Les Annales RECITS (http://www.lrecits.usthb.dz)
Vol. 3, 2016, pages 13-26

A Tabu Search for the Bicriteria Scheduling Problem of Job-Shop with Blocking

Karima BOUIBEDE, Mohamed LAKEHAL and Abdelhakim AIT ZAI

Faculty of Electronics and Computer Science
University of Sciences and Technology HOUARI BOUMEDIENNE
Algiers, Algeria

ka.hocine@gmail.com, mlakehal@hotmail.com, h.aitzai@usthb.dz

Abstract: In the literature a very large number of solution methods have been developed for the job-shop scheduling problem, but a majority of them deal with minimization of one criterion. In this paper, we propose to solve a bicriteria scheduling problem of Job-Shop with Blocking that is issued from Metallurgy and Steel Industry. Two tabu search algorithms are proposed for finding a set of non-dominated solutions such that the makespan and the maximum lateness are simultaneously minimized: the first consists in the minimization of one criterion subject to a bound on the second criterion ($\varepsilon - constraint$ approach) and the second is based on the minimization of a linear combination of criteria. These algorithms are tested on benchmark instances from the literature and the results are discussed.

Keywords: jobshop, bicriteria; tabu search; scheduling, blocking.

1. Introduction

The considered problem in this paper is issued from Metallurgy and Steel Industry. In this type of industry, workshops are mainly compound of necessary resources to manufacture many products. In these workshops, resources are the machines that can be used to process different types of operations; the resource processes an operation and can be selected in a set of resources which is associated to the operation. Each job is a set of operations. In order to perform a job, it is needed to process all the operations in a specific order according to the job. The solution aims to find the starting time for each operation to calculate the completion time of each job. This scheduling is done in order to minimize the makespan criterion. The considered problem is named "a general job shop scheduling problem". The software that solves this job shop scheduling problem is used to help decision-makers to organise the production chain from order taking to delivery while minimizing the completion time of the last job, so it considers the minimization of the makespan.

The solution given by the used software assumes that the capacity of the inter-machines buffer is infinite. However, the company decides to remove the inter-machines buffer for profitability reasons. Indeed, there are no such buffers at all in many situations at industry that leads to the blocking constraints; i.e. if the next machine for a given operation is not free, this operation will remain pending on the current machine even if its processing is over. In this case, we say that the machine is blocked. To analyze the impact of the elimination of inter-machine buffer, we must take the minimization of the maximum of lateness criterion in consideration beside the makespan criterion.

In this article, we deal with the bi-criteria job shop scheduling problem that minimizes both the completion time of the last job and the maximum of lateness. Minimizing the makespan criterion tends to shorten the lead time of the whole production, this may increase the productivity of the shop because machines are quickly available for a new run of jobs. Thus, this criterion reflects an internal measure of efficiency for a firm, whilst the maximum lateness reflects the external measure of efficiency. This last criterion can model situations where the firm has to meet delivery dates which correspond to agreements with the customers. Thereby, minimizing this criterion increases the quality of the firm from the customer viewpoint. The presented work concerns the implementation of the algorithms that consider both the elimination of inter-machine buffer and the bicriteria aspect in the used software.

Using the standard three-field notation of scheduling problems [14], this problem can be referred as $J/d_j, blck/C_{max}, L_{max}$.

Although there is a great number of papers that deals with the job shop problem with blocking constraints in literature, to the best of our knowledge no work related to the $J/d_j, blck/C_{max}, L_{max}$ problem has been done.

In Hall and Sriskandarajah [6], we can find a survey on machine scheduling problems with blocking and no-wait constraints. In Candar [3], several applications of machine scheduling with blocking and no-wait in process are described. In Mascis and Pacciarelli [9], a branch-and-bound method and three specialized dispatching heuristics are proposed for several types of job shop problems (classical, with blocking and no-wait). Pacciarelli [12] , used alternative graphs to describe a complex scheduling problem in a steelmaking plant and proposed a dispatching heuristic to solve this problem. Meloni, Pacciarelli and Pranzo [11] presented a rollout metaheuristic for blocking and no-wait job shop scheduling problem which is based on

an alternative graph formulation. The authors showed that their results outperform the A. Mascis, D. Pacciarelli's results [8]. In Mati et al. [10], a tabu search based on a geometric approach, is proposed for a multi-resource job shop problem with blocking constraints. They also proposed a tabu search which applies a permutation moves on the critical path. In Brizuela et al. [2] a genetic algorithm for solving no-wait and Blocking Job Shop problems is presented. Gröflin and Klinkert [4] introduced a generalized disjunctive graph framework for modeling various types of scheduling problems. They proposed a local search approach for the generalized Blocking Job Shop problem. In AitZai et al [1], authors proposed a new parallel branch and bound method and developed a sequential and parallel heuristics based on Particle Swarm Optimization (PSO). InC. Zeng et al [16] two integer non-linear programming models are proposed to describe the Blocking job shop in an automated guided vehicles problem. A two-stage heuristic algorithm that combines between an improved time tabling method and a local search algorithm is proposed to solve the problem.

We propose to heuristically solve this problem by relying on two tabu search algorithms ([13]). The first one is the algorithm that minimizes a linear combination of the criteria and the second minimizes the first criterion, the makespan, subject to a bound on the second criterion, the maximum lateness. The remainder is organized as follows: Section 2 presents the definition of the studied problem. The section 3 explains the used tabu search algorithms. Computational results are presented and discussed in section 4. Conclusions and future works are given in Section 5.

We propose to heuristically solve this problem by relying on two tabu search algorithms ([13]). The First one is the algorithm that minimizes a linear combination of the criteria $\alpha \times C_{max} + \beta \times L_{max}$and the second minimizes the first criterion, the makspan, subject to a bound on the second criterion, the maximum lateness. The remainder is organized as follows: Section II presents the two tabu search algorithms. Computational results are presented and discussed in section III. Conclusions and future works are given in Section IV.

2. Problem definition and notations

The Job shop scheduling problem with blocking is an extension of the classical job shop. It can be defined as follows: the J is considered as a set of n jobs, denoted by $J = \{j_1, j_2, .., j_n\}$ It must be scheduled on a set $\mathcal{M}$ of m machines, denoted by $\mathcal{M} = \{M_1, M_2, \dots.., M_m\}$ Each job J_j is a sequence of n_j operations, $j_j = \left\{O_{j,1},\ O_{j,2},\ O_{j,3}, \dots, O_{j,n_j}\right\}$, which have to be processed on each machine at most once with no storage space and according to a given order. An operation $O_{i,j}$ requires $p_{i,j}$ time units and must be executed without any interruption. To each job J_j a due date d_j is associated. A due date d_j is a date before which job j_j has to leave its last machine. The aim is to determine a starting time for each operation to calculate the completion time C_j of job J_j. This scheduling is done in order to minimize two criteria. The first one, called as the makespan. The second criterion is the maximum lateness of jobs. In this problem blocking constraint means that after completion of operation $O_{j,i}$, that operation remains on its machine until the next machine of operation $O_{j,i+1}$ becomes available for processing $O_{j,i+1}$.

Using the standard three-field notation of scheduling problems [14] this problem can be referred as $J/d_j, blck/C_{max}, L_{max}$

The aim is to find the so-called Pareto optimal solutions or non-dominated solutions which can be formally defined as follows. Let S be the set of feasible schedules. A schedule $s \in S$ is a strict Pareto optimum if and only if there does not exist another schedule $s' \in S$ such that $(C_{max}(s') \leq C_{max}(s)$ and $L_{max}(s') < L_{max}(s))$ nor $(C_{max}(s') < C_{max}(s)$ and $L_{max}(s') \leq L_{max}(s)$). The set of strict Pareto optimum is denoted by $E \subseteq S$. The criteria vector associated to a strict Pareto optimum is called a non dominated criteria vector and the set of such vectors is denoted by . Notice that $|ND| \leq |E|$. To solve the bicriteria problem we enumerate set ND by calculating one strict Pareto optimum per non dominated criteria vector. The calculation of the strict Pareto optima for this problem is strongly NP-hard, because $|ND|$ can be pseudo-polynomially bounded [15] and contains the optimal solution of $J/d_j, blck/C_{max}, L_{max}$ which is strongly NP-hard.

3. The tabu search algorithms

A tabu search algorithm requires an initial solution and a neighbourhood structure and works by transiting from one solution to another using moves. All the neighbours of a current solution are examined and the best non forbidden move is selected. Note that this move may decrease the quality of the solution. A tabu list stores all the previous exploited moves which are now prohibited. The quality of the tabu search is dependent on the following parameters: the initial solution, moves, searching strategy, number of iterations and size of the tabu list, stopping criteria. In this work we conduct preliminary experiments to fix the values of number of iterations and the size of the tabu list.

In this section, we start by presenting the common elements of the two proposed tabu search algorithms, the first one is ε-constraint approach (TS_{ε}) and the second is the aggregation criteria approach (TS_{CL}), after that we continue with the specific parts for each of them. Note that these approaches were applied by Vilcot et al ([13]) for a multicriteria flexible job scheduling problem.

3.1 Common parts of both algorithms

The common parts of the two tabu search algorithms are:

a. Coding of a solution

The encoding is an important determinant of the efficiency of the tabu search algorithm. In this work we adopted the encoding strategy based on priority rules. The advantage of this encoding is that when we generate neighbours, all the created solutions are feasible. A schedule s can be calculated by the mean of a sequence of the priority rules, denoted by $seq = \{R_1, R_2, \dots, R_{Nb_{Operations}}\}$, where R_i are priority rules, $i = 1 \dots Nb_{Operations}$. The length of seq is equal to the number of operations. For example, the sequence $seq = (LPT, LFT, EFT, EDD, LST, EST, SPT, LWKR)$ represents priority rules vector of a schedule for a problem with eight operations.

Several priority rules exist in literature we have used in our work in all ten priority rules which are widely used. You find the description of each priority rule in figure 2. To construct a schedule s, based on sequence seq, the used decoding algorithm (ref. Figure.1) is a building method in which the choice of the next operation to be scheduled is guided at each step by the priority rules of the current sequence seq.

Input: A sequence $seq = \{R_1, R_2, \dots, R_{Nb_{Operations}}\}$ where R_i are priority rules, $i = 1 \dots Nb_{Operations}$;
Output : Feasible schedules s.
Let C be the set of all ready operations;
For ($i = 1 \dots Nb_{Operations}$) do
Select one operation from C in accordance with $seq[i]$.
1. Determine the corresponding machine;
2. If (the machine is free) then Allocate it
else Do a Blocking
3. Upgrade: set C of ready operations.
End
Schedule the last operation;
End

Figure 1. Decoding algorithm

Rule	Description
EST	Earliest Starting Time: The next scheduled operation is the one whose starting time comes prior to those of other operations.
LST	Latest Starting Time: The next scheduled operation is the one whose starting time comes posterior to those of other operations.
EFT	Earliest Finish Time: The next scheduled operation is the one whose finishing time comes prior to those of other operations.
LFT	Latest Finish Time: The next scheduled operation is the one whose finishing time comes posterior to those of other operations.
SPT	Shortest Processing Time: The next scheduled operation is the one whose duration is less than those of other not yet scheduled operations.
LPT	Longest Processing Time: The next scheduled operation is the one whose duration is larger than those of other not yet scheduled operations.
MOPNR	Most Operations Remaining or MTR Most Tasks Remaining: The next scheduled operation is the first not yet scheduled operation of the job containing the largest number of not yet scheduled operations.
MWKR	Most Work Remaining or LRT Longest remaining processing Time: The next scheduled operation is the first not yet scheduled operation of the job with the longest remaining processing time.
LWKR	Least Work Remaining or SRT Shortest Remaining processing Time: The next scheduled operation is the first not yet scheduled operation of the job with the shortest remaining total processing time.
EDD	The next scheduled operation is chosen between the ready operations with earliest due date.

Figure 2. Priority rules

b. The initial solution

The initial solution is randomly generated. A sequence seq is defined by a vector with $Nb\_Operations$ priority rules. To construct the sequence seq, we first affect an integer number between 1 to 10 to each priority rule. Then $Nb\_Operations$ values are randomly generated from the set {1,2,3,4,5,6,7,8,9,10} by using uniform distribution. Finally, we assign to each value the corresponding priority rule. We use the decoding algorithm to determine a schedule s. The main advantage of this encoding is that when we change the priority rules in seq to get a new sequence (a new solution), we never find non-feasible solutions.

c. Neighbourhood

Three types of move are used to generate a neighbourhood: the insert move, the swap move and the shift move.

- The Insert move consists of replacing randomly one priority rule $seq[i]$ in the sequence seq by another priority rule R_q which is randomly selected from the set of the used priority rules, with $seq[i] \neq R_q$.
- The swap move consists of exchanging the positions, that are randomly chosen, of two priority rules $seq[i]$ and $seq[j]$ in the current sequence seq, with $i \neq j$.
- The shift move works as follows :

Let p be an integer drawn at random using an uniform law between 2 and $\mathrm{Nb\_Operations} - 1$. The new sequence is:

$$\begin{cases} [seq'[i]]_{1 \leq i \leq Nb_{Operations}-p} = [seq[j]]_{p<j \leq Nb_{Operations}} \\ [seq'[i]]_{Nb_{Operations}-p<i \leq Nb_{Operations}} = [seq[j]]_{1 \leq j \leq p} \end{cases}$$

d. Tabu list

The tabu list is used to prevent the search from cycling between solutions. Our tabu list has a fixed size, and when the list is full, the oldest element of the list is replaced by the new element. As mentioned before, the size of the tabu list that was used in our algorithm is fixed by the mean of preliminary experiment.

3.2 Specific parts

a. Tabu search with $\varepsilon - constraint$ approach

This tabu search consists of iteratively solving ε-constrained problems. The ε-constrained problem considered in this paper is the following: we minimize the makespan under the constraint $L_{max} < \varepsilon$. where ε is any given value. Initially, ε is fixed to the higher integer value, and at each step ε will be equal to the maximum of lateness value of the optimal solution of the previous ε-constrained problem. The corresponding scheduling problem is referred as $J/d_j, blck/\varepsilon(C_{max}/\ L_{max})$. In this approach, the best neighbour is the neighbour that is not in the tabu list and which respects the following selection strategy:

- If it exists at least one neighbour that respects the ε bound, we chose as the best neighbour the one that minimizes the makespan and verifies the constraint: $L_{max} < \varepsilon$ as well.
- If there is no neighbour that respects the ε bound and if one neighbour at least exists and improves the C_{max} value of the current solution. We consider as the best neighbour the one that minimizes the L_{max} criteria and which is the nearest to the C_{max} value of the current solution.
- If there is no neighbour that respects the ε bound and if there is no neighbour that improves the C_{max} value of the current solution. We consider as the best neighbour the one which minimizes the L_{max} criteria.

b. Tabu search with linear combination

In this approach a neighbour is evaluated by the following linear combination of criteria C_{max} and L_{max}: $\alpha \times C_{max} + \beta \times L_{max}$. The tabu search algorithm is used for a fixed value of α and β. The best neighbour is the neighbour that is not in the tabu list and with minimum value of the linear combination.

4. Computational experiments

4.1 Due date generation

Fourteen standard job shop benchmark instances are used to evaluate the tabu search algorithms. To introduce due dates in these instances, we have proceed as follows :

For each job Ji, its due date dj is drawn at random between using a uniform distribution law.

$\left(L - \frac{R \times \bar{P}}{2}\right)$ and $\left(L + \frac{R \times \bar{P}}{2}\right)$

With $\bar{P} = \sum_{i=1}^{n_j} p_{ij}$ and $L = \bar{P} \times \left(1 + \frac{S \times n}{m}\right)$

L and R are used to rule the distribution of the due dates and S is a factor which reflect the importance of the due dates. Factor L is used to center the distribution of due dates whilst factor R rules their spreading. . For each problem size (n, m), different values of factors S and R are tested: $R, S \in \{0.2 , 0.4 , 0.6 , 0.8 , 0.8 , 1.0\}$. The values of R and S are resp. fixed to 0.6 and 0.4.

4.2 Preliminary experiments

The two tabu search algorithms have a given set of parameters that influence their effectiveness. Before comparing these two algorithms we have conducted early experiments in order to determine the best values of the two tabu serach parameters. We have used a set of eight instances with different sizes, $\mathrm{T} = \{\mathrm{La05}, \mathrm{La10}, \mathrm{La15}, \mathrm{La20}, \mathrm{La25}, \mathrm{La30}, \mathrm{La35}, \mathrm{La40}\}$, to do this preliminary tests and we have considered the C_{max} minimization problem . Indeed, for each algorithm, we have to provide $Iter_{Max}$ and $\mathrm{Size}_{\mathrm{Tl}}$ parameters:

- $Iter_{Max}$ Corresponds to the number of iterations without improvement before the search is stopped. The tested values of this parameter belong to the set $\{1000,2000,3000,4000\}$.
- $Size_{Tl}$ Defines the size of the tabu list. The tested values of this parameter belong to the set $\{20,40,60,80,100,120,140,150\}$

To fix the tabu list size parameter, we proceed by counting the number of times for which the parameter $Size_{TL}$ gives the smallest makespan on the eight instances (cf. Table 1) for the TS_{CL}, $\#Best_{Cl}$, and the TS_{ε}, $\#Best_{\varepsilon}$. Here, the used number of iteration $Iter_{Max}$ is equal to 5000.

$Size_{Tl}$	20	40	60	80	100	120	140	150
$\#Best_{Cl}$	0	0	0	1	4	2	3	3
$\#Best_{\varepsilon}$	2	1	3	1	3	1	2	3

Table 1. Size of the tabu list for TS_{CL} and TS_{ε}

.As shown in the Table 1, the best result is given by $Size_{Tl}$=100 for the tabu search with linear combination. For the tabu search with $\varepsilon - constraint$ approach, the best result is given by $Size_{Tl}$=60, 100 and 150. So for TS_{CL} algorithm, the $Size_{Tl}$ value is fixed to 100. But for the TS_{ε} algorithm, the $Size_{Tl}$ value is fixed to 60.

For some cases the best solution can be obtained for several values of the tabu list size. For this reason, we can remark that the sum of all values of the best solutions is greater than the used number of instances.

The Table 2, resp. the Table 3, shows the results of the preliminary tests for defining the value of $Iter_{Max}$ for the TS_{CL}, resp. TS_{ε}, algorithm. These tables present the average makespan improvement, which is calculated by the formula (1), and the average CPU increase, which is calculated by formula (2) and $C_{max}(Iter_2) \leq C_{max}(Iter_1)$ which indicates the number of times (over 8) where the number of iteration $Iter_2$ performs better than to the number of iteration $Iter_1$.

$$C_{imp} = \frac{1}{|T|} \times 100 \sum_{i \in T} \frac{C^i_{max}(Itr_1) - C^i_{max}(Itr_2)}{C^i_{max}(Itr_1)} \dots\dots\dots\dots\dots\dots (1)$$

$$CPU_{incr} = \frac{1}{|T|} \times 100 \sum_{i \in T} \frac{CPU^i(Itr_2) - CPU^i(Itr_1)}{CPU^i(Itr_2)} \dots\dots\dots\dots\dots\dots . (2)$$

$Iter_1$	$Iter_2$	C_{imp} (%)	$C_{max}(Iter_2) \leq C_{max}(Iter_1)$	CPU_{incr}(%)
1000	2000	0.922	6	51.844
2000	3000	0.178	6	33.070
3000	4000	0.508	1	26.567
4000	5000	0.131	1	19.994

Table 2. $Iter_{Max}$ for the TS_{CL} algorithm

We can see in the Table 2, that increasing the number of iterations, from 1000 to 2000, without improvement gives the greater improvement of 0.922% and the makespan value is six times better in $\text{Iter}_2 = 2000$ than $\text{Iter}_1 = 1000$. But the CPU time is 51.84% longer. The increasing of the iterations number From 2000 to 3000 gives a small value of C_{imp} (%) value and the number of solutions which are improved from the $Iter_1$ to the $Iter_2$, equal to 6,is interesting. But the CPU time is 33.07% longer. We can also see that increasing the number of iteration from 3000 to 4000 and from 4000 to 5000 improve just one time the makespan value with a value of C_{imp} (%) equal to 0.508%, resp. 0.131%, for the increasing from 3000 to 4000, resp. from 4000 to 5000. But the corresponding percentage of increasing CPU time is more reasonable for the last value of the increasing of the number of iteration. So we have chosen to fix $Iter_{Max}$ to 5000 for TS_{CL}

$Iter_1$	$Iter_2$	C_{imp} (%)	$C_{max}(Iter_2) \leq C_{max}(Iter_1)$	$CPU_{incr}(\%)$
1000	2000	0.704	6	51.599
2000	3000	1.448	6	33.277
3000	4000	0.645	4	35.745
4000	5000	0.958	4	19.890

Table 3. $Iter_{Max}$ for the TS_ε algorithm

We can see, in the Table 3, that increasing the number of iterations gives a small improvement of makespan percentage. But the number of solutions which are improved from the $Iter_1$ to the $Iter_2$ is interesting and the corresponding percentage of increasing CPU time is reasonable for the last ligne of the Table 3. So we have chosen to fix $Iter_{Max}$ to 5000 for TS_ε.

4.3 Computational results

As mentioned previously, no works exists in the literature for the Bicriteria Scheduling Problem of Job-Shop with Blocking. So to evaluate this study, we have chosen to consider in the first time the makespan minimization problem in order to compare the makespan values obtained with those generated by the used software in the company from which the studied problem is issued.

In this section we give the computational results of the makespan minimization problem and the bicriteria minimization problem (the makespan and the maximum of lateness criteria).

a. Makespan minimization problem

We compare the results of the makespan minimization problem with those obtained by tabu search that is proposed in Gröflin & Klinkert [5]. We note that Gröflin & Klinkert's tabu search is the implemented algorithm in the software of the considered company. To use the TS_{CL}, resp. TS_ε, algorithm, we fix $\alpha = 1$ and $\beta = 0$, resp. $\varepsilon = \infty$. Table 3 shows the makespan deviation of our two tabu search algorithms with the algorithm in Gröflin & Klinkert [5], denoted by TS_{GK}. This deviation is calculated as follows:

$$\Delta(TS) = (C_{max}(TS) - C_{max}(TS_{GK}))/C_{max}(TS)$$

With $TS \in \{TS_{CL}, TS_{\varepsilon}\}$. If the $\Delta(TS)$ is a positive value then TS_{GK} results outperforms the results given by the TS algorithm.

Instance	*Size*	C_{max}			$\Delta(TS_{CL})$	$\Delta(TS_{\varepsilon})$
		TS_{GK}	TS_{CL}	TS_{ε}		
La01	10 × 5	832	832	832	0	0
La02	10 × 5	793	852	852	6.924	6.924
La03	10 × 5	747	790	790	5.443	5.443
La04	10 × 5	769	799	833	3.754	7.683
La05	10 × 5	698	743	698	6.056	0
La06	15 × 5	1180	1224	1202	3.594	1.830
La07	15 × 5	1091	1195	1208	8.702	09.68
La08	15 × 5	1125	1202	1205	6.405	6.639
La09	15 × 5	1223	1347	1326	9.205	7.767
La10	15 × 5	1203	1257	1278	4.295	5.868
La11	20 × 5	1584	1666	1673	4.921	5.319
La12	20 × 5	1391	1455	1437	4.398	3.201
La13	20 × 5	1548	1633	1626	5.205	4.797
La14	20 × 5	1620	1666	1694	2.761	4.368
la15	20 × 5	1650	1681	1652	1.844	0.121
la16	10 × 10	1142	1255	1255	9.003	9.003
la17	10 × 10	1026	1078	1087	4.823	5.611
la18	10 × 10	1078	1157	1184	6.828	8.952
la19	10 × 10	1093	1198	1200	8.764	8.916
la20	10 × 10	1154	1238	1215	6.785	5.021
la21	15 × 10	1545	1859	1843	16.890	16.169
la22	15 × 10	1458	1603	1612	9.045	9.553
la23	15 × 10	1611	1834	1796	12.159	10.300
la24	15 × 10	1571	1732	1732	9.295	9.295
la25	15 × 10	1499	1691	1720	11.354	12.849
la26	20 × 10	2162	2329	2394	7.170	9.690
la27	20 × 10	2175	2493	2499	12.755	12.965
la28	20 × 10	2071	2368	2354	12.542	12.022
la29	20 × 10	2124	2283	2298	6.964	7.571
la30	20 × 10	2171	2449	2408	11.351	9.842
la31	30 × 10	3167	3490	3649	9.255	13.209
la32	30 × 10	3418	3935	3908	13.138	12.538
la33	30 × 10	3131	3582	3486	12.590	10.183
la34	30 × 10	3205	3701	3678	13.401	12.860
la35	30 × 10	3311	3720	3665	10.994	9.659
la36	15 × 15	1932	2169	2138	10.926	9.635
la37	15 × 15	2053	2265	2264	9.359	9.319
la38	15 × 15	1875	2115	2127	11.347	11.847
la39	15 × 15	1950	2159	2178	9.680	10.468
la40	15 × 15	1936	2056	2056	5.836	5.836

Table 4. The makespan deviation with TS_{GK}

We can easily see that TS_{GK} outperforms TS_{CL} and TS_{ε}. This is not surprising since our tabu search algorithms are oriented to solve a bicriteria problem. We can see that TS_{ε}outperform TS_{CL} for 17 instances and TS_{CL} outperform TS_{ε} for 17 instances as well and they gives the same values for 6 instances. But at the average the algorithms have a small difference of deviation from the TS_{GK} algorithm, 8.144, resp. 8.073, for TS_{CL} , resp. for TS_{ε}. So the two tabu search algorithm have the same order of effectiveness.

b. Bicriteria minimization problem

To find an approximation, ND^*, of the set ND, the parameters of the two tabu search is fixed as follows:

For the TS_{ε} we first fix the value of ε to ∞ and at each step k, the value of ε is fixed to L_{max}^{k-1}, where L_{max}^{k-1} is the best value of the maximum of lateness obtained at step $k-1$.

For the TS_{CL} , we fix the value of α to 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 and β to $\alpha - 1$.

To evaluate the efficiency of the two tabu search algorithms, we focus on the quality of the approximation computed by the algorithms. Let us refer to $ND_{TS_{\varepsilon}}$, resp. $ND_{TS_{CL}}$, as the approximation calculated by TS_{ε} algorithm, resp. TS_{CL} algorithm. We define ND^* inclu $ND_{TS_{\varepsilon}} \cup ND_{TS_{CL}}$ as the best known approximation, ND contains the non dominated criteria vectors which belongs to $ND_{TS_{\varepsilon}} \cup ND_{TS_{CL}}$. It is clear that the efficiency evaluation of a heuristic is not as simple in the multicriteria case as in the single criterion case. Jaszkiewicz [8] outlines several measures for evaluating the efficiency of multicriteria heuristics. In our work we use four efficiency measures, three of which being relative to the comparison of two fronts and one being relative to the spreading of solutions.

The three measures relative to the comparison between the front $ND_{TS_{\varepsilon}}$, resp. $ND_{TS_{CL}}$L, and the reference front ND^* are defined as follows. Let us denote by $Q_1(A)$ the percentage of potentially non dominated solutions in A which are also in ND^*.

$$Q_1(A) = 100 \times \frac{|A \cap ND^*|}{|ND^*|}$$

The second measure is referred to $Q_2(A)$ as the percentage of solutions in the approximation A which are also in ND^* .

$$Q_2(A) = 100 \times \frac{|A \cap ND^*|}{|A|}$$

The last measure is referred to $Q_2(A)$as the average distance of the front A from the reference front ND^*.

$$Q_3 = \frac{1}{|ND^*|} \sum_{r \in ND^*} \min_{z \in A} \{d(z,r)\}$$

,

Where $d(z,r)$ denotes the Euclidian distance in $\mathcal{R}^2$.

To evaluate the spreading of solutions along the approximation we use spacing measure Q_4.

$$Q_4(A) = \sqrt{\frac{1}{|A|-1}\sum_{z\in A}(\bar{d} - d(z))}$$

Where:

$$d(z) = \min_{z'\in A, z'\neq z}\{\sum_{k=1}^{2}|z_k - z'_k|\}$$

and

$$\bar{d} = \frac{1}{|A|}\sum_{z\in A} d(z)$$

The Table 5, presents the values of the metrics Q_1, Q_2, Q_3 and $Q_4$for each instance. We can see that on the average the tabu search algorithm based on the $\varepsilon - constraint$ approach, TS_ε (column 3), gives a contribution to the Pareto front better than the contribution given by the tabu search algorithm based on the minimization of the linear combination of the criteria, TS_{CL} (column 2). The results of the column 4, $Q_2(ND_{TS_{CL}})$, and the column 5, $Q_2(ND_{TS_\varepsilon})$, show that on the average the TS_ε algorithm outperforms TS_{CL} on the quantity of the solutions. The distance of the front obtained by the TS_{CL} algorithm, column 6, from the best known approximation is greater than that given by the TS_ε algorithm, column 7. We note that the difference is very significant. Columns 8 and 9 illustrate that TS_ε provides non-dominated solutions that have lower average value for the spacing metric. Therefore, the non-dominated solutions obtained by TS_{CL} are more uniformly distributed in comparison with those obtained by TS_ε.

5. Conclusions and future directions

In this paper we have solved the bicriteria Blocking Job Shop Problem $J/d_j, blck/C_{max}, L_{max}$. Because this problem is NP-hard, two versions of a tabu search algorithms have been applied for providing an approximation of non-dominated solutions. One of these algorithms is based on the $\varepsilon - constraint$ approach and the second on a linear combination of criteria. Computational experiments have been conducted, for a single criterion problem and a bicriteria problem, in order to compare the efficiency of the two algorithms and to compare the obtained results with the solutions that are provided by the software which manages the company of Steel Industry. For the problem with a single criterion, minimisation of the makespan, the two algorithms gives almost the same results. For the bicriteria problem, the algorithm TS_ε outperforms the TS_{CL} algorithm but.

In terms of future research, we could consider as extensions to this work, combining the algorithms that we have used with evolutionary algorithms which may yield better solution quality. We can also consider other criteria for this problem.

Instance	Q_1		Q_2		Q_3		Q_4	
	TS_{CL}	TS_{ε}	TS_{CL}	TS_{ε}	TS_{CL}	TS_{ε}	TS_{ε}	TS_{ε}
la01	100	77.777	50	87.5	10.25	1.75	33.589	7.991
la02	33.333	100	50	100	13	0	68.704	0
la03	0	100	0	100	60	0	0.577	0
la04	0	100	0	100	17	0	0	0
la05	16.666	100	25	100	11	0	11.758	10.904
la06	100	100	66.666	66.666	11.666	17	0	0
la07	0	100	0	100	49	0	0	5.196
la08	100	100	100	100	0	0	0	0
la09	33.333	100	25	100	22.75	0	27.135	17.233
la10	66.666	100	25	87.5	24.75	3.5	1.732	15.132
la11	0	100	0	100	53.375	0	0	7.818
la12	0	100	0	100	116	0	17.897	0
la13	0	100	0	100	91.25	0	20.207	27.097
la14	0	100	0	100	67	0	0	0
la15	0	100	0	100	63.75	0	23.813	44.319
la16	60	80	50	66.666	29.166	22.5	51.826	18.253
la17	40	100	40	80	23.4	9.4	8.7	7.505
la18	0	100	0	100	61.625	0	24.62	18.132
la19	0	100	0	100	48	0	32.331	61.199
la20	50	60	25	75	39.5	6	0	14.028
la21	0	100	0	100	226	0	55.852	0
la22	0	100	0	100	83.25	0	16	87.738
la23	0	100	0	100	115.333	0	32.352	48.497
la24	0	100	0	100	163	0	67.926	0
la25	0	100	0	100	86.666	0	0	8.414
la26	50	85.714	14.285	85.714	69.428	10.285	0	32.814
la27	0	100	0	100	127	0	0	43.675
la28	100	100	50	50	42.5	42.5	0	0
la29	100	100	20	100	45	0	0	10.802
la30	33.333	100	20	80	79.2	12	192.257	1.732
la31	0	100	0	100	158	0	0	22.196
la32	0	100	0	100	139.2	0	0	17.038
la33	0	100	0	100	323.5	0	50.519	0
la34	0	100	0	100	268	0	75.491	0
la35	0	100	0	100	265.125	0	41.569	19.3
la36	0	100	0	100	130.75	0	22.818	31.754
la37	0	100	0	100	109	0	3.862	37.883
la38	0	100	0	100	161.5	0	64.893	0
la39	25	100	25	75	126.5	47.25	12.274	97.572
la40	0	100	0	100	114.666	0	13.971	20.51
Average	22.708	97.587	14.648	93.851	91.152	4.304	24.316	18.368

Table 5. Efficiency measures for TS_{CL} and TS_{ε} algorithms

REFERENCES

[1] H. Ait Zai M. Boudhar, Parallel branch and bound and parallel PSO algorithms for the scheduling problem with blocking, Internationnal Journal of Operations Reaserch. (2013).

[2] C.A. Brizuela. Y. Zhao, N. Sonnomiya, No-wait and blockag job-shops: Chalanging problems for GA's, IEEE 0-7803-77-2/ 01 (2001) 2349-2354.

[3] O.Candar, Machine scheduling problems with blocking and no-wait in process, Working paper [April-99], Departement of Industrial Enginnering, Bilkent University, Ankara, Turey, 1999.

[4] H.Gröflin, A.Klinkert, Local search in job shop schediling with synchronization and blocking constraints. Intenal working paper [04-06], Departement of Informatique, University of Fribourg. Switzerland, 2004.

[5] H.Gröflin, A.Klinkert, A new neighborhood and tabu search for the Blocking, Discrete Applied Mathematics, 157, (2009) 3643-3655.

[6] N.G. Hall. C. Striskandarajah, A survey of machine scheduling problems whith blocking and no-wait in process, Operations Research 44 (3) (1996) 510-525.

[7] A. Jaszkiewicz, Evaluation of Multiple Objective Metaheuristics,.In Gandibleux, X., Sevaux, M., Sorensen, K., T'kindt, V. (Eds): Multiple Objective Metaheuristics, Lecture Notes in Economics and Mathematical Systems, {\bf 535} (2004) 1--26.

[8] A. Mascis, D. Pacciarelli, Machine Scheduling via alternative graphs. Research Report, RT-DIA-46-2000, Italy, 2000.

[9] A. Mascis, D. Parcciarelli, JobShop scheduling with blocking and no-wait constarints, European Jornal of Operation Research 143 (2002) 498-517.

[10] Y. Mati, N. Rezg, X. Xie, scheduling problems of Job-Shop with blocking: A taboo search approch, Extended Abstracts, MIC 2001-4th Metaheuristics International Conference, Portugal,, 2001, PP. 643-648.

[11] C. Meloni, D. Pacciarelli, M. Pranzo, A rollout metaheuristic for job-shop scheduling problems, Annals of Operation Research 131 (2004) 215-235.

[12] D. Pacciarelli, Alternative graph formulation for solving complex factory-scheduling problems, International Journal of Production Research 40 (15) (2002) 3641-3653.

[13] G. Vilcot, J. Billaut, A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem, International Journal of Production Research (2011) 1--18.

[14] T'kindt V. et Billaut J-C, .Multicriteria Scheduling: Theory, Models and Algorithms, Springer Berlin, 2005.

[15] T'kindt, V., Bouibede-Hocine, K., Esswein, C, Counting and enumeration complexity with application to multicriteria scheduling, m 4'OR, 3(1), (2005) 1--21.

[16] C. Zeng, J. Tang, C. Yan, Scheduling of no buffer job shop cells with blocking constraints andautomated guided vehicles, Applied Soft Computing24 (2014) 1033–104.