact methods is limited to small-size problems. To solve this problem, many exact and approached methods were developed (Jain and Meeran, 1999), (Blazewicz *et al.*, 1996). Among works using

metaheuristics, we can cite those using simulated annealing (Van Laarhoven *et al.*, 1992), taboo search (Nowicki, 1996), and genetic algorithms (Volta, 1995).

In the instance of classical blocking problems, we can cite the works of (Brucker *et al.*, 2006), as well as (Mati 2002) that propose an exact method and a metaheuristic to solve multi-resources jobshop problem. For RCb constraint, a PLNE modeling and a metaheuristic are presented in (Martinez, 2005) for the case of the flowshop and hybrid flowshop. However, to our knowledge, only the work of (Gorine and Sauer, 2008) concerns optimization of jobshop with this particular blocking constraint.

This paper is organized as follows. A precise description of the problem is given in section 2. Details of blocking constraints are given in section 3. In section 4, we present problems encountered in algorithm construction and choice criterion used in our heuristic. Computational results and performance evaluation are presented in section 5. Last section concludes and gives some perspectives to our work.

2 PROBLEM DESCRIPTION

In jobshop scheduling problem, a set of n jobs, $J = \{J_1, J_2, \dots, J_n\}$, must be executed on a set of m machine, $M = \{M_1, M_2, \dots, M_m\}$. Every job J_i require an operation order, $O_i = \{O_{i1}, O_{i2}, \dots, O_{ini}\}$, that must be executed according to his manufacture process. Operation O_{ij} needs an execution time P_{ij} on machine $M_{ij} \in M$. Every machine can only execute one job at any time. In this paper, no

intermediate buffer space is available and preemptive operation is not authorized. Objective function consists in determining best scheduling in order to reduce makespan i.e. the time where all operations are completed.

Many studies and research were realized about classical jobshop problem and the blocking constraint remains one of the most important challenges to take up. Indeed, some situations presented as a blocking time, generated by a non-existence of buffer space, are practical in some industrial environments.

In the following, we present different cases of jobshop problem: classical jobshop scheduling problem and all of four blocking constraints dealt within this paper. To illustrate these different cases, we consider an example with five jobs and three machines. Routing for each of five jobs is as follows:

- $J_1: M_1 \rightarrow M_3$
- $J_2: M_2 \rightarrow M_1 \rightarrow M_3$
- $J_3: M_2 \rightarrow M_3$
- $J_4: M_3$
- $J_5: M_2 \rightarrow M_1 \rightarrow M_3$

Sequences on the machines which are represented, could be the following:

- $M_1: J_1 J_2 J_5$
- $M_2: J_2 J_3 J_5$
- $M_3: J_1 J_2 J_3 J_4 J_5$

2.1 Classical jobshop

In a jobshop production line, the capacity of intermediate buffer acts on the date of operations passage on different machines; indeed, if we consider that the buffer space capacity is unlimited (case of classical job-shop) a machine will be immediately available to execute next operation after its operation in process is finished.

In the example (Figure 1), machine M_2 is available to treat job J_5 operation as soon as job J_3 operation on M_2 will be finished.

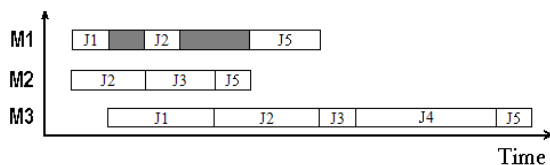


Figure 1: Classical jobshop problem

2.2 Classical blocking constraint RSb

Whereas for a limited or null capacity, a job remains blocked on a machine as long as the following machine in process is not available, or as long as there is no place in stock.

In classical blocking problem, a machine will be immediately available to execute its next operation after the following machine in process is available or that there be place in the stock. This blocking, labelled "Release when Starting Blocking (RSb)" was met in several industrial applications, as for example automated chains of steel production factories.

The example with classical blocking RSb is given Figure 2: the job J_3 remains blocked on machine M_2 as long as the following machine M_3 is not available.

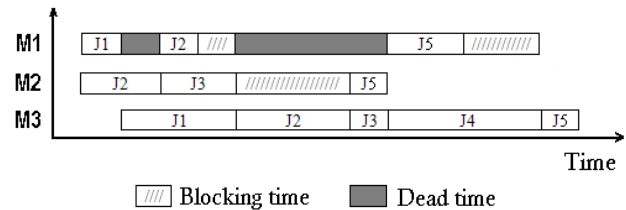


Figure 2: Jobshop problem with classical blocking RSb

2.3 Specific blocking constraint RCb

In this case of blocking constraint, a machine will be immediately available to treat its next operation after operation of its job on the following machine in process is finished and that it has left this machine. This constraint was introduced for the first time in [Dauzère-Pères 2000].

We can find this case of constraint in some industrial environments, such as waste treatment industry and aeronautics part fabrication. Considering waste treatment industries as an example, a company receives different types of industrial and farm waste to be treated. The waste are brought by trucks and unloaded into silos. The cargo is then treated by one and only one machine. As the products are processed from silo to mixer, the silo cannot be available until the process on the mixer of the totality of waste is finished [Martinez 2005].

In our example (Figure 3), machine M_2 remains blocked by job J_5 until the following operation is finished and job J_5 leaves machine M_1 .

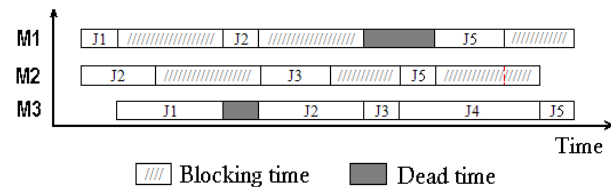


Figure 3: Jobshop problem with specific blocking RCb

2.4 Blocking constraint RCb^*

In this work we defined a variant of the RCb blocking constraint, called RCb^* , for which a machine will be immediately available to treat its next operation after its

job on following machine in process is finished without regard to it leaves or not the machine.

RCb^* constraint (Figure 4) differs of classical blocking RSb by the fact that machine M_2 remains blocked by job J_3 until his operation on machine M_3 is finished.

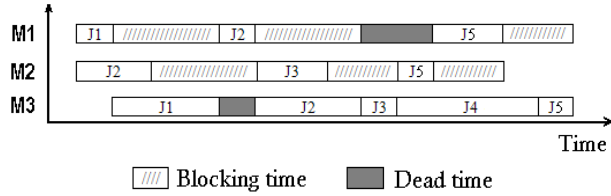


Figure 4: Jobshop problem with specific blocking RCb^*

The difference between a RCb blocking and a RCb^* blocking is underlined by job J_5 (Figure 3 and Figure 4): in a RCb^* blocking problem, machine M_2 remains blocked by job J_5 until the end of execution of the following operation on M_1 , whereas in RCb problem, blocking time is bigger since machine M_2 will be available only when the following operation is finished and job J_5 leaves machine M_1 which corresponds to the beginning of following operation on M_3 .

2.5 No-wait

A model no-wait is also a problem blocking but it presents a special case: It occurs when two consecutive operations must be executed without any interruption. This case is typically like the workshops of surface treatment.

For a no-wait constraint model (Figure 5), both operations of job J_3 must be executed without any interruption, hence the lag of the job J_3 beginning date on the machine M_2 towards the right.

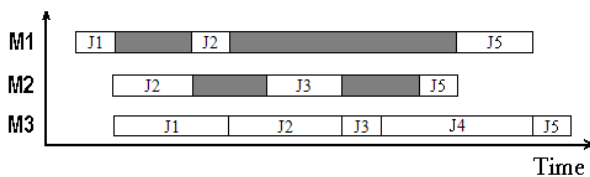


Figure 5: Jobshop problem with no-wait constraint

3 HEURISTICS

Usually, a heuristic is a problems solving method not based on formal models and that provides quickly (in polynomial time) a feasible solution, not necessarily optimal, for a NP-hard optimization problem. Heuristics appeared for overcoming two problems of exact methods:

- Difficulty to formulate a special problem in a framework of modeling often restrictive and too simplifying.
- Treatment computing times necessitated in exact methods can be too long.

We note from the works of [Gorine, 2008] that to solve jobshop problems with RCb blocking constraint, it is not conceivable to use exact methods, because of a too important calculation time. Therefore, we will resort to approached methods by developing heuristics.

In this paper, we use lower bounds proposed in [Gorine 2008] to evaluate obtained solutions in RCb^* constrained cases. Indeed, proposed lower bound in RCb case is also valid in RCb^* case. To evaluate obtained results of RCb constrained problem, we compare them to optimal solutions.

3.1 Used method

We dispose of two types of data: one presenting the passage order of jobs on machines, and the other giving execution time of every operation. We developed an algorithm which gives a feasible operations schedule, in the same time looking for smallest possible makespan.

This algorithm is used a construction method based on the choice between different ways justified by the choice of operation to place. Sometimes, we met situation in which we are unable to place any operation, so that algorithm blocked. This is why we studied these situations, in order to avoid them and thus have directly a feasible schedule.

3.2 Conflicts

In the construction of heuristics, we met cases that do not give reasonable solutions; this is due to situations called "conflicts" from which scheduling becomes no feasible, *i.e.* we meet unsolvable situations.

In both following paragraphs, we give more details about these conflicts in order to avoid them during algorithm construction in RCb and RCb^* blocking cases.

3.2.1 Constraint RCb^*

The RCb^* constraint consists in freeing a machine M of his job J when following operation on next machine in process of this job is finished. Therefore, if an operation O_{ij} processed on machine M_{ij} , then M_{ij} will be available only when the execution of operation $O_{i(j+1)}$ on machine $M_{i(j+1)}$ will be finished (Figure 4).

Then, every operation depends on its following operation and this can generate conflicts that lead to unsolvable situations. We present afterwards different examples of encountered conflicts.

Case 1: Conflict between two jobs with two operations at least

For conflict case presented in Figure 6-a, we notice a conflict that leads to a blocking system and scheduling is no longer feasible, because job J_1 that is processed on the machine M_1 cannot pass to machine M_2 occupied by the job J_2 , as well as for job J_2 that is processed on machine M_2 cannot pass to machine M_1 occupied by job J_1 . Therefore, to obtain a feasible schedule (Figure 6-b), if one operation out of two jobs is placed, it is necessary to finish first job second operation before beginning second job first operation.

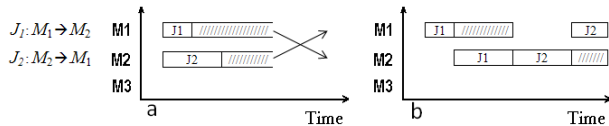


Figure 6: Feasible schedule for conflict between two jobs with two operations at least

Case 2: Conflict between two jobs with three operations at least

Let be a machine M_1 occupied by job J_1 , placing a job J_2 on machine M_3 , such as routing is as indicated in Figure 7, leads us to a conflict similar to first case. Indeed, if we place job J_2 first operation (job J_1 first operation already placed), we will have choice to place either the job J_1 second operation that leads us to conflict case 1 ($M_2 \rightarrow M_3$ and $M_3 \rightarrow M_2$) (Figure 7), or job J_2 second operation which leads us to another conflict case 1 ($M_2 \rightarrow M_1$ and $M_1 \rightarrow M_2$). Therefore, to obtain a feasible schedule, if one first operation of one out of two jobs is placed, then first operation of second job cannot be placed before the end of first sequence job.

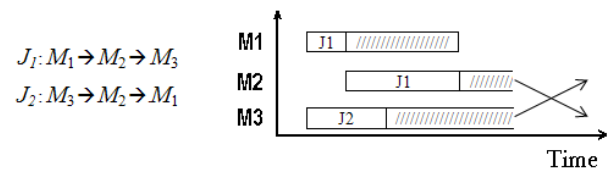


Figure 7: Conflict between two jobs with three operations at least

Case 3: Conflict between three jobs

In order to avoid a loop of occupied machines (Figure 8), we have to verify availability of next operation machine in the process. Therefore, if following machine is not available, we have to verify if it can be liberated or not.

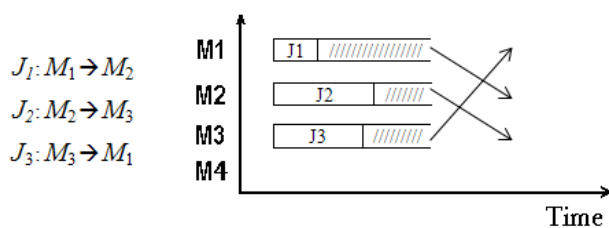


Figure 8: Conflict between three jobs

Case 4: Other conflicts

Several conflicts can be diverted from two first cases: if for a job, we have a part of routing that represents an operation sequence that go through different machines such as this sequence is the same for another job but of opposite direction, then if one of the first operations of one of both jobs is placed, then second cannot be placed before the end of the sequence of the first one.

$$J_1: M_1 \rightarrow M_A \rightarrow M_B \rightarrow \dots \rightarrow M_C \rightarrow M_D \rightarrow M_2$$

$$J_2: M_2 \rightarrow M_D \rightarrow M_C \rightarrow \dots \rightarrow M_B \rightarrow M_A \rightarrow M_1$$

In this case, if first operation on machine M_1 of job J_1 is placed, then first operation on machine M_2 of job J_2 can starts only when the sequence of the first job is finished.

For example, if we consider the conflict of case 1:

$$J_1: M_1 \rightarrow M_2$$

$$J_2: M_2 \rightarrow M_1$$

We demonstrated that if we start with first operation of the job J_1 we are obliged to take following way: (O_{11} - O_{12}) - (O_{21} - O_{22}) (Figures 6-b).

Therefore if we develop this conflict and we put to its extremities another conflict, it becomes:

$$J_1: M_3 \rightarrow M_1 \rightarrow M_2 \rightarrow M_4$$

$$J_2: M_4 \rightarrow M_2 \rightarrow M_1 \rightarrow M_3$$

And this is considered as a conflict if we start job J_2 on machine M_4 when job J_1 on machine M_3 is already placed. Because next operation to place will be O_{12} (resp. O_{22}) which gives the following remaining operations and occupied machines:

$$J_1: M_1 \rightarrow M_2 \rightarrow M_4 \quad (\text{resp.}) \quad J_1: M_3 \rightarrow M_1 \rightarrow M_2 \rightarrow M_4$$

$$J_2: M_4 \rightarrow M_2 \rightarrow M_1 \rightarrow M_3 \quad J_2: M_2 \rightarrow M_1 \rightarrow M_3$$

So we fall on case 2 conflict.

This continues as long as we develop again conflict by other conflicts at extremities.

3.2.2 Constraint RCb

RCb constraint consists in freeing a machine M of his job J when following operation of same job is finished and leaves following machine. Therefore, if an operation O_{ij} passes on machine M_{ij} , then M_{ij} will be available only when operation $O_{i(j+2)}$ starts on machine $M_{i(j+2)}$ (Figure 3).

So, each operation depends on its two following operations and this can generate more conflicts than those met in *RCb** blocking case. Indeed, conflicts defined in *RCb** blocking case are verified in *RCb* blocking case. We present afterward different examples of met conflicts.

We take same used cases in preceding section of *RCb** constraint up again, but with replacing immediately following operation by two operations:

Case 1:

$$J_1: M_1 \rightarrow M_X \rightarrow M_Y \quad (\text{with } X \text{ or } Y = 2)$$

$$J_2: M_2 \rightarrow M_{X\emptyset} \rightarrow M_{Y\emptyset} \quad (\text{with } X\emptyset \text{ or } Y\emptyset = 1)$$

Case 2:

$$J_1: M_1 \rightarrow M_R \rightarrow M_T \rightarrow M_X \rightarrow M_Y \quad (\text{with } X \text{ or } Y = 3)$$

$$J_2: M_3 \rightarrow M_S \rightarrow M_T \rightarrow M_{X\emptyset} \rightarrow M_{Y\emptyset} \quad (\text{with } X\emptyset \text{ or } Y\emptyset = 1)$$

Case 3:

$$J_1: M_1 \rightarrow M_X \rightarrow M_Y \quad (\text{with } X \text{ or } Y = 2)$$

$$J_2: M_2 \rightarrow M_{X\emptyset} \rightarrow M_{Y\emptyset} \quad (\text{with } X\emptyset \text{ or } Y\emptyset = 3)$$

$$J_3: M_3 \rightarrow M_{X\emptyset\emptyset} \rightarrow M_{Y\emptyset\emptyset} \quad (\text{with } X\emptyset\emptyset \text{ or } Y\emptyset\emptyset = 1)$$

These three cases can lead to unsolvable situations but there are some others derivatives to these cases and it becomes more complicated to describe them. Therefore, we currently limited our study on small-sizes problems: 3 jobs ó 3 machines and 3 jobs ó 4 machines.

3.3 Proposed heuristic

Proposed heuristics are based on construction method. In jobshop problem, every job has an own sequence to him (*i.e.* any order of process on different machines, from which the idea to schedule jobs "by operation").

From a partial solution, it is necessary to choose which operation of which job we must place next. For that, we chose a criterion that combines three parameters:

- *Cpmax*: Partial Makespan. It is about calculating completion time of new placed operation as taking account of generated blocking.
- *SomTpsIn*: Sum of inactive times. It is about calculating time during which machines are inactive while including dead times and blocking times.
- *SomTpsEx*: Sum of placed operations in partial schedule execution times.

To have a most optimal possible scheduling, we have tendency on one hand to minimize inactive times of machines and completion time, and on the other hand to maximize machines use, from which comes criterion:

$$Cr = \min (Cpmax + SomTpsIn \text{ ó } SomTpsEx).$$

These calculations are done only when conflicts conditions treated in paragraph 3.2 are verified so we can't place all operations in partial schedule.

Then, *n* schedules are constructed by switching job to place at first and finally we choose one that gives the smallest final makespan *Cmax*.

Example: Let be following jobs to order:

$$J_1: (M_1, 3) \rightarrow (M_2, 3) \rightarrow (M_3, 3)$$

$$J_2: (M_2, 4) \rightarrow (M_3, 3) \rightarrow (M_1, 1)$$

$$J_3: (M_3, 4) \rightarrow (M_1, 2) \rightarrow (M_2, 2)$$

With (M_i, P) : operation executed on machine M_i with P execution time.

Let have as initial placed operation the first operation O_{11} , we have three possibilities (since there are three jobs to order) to place the following operation: O_{12} , O_{21} or O_{31} (Figure 9). We use the choice criterion between these three cases to choose the operation that gives the smaller criterion.

We will explain how we calculated these different parameters through first case *i.e.* when the operation O_{12} is placed (Figure 9-a):

- *SomTpsEx* = 6 ut: we placed two first operations of first job and each placed operation presents three time units of execution on machine.
- *SomTpsIn* = 9 ut: second operation starts at time 3, therefore interval [0-3] where machine M_2 is available but that does not treat any job, is considered as a dead time, as for blocking time is of six time units because machine M_1 remained blocked during interval [3-6] and M_2 will remain blocked during at least three time units that present execution time of first job third operation not yet executed.
- *Cpmax* = 9 ut: partial makespan represents the minimal date from which all machines will be available.

$$\text{From which } Cr(a) = 9 + 9 \text{ ó } 6 = 12.$$

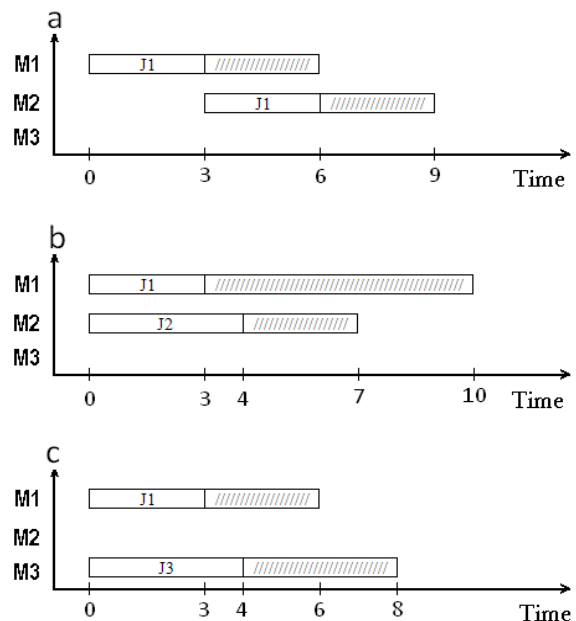


Figure 9: Possible scheduling cases

The results of each possibility (O_{12} , O_{21} or O_{31}) are given Table 1. In this example, the operation O_{31} is the second operation which is placed in the partial sequence.

	Execution time « $SomTpsEx$ » (ut)	Dead time + Blocking time « $SomTpsIn$ » (ut)	Partial makespan « $Cpmax$ » (ut)	Cr
A	6	3 + 6	9	12
B	7	0 + 10	10	13
C	7	0 + 7	8	8

Table 1: Choice criterion used to place next operation

Now, we have two placed operations O_{11} and O_{31} (Figure 9-c), so the following operation to place is one of : O_{12} , O_{21} or O_{32} . Here, O_{12} is the only possible operation to be placed (Figure 10); indeed, if we place O_{21} on M_2 , all machines will be occupied which is not possible and we can not place O_{32} because M_1 is occupied by the job J_1 .

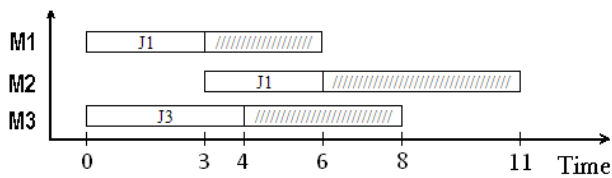


Figure 10: Partial scheduling

At this stage, different parameters are calculated as follows :

- $SomTpsEx = 10$ ut.
- $SomTpsIn = 15$ ut.
- $Cpmax = 11$ ut.

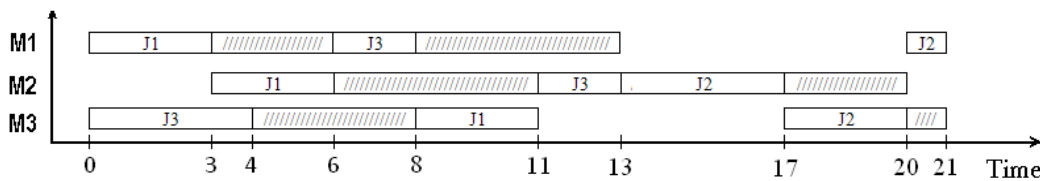


Figure 11: Complete scheduling for $J3||C_{max}$

4 EXPERIMENTAL RESULTS

To evaluate our heuristics we compared our results to lower bounds for RCb^* constraint case and to the optimal solutions for RCb constraint case.

4.1 Constraint RCb^*

In this section, we present obtained results of three problems RCb^* with different dimensions (three jobs - three machines, five jobs - three machines and five jobs - five machines).

From which $Cr(a) = 11 + 15 \div 10 = 16$.

A final solution for this $J3||C_{max}$ example is presented by the Gantt diagram (Figure 11).

Next, we present general structure of used algorithm to solve jobshop problems with RCb and RCb^* constraints.

Algorithm

- 1 Introduce problem data (jobs number, operations number, machines number, routing for each of all jobs and operations execution time)
- 2 For $i = 1$ to n (we calculate all possibilities by switching job to place at first)
 - 2.1 We place first operation of job J_i
 - 2.2 As long as it still remains operations to place, do
 - 2.2.1 For $j = 1$ to n (we study remaining operations)
 - 2.2.1.1 We verify if we can place first operation non-placed O_{kj} of job J_j (we test conditions treated in paragraph 3.2)
 - 2.2.1.2 If all conditions are verified
Calculate $Cpmax$, $SomTpsIn$, $SomTpsEx$
Calculate Cr_j (Criterion after placing a job J_i operation)
End if
 - 2.2.2 Place operation that has the smallest Cr_j
 - 2.3 Calculate $C_{max,i}$ (final makespan while placing job J_i at first)
- 3 Choose scheduling that gives the smallest final makespan $C_{max,i}$

For each problem with fixed dimension, we generated 20 different instances which we know their lower bounds obtained in works of [Gorine 2008]. For each of these instances, routing and execution time were generated randomly. Execution times were generated uniformly in meantime [0, 99].

Table 2 gives mean error percentage between lower bounds and calculated makespan from proposed heuristics, as well as the number of time where C_{max} is equal to corresponding lower bound. Error percentage is calculated as following:

$$\%err = \frac{C_{max} - Binfi}{C_{max}} \times 100$$

equal to optimal solution (once out of two for 3 jobs and 3 machines case) else, mean error is often great and this may be due to quality of our heuristics and may be due to lower bounds.

These results indicate that in some cases, lower bound is

3 jobs and 3machines

Pb.	B _{inf}	C _{max}	Error
1	294	294	0,00%
2	263	263	0,00%
3	75	75	0,00%
4	181	181	0,00%
5	114	117	2,56%
6	186	186	0,00%
7	237	311	23,79%
8	121	121	0,00%
9	116	116	0,00%
10	204	290	29,66%
11	101	101	0,00%
12	188	216	12,96%
13	323	415	22,17%
14	269	269	0,00%
15	326	451	27,72%
16	162	162	0,00%
17	290	364	20,33%
18	270	345	21,74%
19	248	312	20,51%
20	299	363	17,63%
	Mean error	9,95%	
	Binf = Cmax	10	

5 jobs and 3 machines

Pb.	B _{inf}	C _{max}	Error
1	163	199	18,09%
2	314	314	0,00%
3	285	328	13,11%
4	201	247	18,62%
5	259	320	19,06%
6	389	406	4,19%
7	364	459	20,70%
8	554	651	14,90%
9	251	251	0,00%
10	403	486	17,08%
11	375	383	2,09%
12	481	522	7,85%
13	371	371	0,00%
14	320	320	0,00%
15	346	439	21,18%
16	401	415	3,37%
17	418	487	14,17%
18	523	686	23,76%
19	517	606	14,69%
20	439	493	10,95%
	Mean error	11,19%	
	Binf = Cmax	4	

5 jobs and 5 machines

Pb.	B _{inf}	C _{max}	Error
1	414	516	19,77%
2	454	489	7,16%
3	273	314	13,06%
4	346	346	0,00%
5	383	525	27,05%
6	392	535	26,73%
7	399	516	22,67%
8	384	439	12,53%
9	364	461	21,04%
10	359	420	14,52%
11	355	546	34,98%
12	439	580	24,31%
13	450	611	26,35%
14	385	509	24,36%
15	448	498	10,04%
16	549	745	26,31%
17	344	498	30,92%
18	374	476	21,43%
19	431	538	19,89%
20	518	667	22,34%
	Mean error	20,27%	
	Binf = Cmax	1	

Table 2: Error percentage between lower bounds and makespan and number of time where $B_{inf} = C_{max}$

4.2 Constraint RCB

In this section, we present obtained results of two RCB problems with different dimensions (three jobs - three machines and three jobs ó four machines).

By same manner used in preceding section for RCB* constraint case, we consider 30 different instances which are generated by [Gorine 2008] and their optimal solutions are calculated in this same works.

Table 3 gives mean error percentage between optimal solutions and calculated makespan from proposed heuristics, as well as number of time where C_{max} is equal to corresponding optimal solution.

For jobshop problem with constraint RCB, Table 3 results indicate that in several cases, C_{max} is equal to optimal solution else, mean error is sometimes close of optimal solutions, sometimes rather great. We note that when optimal solution is not reached,

the error is often great (17% to 27%), that is may be due to the quality of our heuristic.

5 CONCLUSION

In this paper, we described a jobshop problem with different blocking constraints, in particular RCB and RCB* blocking. We defined different conflicts met at heuristics construction and characteristics of blocking constraints studied in this paper. Then, we proposed heuristics to solve jobshop problems with RCB and RCB* constraints and with objective function minimization of last operation completion time.

We noted that these heuristics are valid for small-size problems and give solutions very close to optimal solutions.

Future research will consist of control all possible conflicts cases in big-size problems, then to develop metaheuristics based on solutions obtained by these heuristics.

3 jobs 3 machines

<i>Pb.</i>	S_{opt}	C_{max}	<i>Error</i>
1	294	294	0,00%
2	263	263	0,00%
3	77	92	16,30%
4	181	181	0,00%
5	117	117	0,00%
6	186	186	0,00%
7	311	311	0,00%
8	121	121	0,00%
9	116	116	0,00%
10	290	290	0,00%
11	101	125	19,20%
12	216	216	0,00%
13	415	415	0,00%
14	269	269	0,00%
15	451	451	0,00%
16	162	162	0,00%
17	184	184	0,00%
18	413	430	3,95%
19	138	138	0,00%
20	298	345	13,62%
21	209	209	0,00%
22	66	66	0,00%
23	500	500	0,00%
24	197	197	0,00%
25	90	90	0,00%
26	268	268	0,00%
27	355	374	5,08%
28	308	308	0,00%
29	297	318	6,60%
30	363	363	0,00%
	Mean error		2,16%
	$S_{opt} = C_{max}$		22

3 jobs 4 machines

<i>Pb.</i>	S_{opt}	C_{max}	<i>Error</i>
1	97	97	0,00%
2	244	244	0,00%
3	425	455	6,59%
4	489	503	2,78%
5	296	296	0,00%
6	336	347	3,17%
7	265	280	5,36%
8	345	345	0,00%
9	314	314	0,00%
10	320	404	20,79%
11	254	254	0,00%
12	508	508	0,00%
13	119	119	0,00%
14	348	359	3,06%
15	231	231	0,00%
16	475	475	0,00%
17	438	462	5,19%
18	158	158	0,00%
19	249	249	0,00%
20	261	261	0,00%
21	188	188	0,00%
22	309	390	20,77%
23	378	393	3,82%
24	191	218	12,39%
25	374	419	10,74%
26	479	504	4,96%
27	391	451	13,30%
28	484	506	4,35%
29	458	458	0,00%
30	477	477	0,00%
	Mean error		3,91%
	$S_{opt} = C_{max}$		16

Table 3: Error percentage between optimal solutions and makespan and number of time where $S_{opt} = C_{max}$

REFERENCES

Blazewicz J., W. Domschke, E. Pesch, 1996. The job shop scheduling problem. *European Journal of Operational Research*, 93, p. 1-33.

Brucker, P. S. Heitmann, J. Hurink, T. Nieberg, 2006. Job Shop scheduling with limited capacity buffers. *OR Spectrum*, vol. 28, pp. 1516176.

Dauzère-Pérès S., C. Pavageau, N. Sauer, 2000. Modélisation et résolution par PLNE d'un problème réel d'ordonnement avec

contraintes de blocage. *3ème congrès ROADEF*, pp. 216-217, Nantes.

Dorigo M., G. Di Caro, 1999. *The Ant Colony Optimization Meta-Heuristic, New Ideas in Optimization*, McGraw-Hill.

Dréo J., A. Petrowski, P. SIARRY, E. Taillard, 2003. *Métaheuristiques pour l'optimisation difficile*, Eyrolles.

- El Bahloul S., 2008. *Flow-shop à deux machines avec des temps de latence : approche exacte et heuristique*. Thèse, Université de Québec.
- Garey M.R., D.S. Johnson, 1979. Computers and Intractability: a guide to the theory of NP-completeness. *Freeman and Company*. San Francisco, CA.
- Gorine A., Nathalie Sauer, 2008. Méthode exacte et bornes inférieures pour le Job Shop avec contrainte de blocage particulière. *Conférence Internationale Francophone d'Automatique*, CIFA, Bucarest.
- Jain A.S., S. Meeran, 1999. Deterministic jobshop scheduling: past, present and future. *European Journal of Operational Research*, vol. 113, n°2, pp. 390-434.
- Martinez S., 2005 *Scheduling de systems de production avec contraintes de blocage*. Thèse, Université de Nantes
- Mati Y., 2002. *Les problèmes d'scheduling dans les systèmes de production automatisés : Modèles, complexité et approches de résolution*. Thèse, Université de Metz.
- Nowicki. E, C. Smutnicki, 1996. A fast tabu search algorithm for the job shop problem. *Management Science*, vol. 42, pp. 797-813.
- Roy B., B. Sussman, 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. *Technical report*, SEMA, Paris, France.
- Van Laarhoven. P.J.M, E.H.L. Aarts, J.K. Lenstra, 1992. Job shop scheduling by simulated annealing. *Operations Research*, vol. 40, n°1, pp. 113-25.
- Volta. R, G. Della Croce, F. Tadei, 1995. A genetic algorithm for the job shop scheduling problem. *Computers and Operations Research*, vol. 22, pp. 15-24.