Exact and Heuristic Methods for Minimizing the Total Completion Time in Job-shops

Yacine Benziani, Imed Kacem, Pierre Laroche, Anass Nagih

Exact and Heuristic Methods for Minimizing the Total Completion Time in Job-shops

Yacine BENZIANI, Imed KACEM, Pierre LAROCHE, Anass NAGIH

LCOMS EA 7306, Université de Lorraine,
Ile du Saulcy, Metz, 57000, France,
{yacine.benziani, imed.kacem, pierre.laroche, anass.nagih}@univ-lorraine.fr

Abstract: In this paper we consider the total completion time minimization in a job-shop. We propose a new mathematical formulation based on a strip packing model. This formulation is enhanced by introducing some valid inequalities in order to compute an efficient lower bound. It is also exploited to derive an exact method; branch-and-bound algorithm which uses an improved solution of a genetic algorithm. The proposed algorithms are tested on standard benchmarks and the results are satisfactory in term of solution quality and the distance to the optimal solution.

Keywords: Job Shop; Total completion time; Scheduling problem; Mixed Integer Programming; Strip Packing.

1. Introduction

The job shop scheduling problem consists in organizing a set J of n jobs to be processed on a set M of m machines. A job j is a set of μ_j operations O_{ij} to be processed on machines without interruption in a predetermined order. The aim is to minimize the sum of the completion times of the last job operations.

The job shop scheduling problem (JSS) is NP-hard even for three machines [1]. It has been widely studied, in particular to minimize the makespan. The first formulation leading to an exact method for the job shop problem was due to Roy and Sussmann [2]. This formulation is based on the disjunctive programming model and the well-known disjunctive graph representation. Carlier and Pinson [3] used this formulation and develop a branch-and-bound method to solve the problem. They used also the one-machine relaxation to obtain a lower bound. The disjunctive programming model can be turned into a mixed integer problem by introducing a binary variable. This new model was used by Applegate and Cook [4]. They used both the disjunctive and mixed integer formulation and applied a cutting plane approach to obtain an initial solution of the problem by solving the relaxed problem, and then they added valid inequalities by dropping the disjunctive constraints and relaxing the binary variables. Another way to model the job shop is the packing formulation which is a model for the job shop feasibility problem. It consists to say "*Does there exist a set of job schedules such that no two schedules require the same machine at the same time*" where the job schedule is an assignment of operation starting times of one job such that no operation starts before the ending of its predecessors. Martin and Shmoys [5] used this formulation to introduce a lower bound based on the fractional packing. The last model is based on the time oriented approach. It was used by Martin [6] and Martin and Shmoys [5] to develop a branch-and-bound algorithm to solve the problem.

In this paper, a new mixed integer programming (MIP) formulation of the job shop is proposed, and some new inequalities are added to obtain a better lower bound. Moreover, two genetic algorithms are tested, and finally the lower bound and the upper bound are used both to improve a branch-and-bound algorithm used to solve optimally the problem. Finally, the numerical experiments and the obtained results are presented and discussed.

2. The MIP Formulation of the 2-Strip Packing

For self-consistency, we recall in this section the definition and the formulation of the two dimensional strip packing (*2SP*). The *2SP* is a combinatorial optimization problem. It consists in packing a set of n items in one bin; this bin has a width W and an infinite height. Each item i has a width w_i and a height h_i. The dimensions of the bin and the items are integers. The objective is to minimize the height of the used strip (bin) to pack all items without overlapping, where w-edges of items have to be parallel to the W-edge of the strip. The *2SP* problem is NP-hard in the strong sense [7].

The mixed integer program used in [8] is an adaptation of the model presented by Pisinger and Sigurd [9] for the 2-Dimesional Bin Packing problem. The 2SP formulation is described as follows. Let E be a set of n items (E={1,2,…,n}) to be packed on the strip having a total width W and a maximum height H_h which can be computed by a heuristic. For example, we can set H_h as follows:

$$H_h = \sum_j h_j, \quad (1)$$

2.1 Decision variables

- (x_i, y_i) denotes the Cartesian coordinates of the lower-left point of item i;
- l_{ij} is equal to 1 if item i is located on the left of item j, 0 otherwise;
- b_{ij} is equal to 1 if item i is located below item j, 0 otherwise.

2.2 Objective function

- H denotes the height of the used strip (i.e., the objective function to be minimized).

2.3 The mixed integer program

The mathematical program is described as follows:

min H subject to:

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} \geq 1 \qquad i,j \in E, i < j, \quad (2)$$

$$x_i - x_j + W.l_{ij} \leq W - w_i \quad i,j \in E, \quad (3)$$

$$y_i - y_j + H_h.b_{ij} \leq H_h - h_i \quad i,j \in E, \quad (4)$$

$$0 \leq x_i \leq W - w_i \qquad i \in E, \quad (5)$$

$$0 \leq y_i \leq H_h - h_i \qquad i \in E, \quad (6)$$

$$H \geq y_i + h_i \qquad i \in E, \quad (7)$$

$$l_{ij}, b_{ij} \in \{0,1\} \qquad i,j \in E, \quad (8)$$

$$H, x_i, y_i \in \mathbb{R} \qquad i \in E, \quad (9)$$

Constraints (2, 3, 4) avoid the overlapping between items (item i should be in the left or in the right side of item j, and it should be below or above or both). In the constraints (3), if l_{ij} = 1 then $x_j \geq x_i + w_i$, and that means item i is located in the left of item j (if l_{ij} = 0 then the constraint is redundant). In the constraints (4), if b_{ij} = 1 then we have item i below item j (if b_{ij} = 0 then the constraint is redundant). Constraints (5) and (6) ensure that the items do not exceed the edges of the strip. Finally, the constraints (7) calculate the used height of the strip which must be great or equal to the position of the upper coordinate of all items.

3. The MIP Formulation of the Job Shop using Strip Packing Formulation

The job shop scheduling problem is defined as follows:

- a set M of m machines and a set J of n jobs where each of them is composed of m operations;
- the processing time of all operations;
- the precedence relation between operations of every job which defines the specified processing order through the machines.

The operations of jobs which have to be processed on the same machine could be considered as items to be packed on a strip. Each operation has a processing time p_i *which* can be seen as a height h_i of item i with a unit width w_i = 1.The total width of the strip is equal to the total number of machines W=m. We add additional constraints to ensure that operations are positioned in the right location according to the machine processing order and the position of machines in the strip.

Let J be the set of all operations where each m first elements in this set define the operations of one job. We also define the set of precedence relations *Prec* between operations of every job.$(i,j) \in Prec$ means that operation i proceeds j. We set l_{ij} = 1 if the machine which executes operation i is located before the machine executing operation j, so that, if $l_{ij} = l_{ji}$ = 0, this means that the two operations run on the same machine (i.e., $a(i)=a(j)$, where $a(i)$ denotes the machine executing operation i).

The variable y_i represents the starting time of operation i. The completion time of operation i is $C_i=y_i+p_i$. According to the execution order of job operations, we set $y_i+p_i \leq y_j$ for each pair of operations *(i, j) in Prec.* The constant $C = \sum_j p_j$ is an upper bound on the maximum completion time, makespan. The value of C represents here the value H_h in the *2SP* formulation.

Here the objective is to minimize the sum of job completion times. The MIP of our problem is written as follows:

$min \sum_j C_j$ subject to

$$b_{ij} + b_{ji} \geq 1 \quad i, j \in J, \quad i < j, \quad a(i) = a(j), \quad (10)$$

$$y_i - y_j + C.b_{ij} \leq C - p_i \qquad i, j \in J, \qquad (11)$$

$$0 \leq y_i \leq C - p_i \qquad i \in J, \qquad (12)$$

$$y_i + p_i \leq y_j \qquad (i, j) \in Prec, \qquad (13)$$

$$b_{ij} \in \{0,1\} \qquad i, j \in J, \qquad (14)$$

$$y_i \in \mathbb{R} \qquad i \in J, \qquad (15)$$

Note that the use of variables b_{ij} avoid here the overlapping between operations performed on the same machine. By relaxing the integrity constraints variables we obtain a linear program which can be solved to optimality by a linear programming algorithm and we obtain the first lower bound lb_0.

4. Lower Bounds for the Job Shop Scheduling Problem (JSSP)

Several researchers investigated the design of lower bounds for the job shop scheduling problem, especially for the makespan criterion. The one-machine relaxation of Carlier and Pinson [3] was the first lower bound used to solve the legendary instance 10 x 10 of Fisher and Thomson [10]. Brucker and Jurisch [11] and Brucker, Jurisch and Sievers [12] developed the two-job relaxation lower bound. This lower bound is obtained by solving the job shop scheduling problem with two jobs for every pair of jobs using the graphical method. In the cutting plan lower bound of Applegate and Cook [4], several cuts have been added to the problem based on one machine relaxation, two job cuts, triangle cuts and half cuts.

In our approach, valid inequalities have been added to the previous formulation to obtain new lower bounds. A preliminary study shows that our technique is effective for the makespan criterion [13]. Such a study motivated us to investigate this approach for minimizing the total completion time.

4.1 Inequalities based on the precedence constraints

If operation j is located above one or more other operations then y_j is greater or equal than the sum of processing times of operations preceding it, and we have:

$$y_j \geq \sum_{i:a(i)=a(j)} p_i b_{ij}, \qquad (16)$$

Let $y'_j = C - y_j - p_j$. If operation j is located below one or more other operations then y'_j is greater or equal to the sum of operations processing time which follow j.

In other words, we have:

$$y'_j \geq \sum_{i:a(i)=a(j)} p_i b_{ji}$$

which is equivalent to:

$$C - y_j - p_j \geq \sum_{i:a(i)=a(j)} p_i b_{ji} \qquad (17)$$

4.2 Inequalities related to the one-machine scheduling problem

This inequality comes from solving the scheduling problem on a single machine with the aim of minimizing the weighted completion time where each job has a release date. This problem, known as $1|r_i|\sum w_i C_i$, is defined for every machine z and it is noted here (π_z). In such a problem, the release date of operation i is obtained by summing the processing times of the predecessor operations according to the precedence constraints. This problem is NP-hard [14], even if $w_i = 1$ or for the preemptive version [15].

On each machine z we have the following inequalities where *opt* defines a function equal to the optimal weighted completion time of the associated problem $1|r_i|\sum w_i C_i$:

$$\sum_{i \in M_z} w_i (y_i + p_i) \geq opt(\pi_z), \qquad (18)$$

It is well-known that the preemptive variant with $w_i=1$ $(1|r_i, pre|\sum C_i)$ is polynomial and we can solve it by applying the shortest remaining processing time (*SRPT*) rule. We obtain the following inequalities:

$$\sum_{i \in M_z} (y_i + p_i) \geq SPRPT(\pi_z), \qquad (19)$$

$$\sum_{i \in M_z} (C - y_i) \geq SRPT(\pi_z), \qquad (20)$$

Remark: Note that in Inequality (20), the release dates are computed by inversing the precedence constraints. It is also the case for the next inequalities (22).

If $w_i = p_i$ The obtained problem $(1|r_i|\sum p_i C_i)$ is also polynomial and it can be solved by First In First Out (*FIFO*) rule and this gives the following inequalities:

$$\sum_{i \in M_z} p_i (y_i + p_i) \geq FIFO(\pi_z), \quad (21)$$

$$\sum_{i \in M_z} p_i (C - y_i) \geq FIFO(\pi_z), \quad (22)$$

4.3 Acyclic three jobs

The acyclic three jobs inequalities ensure that for every three operations *i*,*j* and *k* on the same machine, at least one before the other operations and at most two operations before the last one. Hence, for each triplet of operations which are processed on the same machine we have the following inequalities (indeed, it is easy to see for example that if $b_{ik}=b_{kj}=1$ then $b_{ji}=0$):

$$b_{ik} + b_{kj} + b_{ji} \leq 2, \quad (23)$$

$$b_{ij} + b_{jk} + b_{ki} \leq 2, \quad (24)$$

5. Genetic Algorithm for the Job Shop Scheduling Problem

In this section, we describe a genetic algorithm (GA) with the aim of finding a good upper bound for the considered problem. In particular, we compared two chromosome representations on the total completion time criterion to determine the best one. The most suitable parameters will be then used in this GA in the next algorithms.

The GAs are stochastic methods to find a good solution which can be optimal by using the evolutionary process. We define a population which contains several individuals; each one represents a solution of our problem. We evaluate every individual by a fitness function according to the objective function. At each step of the algorithm we select some individuals and apply the crossover and mutation operators to obtain new individuals. The used GA is based on several elements which are described in the remainder of this section.

5.1 Chromosome representation

The chromosome is a representation of solution, each solution has a unique coding representation and each chromosome corresponds to one solution. It could be binary or integer.

In this paper we use two chromosome representations, the first one is based on *working sequence* proposed in [16] and the second one is a *job sequence* proposed in [17].

1. Working sequence: The chromosome is defined by the occurrence order of the jobs. It is a list of integer variables, each one contains a job. Each job appears m times (m = number of operations).
2. Job sequence: The job sequence representation is an array of m machines; in each line of the array we have the order of the performed jobs on the associated machine.

For example, we take a 3x3 instance of the job shop scheduling problem; the assignment matrix and the chromosomes according to the first and the second coding are given as follows (Tables 1-3).

Table 1. Assignment matrix

Job	**J1**	**J2**	**J3**
Machine	1	2	3
	2	3	1
	3	1	2

Table 2. Working sequence

	Sequence								
Chromosome	3	1	2	3	2	2	1	1	3
Operation	1-3	1-1	1-2	2-3	2-2	3-2	2-1	3-1	3-3
Machine	M_3	M_1	M_2	M_1	M_3	M_1	M_2	M_3	M_2

Table 3. Job sequence

Working	**Sequence**		
M1	1	3	2
M2	2	1	3
M3	3	2	1

5.2 Fitness function

The fitness function is used to evaluate each chromosome according to the objective

function. In our case, the fitness function is the total completion time.

In both, the first and the second representation we set the binary variables b_{ij} in constraints (5) and (6) according to the chromosome sequence and we obtain a linear program which can be solved to optimality by the CPLEX Software (the software solves the linear program with the Barrier Method or Simplex Method). The obtained solution is the evaluation of the chromosome.

5.3 Population

The initial population contains several different solutions. We obtain these solutions randomly and by using priority rules [18]. At each generation, after applying the mutation and crossover we obtain a solution by replacing the old parent by the new one.

- Heuristic and Priority rules: Adams et al. [19] presented the shifting bottleneck heuristic. It consists to solve for each machine the one-machine scheduling problem. The first genetic algorithm was proposed by Davis [20]. Yamada and Nakano [6] use the binary representation and develop a genetic algorithm.

In practice the priority rules defined by the algorithm of Giffler and Thompson [18] are the most used for solving the Job Shop Scheduling Problem since the computational time is very short. There are twelve rules in practice:

- SOT (Shortest Operation Time): is an operation with shortest processing time on the considered machine;
- LOT (Longest Operation Time): is an operation with longest processing time on the considered machine;
- SRPT (Shortest Remaining Processing Time): an operation with shortest remaining job processing time;
- LRPT (Longest Remaining Processing Time): an operation with longest remaining job processing time;
- LORPT (Longest operation Remaining Processing Time): an operation with highest sum of tail and operation processing time;
- Random: an operation for the considered machine is randomly chosen;
- FCFS (First Come First Served): The first operation in the queue of jobs waiting for the same machine;
- SPT (Shortest Processing Time): A job with a smallest total processing time;
- LPT (Longest Processing Time): A job with a longest total processing time;
- LOS (Longest Operation Successor): an operation with a longest subsequent operation processing time;
- SNRO (Smallest Number of Remaining Operations): An operation with a smallest number of subsequent operations in the job;
- LNRO (Largest Number of Remaining Operations): An operation with a largest number of subsequent operations in the job.

5.4 Mutation

The mutation operation is used to diversify the population, so that, we can jump from a local optima to another area in the feasible solutions domain.

In the first representation, we choose randomly two positions in the working sequence, containing two different jobs, and then we swap the selected operations (see Table 4 and Table 5).

Table 4. Chromosome before mutation

	Sequence								
Ch	3	1	**2**	3	2	2	1	**1**	3
Op	1-3	1-1	1-2	2-3	2-2	3-2	2-1	3-1	3-3

Table 5. New solution after applying the mutation

	Sequence								
Ch	3	1	**1**	3	2	2	1	**2**	3
Op	1-3	1-1	2-1	2-3	1-2	2-2	3-1	3-2	3-3

In the second representation, we choose randomly one machine and two operations on this machine and then we swap the two selected operations. A new job sequence, which can be not feasible solution, is obtained. Note that, we can choose two operations on each machine in the selection step.

5.5 Crossover

After choosing two arbitrary individual from the population, we apply the crossover operator. We choose an arbitrary position in the chromosome of the first parent and we build the new individual by taking the first part from the first parent and we complete the chromosome by the genes appearing in the second parent. We apply the same technique to the two parents in order to obtain two new individuals. In the example below, we generate two different solutions.

Table 6. Parent 1

	Sequence								
P1	**3**	**1**	**2**	3	2	2	1	1	3
Op	**1-3**	**1-1**	**1-2**	2-3	2-2	3-2	2-1	3-1	3-3

Table 7. Parent 2

	Sequence								
P2	**1**	**3**	**1**	2	3	3	2	2	1
Op	**1-1**	**1-3**	**2-1**	1-2	2-3	3-3	2-2	3-2	3-1

The two new solutions are described in Table 8 and Table 9.

Table 8. Child 1

	Sequence								
Ch1	3	1	2	**1**	**3**	**3**	**2**	**2**	**1**
Op	1-3	1-1	1-2	**2-1**	**2-3**	**3-3**	**2-2**	**3-2**	**3-1**

Table 9. Child 2

	Sequence								
Ch2	1	3	1	**2**	**3**	**2**	**2**	**1**	**3**
Op	1-1	1-3	2-1	**1-2**	**2-3**	**2-2**	**3-2**	**3-1**	**3-3**

In the job sequence coding, either for the crossover and mutation operators, the new solutions could be infeasible. The non-feasibility of the solutions is due to the precedence constraints violation. To transform the non-feasible schedule into a feasible one we apply the Giffler and Thompson technique [18].

5.6 The genetic algorithm

The genetic algorithm runs as follows. First, we generate randomly and by following the priority rules all individuals; in our implementation we use 500 individuals. In each generation we choose arbitrary, from some good solutions, two individuals who represent the parents. On these parents we apply the mutation and the crossover operators with a fixed probability, 0.1 for mutation and 0.8 for crossover, to obtain two new children, if the new individuals are not feasible, we covert them into feasible individuals. The algorithm stops when the stopped condition is satisfied, in our test we stop after 2000 iterations. Note that the GA version using the *job sequence* representation is denoted as AG1. The GA incorporating the *working sequence* representation is denoted AG2.

6. Branch-and-Bound Algorithm

To solve the problem to the optimality we use the branch-and-bound approach: the problem (the first node) is evaluated and then separated in sub problems (branching) which constitute the next nodes, and each node is evaluated and separated if necessary until the optimality of the best solution is proved. The algorithm is based on the use of the mathematical model enhanced by the valid inequalities.

6.1 Evaluation

The problem is evaluated by solving the relaxed problem (i.e., the relaxed model enhanced by the introduced inequalities); the lower bound obtained is the evaluation of our problem. If the solution is feasible for the job shop scheduling problem then the node will not be separated.

The node used to be evaluated can be selected with two strategies, the best bound first or the depth first search.

6.2 Branching

The solution of the relaxed problem can be not feasible if one or more variables b_{ij} is not binary, in this case the problem will be separated following the fractional variables.

Two sub-problems are created. In the first one we fix $b_{ij} = 1$ and $b_{ji} = 0$. In the second sub-problem we fix $b_{ij} = 1$ and $b_{ji} = 0$. The sub-problems provide the new nodes (branches).

The variables to be separated are selected with different strategies: the minimum fractional, the maximum factional or the nearest from ½.

If the fractional variable does not exist then the solution is feasible for the problem and constitutes an upper bound.

7. Computational Results

In this section we describe the numerical tests we carried out and the different obtained results. We apply our algorithms to the instances described in the benchmark *OR-Library* (dedicated to the makespan minimization) and we change the objective function to the total completion time criterion. More precisely, we use the instances of Fisher and Thompson [10] (MT6, MT10 and MT20), the instances of Adams, Balas and Zawack [19] (ABZ5, 6 and 7), instances of Applegate and Cook [4] (ORB1 to OBR9) and finally some instances of Lawrence (LA1, LA6, LA16, LA21) [21].

Table 10 contains the name of instances, the lower bound without cuts lb_0, and the lower bound including the valid inequalities lb_c (these bounds are obtained by solving the linear program with relaxed integrity constraints of variables b_{ij}), the solution obtained by the first genetic algorithm (AG1) and with the second

Table 10. Results

Instances	lb_0	lb_c	AG1	AG2	B&B	Gap (lb_c, AG2)
MT6	169	220	267	237*	237*	7.2%
MT10	4651	6073	8785	8646	7002	13.27%
MT20	4208	10941	17101	16708	13939	21.51%
ABZ5	6957	8630	11487	10739	9891	12.75%
ABZ6	5244	5986	8848	7761	7171	16.52%
ABZ7	6907	9163	16311	13814	12890	28.91%
ABZ8	7033	8877	17713	14162	13260	33.05%
ORB1	4803	6389	10013	8657	7692	16.94%
ORB2	4746	5796	8675	7867	6866	15.58%
ORB3	4663	6059	9780	9135	7650	20.80%
ORB4	5022	6305	9998	8742	7287	13.48%
ORB5	4389	5789	8020	7772	6468	10.50%
LA1	2186	3760	5272	4568	4174	9.92%
LA6	3208	7140	9860	9636	8095	11.80%
LA16	4686	5808	8447	7793	6783	14.37%
LA21	7137	9572	16144	14588	13596	29.60%

genetic algorithm (AG2) and the solution obtained by the branch-and-bound algorithm (B&B) after five minutes of execution. The last column contains the gap between the best upper bound (AG2) and the best lower bound (lb_c).

the branch-and-bound algorithm in few seconds. However for instances with more than 10 machines and 10 jobs the gap is more important and significant computational time is needed to solve the problem. Nevertheless,

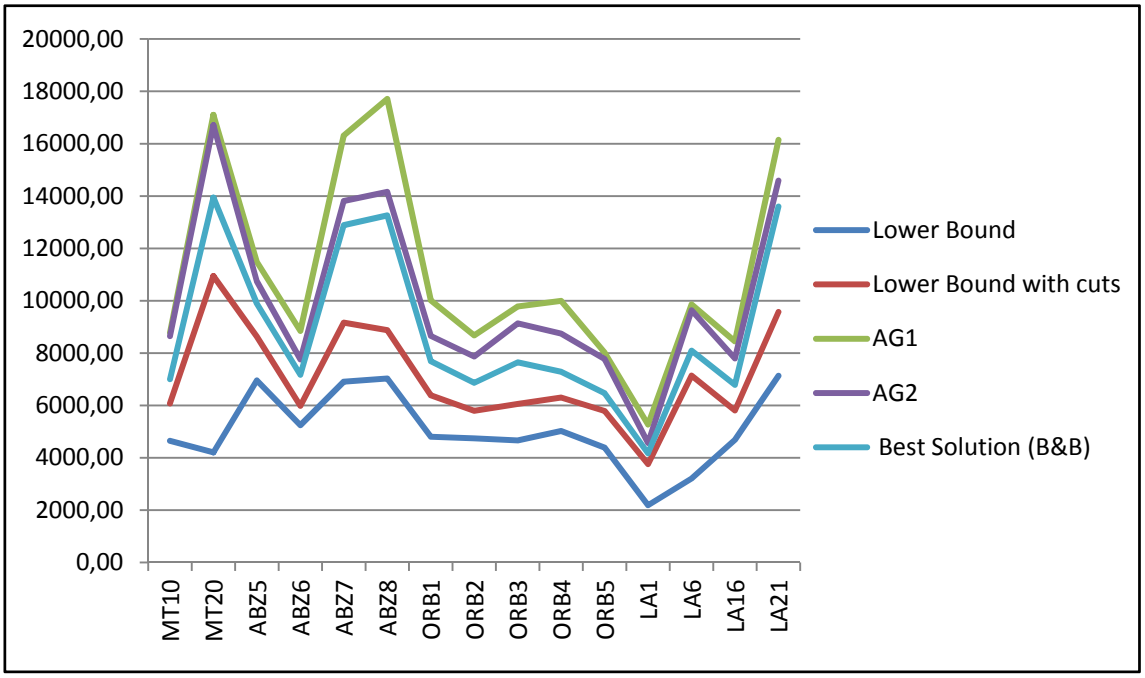

Figure 1. Graphical representation of results.

Figure 1 describes the gap between the different values of the lower bounds, the values of the two genetic algorithms and the best solutions (obtained by the B&B).

First, we observe that the second algorithm (AG2) outperforms the first one (AG1). This is due to the fact that AG1 generates non-feasible solutions after applying the crossover and mutation operators. Hence a part of the running time is lost instead of an effective exploration of the solution space. Contrarily, AG2 generates only feasible solutions and its exploration is more effective.

The results related to the lower bounds performances show that the valid inequalities improve significantly the obtained performances. The computed gap is better for the lower bound incorporating these inequalities. The gap between the best solution of the two algorithms (GA and GA2) and the lower bound (lb_c) is about 25% in average, which means that one of the two bounds is at most of 12,5% from the optimal solution.

For the small instances, up to 9 machines and 9 jobs, the problem is solved to optimality by the obtained results show that the branch-and-bound proves the satisfactory performance of GA2 and the promising effectiveness of lower bound lb_c. More effective bounds can be obtained by introducing further improvements to GA2 and lb_c.

8. Related Problems

Our algorithms can be applied to other close related problems. As possible extensions we present the two following problems.

8.1 Job shop with no-wait constraints

In this case, the job operations must be processed without interruption, so that consecutive operations of job j will be continuously processed one after the other.

To do so, the constraints (13) $y_i + p_i \leq y_j, (i, j) \in Prec$ should be replaced by:

$$y_i + p_i = y_j, (i, j) \in Prec, \quad (25)$$

8.2 Flow Shop

In the flow-shop scheduling problems, the jobs have the same order to follow on the machines. The change is occurred only on the data of the problem and the different algorithms are directly applied.

9. Conclusion

In this paper, a new mathematical formulation of the job shop problem is proposed to deal with total completion time minimization, and some new inequalities are added to obtain an effective lower bound. Two genetic algorithms based on different chromosome representations are also analyzed and tested. Moreover, the lower bound and the upper bound are used both to elaborate a branch-and-bound algorithm in order to solve optimally the considered problem. Finally, the numerical experiments and the obtained results are presented and discussed. The results show the satisfactory effectiveness of our bounds and our algorithms for this hard optimization.

As a future perspective we aim to improve the branch-and-bound algorithm by possibly introducing new inequalities to the mathematical formulation and a better upper bound. Moreover, the extension of this work to the open shop problem and the investigation of the real-time context [22] seem to be pertinent questions.

REFERENCES

1. GAREY, M. R., D. S. JOHNSON, R. SETHI, **The Complexity of Flowshop and Jobshop** S**cheduling,** Mathematics of Operations Research, vol. 1, issue 2, 1976, pp. 117-129.
2. ROY, B., B. SUSSMANN, **Les Problèmes d'Ordonnancement avec Contraintes Disjonctives**, Note ds, vol. 9, 1964.
3. CARLIER, J., E. PINSON, **An Algorithm for Solving the Job-Shop Problem**. Management Science, vol. 35, no. 2, 1989, pp. 164-176.
4. APPLEGATE, D., W. COOK, **A Computational Study of the Job-Shop Scheduling Problem**, ORSA Journal on Computing, vol. 3(2), 1991, pp. 149-156.
5. MARTIN, P., D. B. SHMOYS, **A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem**, In Integer Programming and Combinatorial Optimization, Springer Berlin Heidelberg, 1996, pp. 389-403.
6. MARTIN, P. D., **A Time-Oriented Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem**, Cornell University, Ithaca, NY, 1996.
7. Ben MESSAOUD, S., C. CHU, M. L. ESPINOUSE, **Characterization and Modelling of Guillotine Constraints**, European Journal of Operational Research, vol. 191(1), 2008, pp. 112-126.
8. BEKRAR, A., I. KACEM, **An Exact Method for the 2D Guillotine Strip Packing Problem**, Advances in Operations Research, vol. 2009, Article ID 732010, 20 pages, 2009.
9. PISINGER, D., M. SIGURD, **The Two-Dimensional Bin Packing Problem with Variable Bin Sizes and Costs**, Discrete Optimization vol. 2(2), 2005, pp. 154-167.
10. FISHER, H., G. L. THOMPSON, **Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules**, Muth, J. F., Thompson, G. L. (Eds.), Industrial Scheduling. Prentice-Hall, Englewood Cliffs, NJ, pp. 225-251.
11. BRUCKER, P., B. JURISCH, **A New Lower Bound for the Job-Shop Scheduling Problem**, European Journal of Operational Research, vol. 64, no. 2, 1993, pp. 156-167.
12. BRUCKER, P., B. JURISCH, B. SIEVERS, **A Branch and Bound Algorithm for the Job-Shop Scheduling Problem**, Discrete Applied Mathematics, vol. 49(1), 1994, pp. 107-127.
13. BENZIANI, Y., I. KACEM, P. LAROCHE, A. NAGIH, **Lower Bounds for the Makespan Minimization in Job Shops**, In Networking, Sensing and Control (ICNSC), 10th IEEE International Conference on. IEEE, 2013, pp. 442-445.
14. LENSTRA, J. K., A. R. KAN, P. BRUCKER, **Complexity of Machine Scheduling Problems**, Annals of Discrete Mathematics, vol. 1, 1977, pp. 343-362.

15. LABETOULLE, J., E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN, **Preemptive Scheduling of Uniform Machines Subject to Release Dates**, in Progress in Combinatorial Optimization, W. R. Pulleyblank, ed., Academic Press, New York, 1984, pp. 245-261.

16. LI, Y., Y. CHEN, **A Genetic Algorithm for Job-Shop Scheduling**, Journal of Software, vol. 5(3), 2010, pp. 269-274.

17. YAMADA, T., R. NAKANO, **Genetic Algorithms for Job-Shop Scheduling Problems**, Proceedings of Modern Heuristic for Decision Support, UNICOM seminar, 1997, pp. 67-81.

18. GIFFLER, B., G. L. THOMPSON, **Algorithms for Solving Production Scheduling Problems**, Operations Research, vol. 8(4), 1960, pp. 487-503.

19. ADAMS, J., E. BALAS, D. ZAWACK, **The Shifting Bottleneck Procedure for Job Shop Scheduling**, Management Science, vol. 34(3), 1988, pp. 391-401.

20. DAVIS, L., **Job Shop Scheduling with Genetic Algorithms**, Proceedings of the 1st International Conference on Genetic Algorithms. 1985, pp. 136-140.

21. LAWRENCE, S., **Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques**, (Supplement), PhD. Thesis Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1984.

22. FILIP, F. G., G. NEAGU, D. A. DONCIULESCU, **Job Shop Scheduling Optimization in Real-Time Production Control**, Computers in Industry, vol. 4(4), 1983, pp. 395-403.