



HAL
open science

No-wait Scheduling of a Two-machine Flow-shop to Minimize the Makespan under Non-Availability Constraints and Different Release Dates

Faten Ben Chihaoui, Imed Kacem, Atidel B. Hadj-Alouane, Najoua Dridi, Nidhal Rezg

► **To cite this version:**

Faten Ben Chihaoui, Imed Kacem, Atidel B. Hadj-Alouane, Najoua Dridi, Nidhal Rezg. No-wait Scheduling of a Two-machine Flow-shop to Minimize the Makespan under Non-Availability Constraints and Different Release Dates. *International Journal of Production Research*, 2011, pp.1. 10.1080/00207543.2010.531775 . hal-00662385

HAL Id: hal-00662385

<https://hal.science/hal-00662385>

Submitted on 24 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



No-wait Scheduling of a Two-machine Flow-shop to Minimize the Makespan under Non-Availability Constraints and Different Release Dates

Journal:	<i>International Journal of Production Research</i>
Manuscript ID:	TPRS-2010-IJPR-0198.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	07-Aug-2010
Complete List of Authors:	Ben Chihaoui, Faten; ENIT Kacem, Imed; Universite Paul Verlaine Metz, Informatique Hadj-Alouane, Atidel; ENIT Dridi, Najoua; ENIT REZG, Nidhal; Universite Paul Verlaine, Metz
Keywords:	FLOW SHOP SCHEDULING, AVAILABILITY, MAKESPAN, MAINTENANCE SCHEDULING, META-HEURISTICS
Keywords (user):	scheduling, flow-shop, no-wait, non-availability, branch-and-bound algorithm

SCHOLARONE™
Manuscripts

No-wait Scheduling of a Two-machine Flow-shop to Minimize the Makespan under Non-Availability Constraints and Different Release Dates

Faten Ben Chihaoui¹, Imed Kacem^{2*}, Atidel B. Hadj-Alouane¹, Najoua Dridi¹, Nidal Rezg³

¹ OASIS, Ecole Nationale d'Ingénieurs de Tunis, Tunisia

² LITA, Université Paul Verlaine Metz, France

³ LGIPM, COSTEAM INRIA, France

Abstract

In this paper, we consider the two-machine no-wait flow-shop scheduling problem, when every machine is subject to one non-availability constraint and jobs have different release dates. The non-availability intervals of the machines overlap and they are known in advance. We aim to find a non-resumable schedule that minimizes the makespan. We propose several lower bounds and upper bounds. These bounding procedures are used in a branch-and-bound algorithm. Computational experiments are carried out on a large set of instances and the obtained results show the effectiveness of our method.

Keywords: scheduling, flow-shop, no-wait, non-availability, branch-and-bound algorithm.

1. Introduction

In this paper, we study the two-machine no-wait flowshop problem under non-availability constraints, when jobs have different release dates. The aim is to minimize the makespan under the three mentioned assumptions. We assume that each machine is unavailable during a fixed interval. The intervals overlap and are known in advance. We assume that if a job cannot be finished before the non-availability period of a machine, the job needs to completely restart once the machine becomes available. This practical assumption is motivated by several real practical situations (preventive maintenance for instance). The second practical assumption considered in this work is related to the no-wait constraint. The main reasons of such a constraint consist in the technological

* kacem@univ-metz.fr (Corresponding author)

1
2
3 structure of the shop itself. In no-wait scheduling, a job has to be continuously processed without
4 idle-time between successive machines. Given the aim of this study, we recall some works related
5 to the considered application and assumptions.
6
7

8 The numerous applications of our first assumption can be found in the paper by [Bagchi et al \(2006\)](#),
9 in which they proposed several no-wait and blocking scheduling models. Moreover, they illustrated
10 some ways in which the used modern manufacturing systems such as robotic cells may be modelled
11 as a *TSP* (Travelling Salesman problem). [Ronconi \(2005\)](#) considered the minimization of the
12 *makespan* criterion for the flowshop problem with blocking. A lower bound exploiting the
13 occurrence of blocking is proposed. A branch-and-bound algorithm incorporating this lower bound
14 is described and its efficiency is evaluated on several problem instances. The makespan
15 minimization problem in a two-machine flowshop under no-wait constraints can be solved to
16 optimality in $O(n \log n)$ time, where n is the number of jobs ([Gilmore and Gomory 1964](#)). However,
17 this problem is strongly *NP-hard* for $m \geq 3$ where m is the number of machines, even if the buffer
18 storage is limited ([Röck, 1980](#)). Moreover, if the no-wait constraint is restricted to a sub-set of jobs
19 then, the problem remains *NP-hard* in the strong sense ([Finke et al 1997](#)). The m -machine no-wait
20 flowshop scheduling problem with the aim of minimizing the *makespan* and the total completion
21 time was studied in [Allahverdi and Aldowaisan \(2002\)](#). A dominance rule and heuristics were
22 proposed and used in a branch-and-bound algorithm. For more details on no-wait and blocking
23 scheduling problems, the reader is invited to consult the state-of-the-art paper by [Hall and](#)
24 [Sriskandarajah \(1996\)](#).
25
26
27
28
29
30
31
32
33
34
35
36
37
38

39 The second assumption, that is, the non-availability constraint is one of the new modern concepts in
40 the scheduling theory. Abundant literature exists on the related problems. Lee was the pioneer of
41 this research field in scheduling theory ([Lee 1997](#)). Many papers were published during the last two
42 decades and they involved various works on the flow-shop configuration. The *makespan*
43 minimization on the two-machine flow-shop problem under non-availability constraints was proven
44 to be *NP-hard*, even with a single non-availability period ([Espinouse et al 1999](#)). Aggoune and
45 Portmann proposed a heuristic method for the general flow-shop problem under non-availability
46 constraints on a subset of the machines ([Aggoune and Portmann 2006](#)). Lee studied the two-
47 machine flowshop problem under the assumption that the non-availability time is known in advance
48 ([Lee 1999](#)). Moreover, he considered the *semiresumable*, the *resumable* and the *nonresumable*
49 cases. Lee also conducted a complexity analysis and elaborated a pseudo-polynomial dynamic
50 programming algorithm to solve the problem to optimality. Heuristic algorithms were proposed and
51 evaluated in the worst-case ([Lee 1999](#)). Allaoui et al. considered the same problem with a single
52
53
54
55
56
57
58
59
60

1
2
3 non-availability interval on the first machine under the *nonresumable* scenario (Allaoui et al 2006).
4 They improved the dynamic programming model proposed by Lee (1997). This method allowed
5 them to reduce the computational effort. Some conditions, where *Johnson's* rule gives the optimal
6 solution were specified. They proved that the worst-case performance bound of *Johnson's* rule is 2
7 (Allaoui et al 2006). For more details on the non-availability constraints in scheduling problems, we
8 refer to the survey-papers by Schmidt (2000) and Ma et al. (2010).
9

10
11 Finally, our third assumption states that jobs have different release dates. Several literature works
12 considered this case. A sample of them is summarized as follows. First, we note that the classical
13 version consisting in the *makespan* minimization on two-machine flow-shop subject to jobs release
14 dates ($F2/r_j/C_{\max}$) is *NP-hard* in the strong sense (Lenstra et al 1977). Thus, heuristic approaches
15 were widely studied. In (Potts 1985) four heuristics were proposed to solve the same problem. For
16 three of them, the worst-case performance ratio is of 2. Each one of the heuristics can be
17 implemented in $O(n \log n)$ time. The fourth one is based on the iterative use of the third heuristic
18 and it has a worst-case performance ratio of $5/3$ and a time complexity of $O(n^3 \log n)$. In
19 (Kashyrskikh 2001), by modifying Potts algorithm (Potts 1985), the authors reduced the worst-case
20 performance ratio to $3/2$ however, the time complexity remains in $O(n^3 \log n)$. A polynomial time
21 approximation scheme (PTAS) was proposed in (Hall 1995). In (Kovalyov and Werner 1997) a
22 polynomial approximation procedure was elaborated for the $F2/r_j/C_{\max}$ problem. Such a
23 procedure was based on a dynamic programming approach using modified release dates and
24 processing times. Compared to the one proposed in (Hall 1994) a better time complexity was
25 obtained for large values of n . Branch-and-bound methods were exploited in (Cheng et al 2001) for
26 solving the $F3/r_j/C_{\max}$ problem by incorporating a lower bound determined for some particular
27 cases. Two dominance rules were used to construct an initial schedule. These rules reduced the
28 search space by decomposing the problem into a finite set of sub-problems.
29

30
31 To conclude, according to this literature review we can mention that in all studied papers, at most
32 two of the three assumptions were considered. A sample of these works includes Cheng and Liu
33 2003, Espinouse et al 1999, Espinouse et al 2001, Kubzin and Strusevich 2004 and Wang and
34 Cheng 2001, where no-wait and non-availability constraints were jointly considered. Finally, the
35 non-availability constraint and the different job release dates were considered at the same time by
36 França et al 2006, where genetic algorithms were elaborated. Moreover, the same problem was
37 solved in Bianco et al 1999 by a mathematical programming method and two heuristics. However,
38 according to the best of our knowledge there is no previous work related to the studied problem,
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 that is, the minimization of the *makespan* in two-machine no-wait flow-shop under non-availability
4 constraints and different job release dates assumption. For this reason, this paper is a first successful
5 attempt to design a branch-and-bound algorithm with an interesting performance.
6
7

8
9 The reminder of the paper is organized as follows. Section 2 gives a precise formulation. In Section
10 3, we describe the proposed branch-and-bound method. Computational results are given and
11 discussed in Section 4. Finally, we conclude the paper by some remarks and perspectives in the last
12 section.
13
14

15 16 17 18 **2. Problem formulation**

19
20 The problem can be stated as follows. We have a set $J = \{1, \dots, n\}$ of n jobs to be performed on two
21 machines M_1 and M_2 . Every job has to be processed first on M_1 then on M_2 . For every job, the
22 second operation has to start immediately at the end of the first operation (no idle-time is allowed
23 between two consecutive operations of a given job). Every machine M_i ($i=1,2$) is unavailable during
24 the interval (s_i, t_i) . Due to practical requirements we assume that the two intervals (s_1, t_1) and
25 (s_2, t_2) overlap and that $s_1 \leq s_2$ and $t_1 \leq t_2$. Jobs have to be processed under the *nonresumable*
26 scenario (a job has to completely restart once interrupted by a non-availability interval). Every job j
27 ($j=1, \dots, n$) has a positive release date r_j known in advance. Its first (respectively the second)
28 operation on the first (respectively the second) machine has a positive processing time of a_j
29 (respectively of b_j). The objective is to find a feasible schedule with the aim of minimizing the
30 *makespan* (i.e., the completion time of the last operation performed on the second machine). From
31 Section 1, this problem is *NP-hard* in the strong sense since it is a generalization of other problems
32 of this type (for instance, see the $F2/r_j/C_{\max}$ problem in [Lenstra et al 1977](#)).
33
34
35
36
37
38
39
40
41
42
43
44
45
46

47 48 **3. Branch-and-bound algorithm**

49
50 Motivated by the practical advantages of the branch-and-bound approach, we elaborated an
51 algorithm of this type to solve our considered problem. Such an approach aims to find an optimal
52 solution (or an enhanced one) by reducing the search space based on different tools (lower bounds,
53 dominance rules, upper bounds...). When it is not possible to obtain the optimal solution such an
54 approach allows us to improve the result of other heuristic methods that can be used as an initial
55 upper bound. In this section we give the description of our branch-and-bound algorithm. It will be
56 noted B&B in the remainder of the paper.
57
58
59
60

3.1. Branching scheme and search strategy

The B&B starts by computing an initial solution which provides the first upper bound. Every node represents a partial schedule. The branching scheme consists in scheduling a new job after a partial schedule. The search space is explored by using the *depth first strategy*. Before creating a new node, a lower bound is computed. If the value of such a lower bound is greater than the value of the upper bound, then this node is removed. Moreover, before any branching procedure, we increase the release date of every jobs $j \in J$ satisfying one of the two following conditions: $(r_j + a_j > s_1)$ or $(r_j + a_j + b_j > s_2)$. Indeed, in the two cases the first operation of job j cannot be performed before s_1 . Thus, we can increase the release date as follows: $r_j = \max\{r_j, t_1, t_2 - a_j\}$. Consequently, all the partial sequences beginning with these jobs verifying one of the conditions are eliminated.

3.2. Upper bounds

The quality of the used upper bounds is very important to enhance the effectiveness of any branch-and-bound algorithm. For this reason, we investigated different ways to determine the initial upper bound. Three methods are used:

- A greedy search (denoted as *GS*): it consists in applying the B&B for only n nodes by using the *depth best first strategy*.
- Two Heuristics (*H1* and *H2*) developed by [Ben Chihaoui et al 2009](#).
- A genetic algorithm (denoted as *GA*).

For self-consistency, we recall the principle of of Heuristics (*H1* and *H2*). These two heuristics are based on the algorithms in [Gilmore-Gomory 1964](#) and [Cheng and Liu 2003](#). Before describing these heuristics, we present additional notations.

- I_k : Set of jobs arriving at the time r_k ;
- $O_k (D_k)$: Set of jobs of the sub-problem k in *H1* (*H2*);
- $\sigma_k^1 (\sigma_k^2)$: The sequence found by scheduling the jobs of $O_k (D_k)$ in *H1* (*H2*);
- *CL*: Algorithm 3 of [Cheng and Liu 2003](#).

Description of *CL*

In [Cheng and Liu 2003](#), the authors study the two-machine no-wait flowshop problem in which each machine may have an unavailable interval. A $3/2$ -approximation algorithm is developed for the problem resolution when the unavailable intervals on the two machines overlap. It consists in the following steps:

1. Try to find a good schedule in which the availability constraint is inactive (a fictitious job is added).
2. Relax the availability constraint, and then move some jobs from the beginning to the end or vice versa to meet the unavailability constraint.
3. Optimally schedule some critical job and its adjacent jobs and schedule the other jobs according to Gilmore and Gomory's algorithm.

Description of *H1*

The first heuristic, *H1*, is based on the decomposition of the problem into many sub-problems. The first sub-problem deals with the set I_0 of jobs that are available at the date r_0 . The (CL) algorithm is used for its resolution. From the obtained solution, we keep the sequence σ_0^1 of jobs starting their execution on M_1 , before the date r_1 . The next sub-problem concerns jobs of the set O_1 , which includes jobs that are in I_1 but not in σ_0^1 . The solution obtained for this sub-problem is concatenated to σ_0^1 to obtain a new partial sequence. Then, we consider as many sub-problems as the number of release dates r_k .

Description of *H2*

The second heuristic, *H2*, is based on a modification of *H1* to use the concept of a reactive scheduling. Indeed, a job j can arrive to the shop after a job i . However, the execution of j before i may give a better solution.

In *H2*, sub-problems are defined as follows. First, jobs arriving at r_i are temporarily scheduled to obtain their finish date, referred to as f_i . Then the sub-problem consists in rescheduling all jobs arriving at r_i along with jobs arriving before f_i .

The first sub-problems concern the jobs in set D_0 . We apply *CL* to schedule set I_0 . The makespan obtained is considered as f_0 . Set D_0 includes jobs of set I_0 and those which release dates verify the condition: $r_0 \leq r_j \leq f_0$. Let $r_k = \max_j \{r_j\}$. We keep from the solution obtained by the application of *CL* on D_0 , the sequence of jobs beginning their execution on M_1 before the date r_{k+1} . Let σ_k^2 be the obtained sequence.

The second sub-problem concerns jobs of set D_1 .

CL is applied to schedule the set I_{k+1} to which we include jobs of the set D_0 not kept in the sequence σ_0^2 . The makespan of the resulted solution is f_1 . Beside the jobs of the set I_{k+1} , D_1 includes

1
2
3 jobs arriving after r_{k+1} in the time period of f_1 . The solution obtained for this sub-problem is
4 concatenated to σ_0^2 to obtain a new partial sequence (Ben Chihaoui et al 2009).
5
6

7 **Description of the genetic algorithm:**

8
9 We use a classical genetic algorithm in which the solutions or chromosomes are represented as a
10 permutation of J indicating for every job its position in the schedule (Fig.1 (a)). The mutation
11 operator consists in choosing two random positions of the mutated genes and in swapping the two
12 corresponding jobs (Fig.1 (b)). The crossover operator consists in selecting a random position and
13 in exchanging the genetic information between two parents to construct two offspring according to
14 the selected position (Fig.1 (c)). The construction consists in copying each subset of jobs in the
15 same order as they appear in the corresponding parent so that the offspring will be feasible. The
16 number of generations is fixed to 100 and the initial population is fixed to 100 chromosomes
17 randomly generated by iteratively mutating the FIFO sequence (the FIFO sequence consists in
18 scheduling jobs in non-decreasing order of their release dates).
19
20
21
22
23
24
25
26

27 The initial upper bound used in the B&B is the minimum of the obtained values with these three
28 methods.
29
30

31
32 **//Please insert Figure 1 about here**
33

34 **3.3. Lower bounds**

35
36 It is well known that the quality of the lower bound is one of the most critical elements of any
37 branch-and-bound algorithm. In most cases, computing a lower bound consists in relaxing some
38 constraints (in different ways) and in solving a new easier problem. In this paper, we consider two
39 main relaxations to derive lower bounds. The first one replaces the two non-availability intervals by
40 two fictitious jobs. The second relaxation reduces the flow-shop configuration to a *single-machine*
41 *problem*. Before presenting the proposed lower bounds we need to define the data of the fictitious
42 jobs. Let $f1$ be the first fictitious job and $f2$ be the second one. Their processing times are defined as
43 follows: $a_{f1} = s_2 - s_1, b_{f1} = t_1 - s_2, r_{f1} = s_1, a_{f2} = t_1 - s_2, b_{f2} = t_2 - t_1, r_{f2} = s_2$. Moreover, we define
44 a global set of jobs J_f as follows $J_f = J \cup \{f1, f2\}$.
45
46
47
48
49
50
51
52
53
54

55 We derived five lower bounds denoted: $LB1, LB2, LB3, LB4$ and $LB5$. Before computing each lower
56 bound the release date of every job $j \in J$, not yet scheduled, is updated according to whether or not
57 the completion time of the scheduled jobs is greater than the starting time of the non-availability
58 interval on machine M_j . The construction of every lower bound is described as follows.
59
60

3.3.1. LB1

In this lower bound, we transform the problem into an instance of a single-machine problem of the type $1/r_j/C_{\max}$. More precisely, the first machine is removed and we only consider the second machine. In the new obtained instance, we first generate the two fictitious jobs. Then, with every $j \in J_f$ it is associated a new job to be performed on the single-machine such that the release date and the processing time are respectively equal to $r_j + a_j$ and b_j . The resulting problem is optimally solved by the *FIFO* (*First In First Out*) rule.

3.3.2. LB2

In this lower bound, we transform the problem into an instance of the type $1, h_1/r_j, q_j/C_{\max}$. More precisely, the second machine is removed and the second operation of every job $j \in J_f$ is replaced by a tail (or a delivery time) equal to b_j . In the new obtained instance, we first generate the two fictitious jobs. Then, with every $j \in J_f$ it is associated a new job to be performed on the single-machine such that the release date, the processing time and the tail are respectively equal to r_j , a_j and b_j . Moreover, the new fictitious jobs must start exactly at their respective release dates. The resulting problem is NP-Hard, but the lower bound is obtained by solving the preemptive version to optimality using Jackson's rule (see [Carlier et al 2010](#)).

3.3.3. LB3

The principle of this bound is similar to the previous one. We follow the same relaxations as in *LB2* to transform the problem into an instance of the type $1/r_j, q_j/C_{\max}$. The only difference is the fact that the new jobs associated with the fictitious jobs are not constrained to start exactly at their respective release dates. The resulting problem is solved by the branch-and-bound proposed in [Carlier \(1982\)](#).

3.3.4. LB4

The principle of this bound is based on the relaxations used in *LB1*. We follow the same relaxations. The only difference is that the new jobs associated with the fictitious jobs are constrained to be performed exactly in the interval (s_2, t_2) . Thus, they can be considered as a non-availability interval. Moreover, the *resumable* scenario is considered. The resulting problem belongs to the $1, h_1/r_j, rs/C_{\max}$ family and it is solved to optimality by the preemptive *FIFO* rule. Note that an analytical comparison can easily show that *LB4* gives the same result as *LB1*. Hence, this bound will be omitted. However, we felt it is very important to report it to show that this idea is not more

productive than *LB1*.

3.3.5. *LB5*

The fifth lower bound is obtained by transforming the problem into an instance of the $1, h_j/r_j, nrs/C_{max}$ type. In such a case, the first machine is removed and we only consider the second machine. In the new obtained instance, we do not generate any fictitious job. Then, with every $j \in J$ it is associated a new job to be performed on the single-machine such that the release date and the processing time are respectively equal to $r_j + a_j$ and b_j . Moreover, the single-machine is considered as non-available during the interval (s_2, t_2) . The resulting problem is solved (under the non-resumable scenario) by the dynamic programming algorithm proposed in [Kacem and Haouari \(2009\)](#). Clearly, this bound outperforms *LB1*. However, it needs more computation time.

4. Numerical experiments

In this section, we describe the numerical experiments carried out in order to evaluate our algorithms. The B&B was implemented in the C language and tested on an Intel Pentium IV 3 GHz processor and 512 M RAM, in the WINDOWS XP environment. The instances were randomly generated. For the experiments, we generated ten instances for every combination of parameters as follows. The number of jobs n was chosen in $\{5,10,15,20\}$. The release dates were uniformly distributed in the interval $[1,100 * R]$, where $R \in \{1,2,n,2n\}$. Processing times a_j and b_j were uniformly distributed in the interval $[1,100]$. The horizon T of the schedule is defined as the sum of all the processing times $T = \sum_{j \in J} a_j + \sum_{j \in J} b_j$. Then, the non-availability periods were fixed at the beginning ($s_1 = 0.25 \times T$) or in the middle ($s_1 = 0.5 \times T$) of the defined horizon, where $t_1 = s_1 + T/n$, $s_2 = s_1 + 0.25 \times T/n$ and $t_2 = s_2 + 1.25 \times T/n$. Based on the values of parameters R and s_1 , we define eight groups of instances (1, 2, 3, 4, 5, 6, 7 and 8) as it is shown in Table 1.

//Please insert Table 1 about here

4.1. Lower and upper bounds evaluation

To evaluate the performance of the lower bounds presented in this paper, we compute the average value obtained from ten instances for every combination of parameters. In order to check the effectiveness of *GS* and *GA* compared with *H1* and *H2*, we compute the average value obtained from the ten instances. The different results are reported in Tables 2-6.

1
2
3 For every group the best lower bound (LB) is written in boldface. For the upper bound (UB) the
4 Gap in percentage is calculated with respect to the best lower bound.
5
6
7

8 **//Please insert Table 2 about here**

9
10 **//Please insert Table 3 about here**
11 **//Please insert Table 4 about here**

12
13 **//Please insert Table 5 about here**

14
15 **//Please insert Table 6 about here**
16
17
18
19
20

21 From Tables 2-6, we can make the following remarks:

- 22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
- Except for the case where $n=5$, one can remark that either GS or GA gives the best results in most cases.
 - For the lower ranges of release dates, $H2$ gives better results than $H1$. This is due to the reactive aspect of $H2$. However, for a larger range of release dates, $H1$ gives better results than $H2$.
 - For the lower range of the release dates, $LB5$ is the best lower bound. However, we can remark that $LB1$ gives almost the same results as $LB5$. Hence, $LB1$ represents an interesting bound with a good quality and a short computation time.
 - For a larger range of the release dates values, the different lower bounds give similar results.
 - When the non-availability is in the middle of the horizon, the bounds values are generally better (the gap is smaller), and we notice that $LB5$ and $LB1$ generally remain the best in the two cases.
 - The computation time of the different lower bounds is close to zero. However, the computation time of $LB5$ for $n=20$ is about 120 s.
 - The gap between the lower and upper bounds values is relatively small. This allows us to elaborate an efficient B&B.

49 Given the above remarks, and taking into account the time complexity of every lower bound, we
50 have decided to mainly use $LB1$ in the B&B and to employ $LB3$ and $LB5$ in the Greedy Search.
51
52
53
54
55

56 4.2. B&B performance

57 In order to evaluate the B&B performance, computational experiments are conducted on the same
58 instances used to evaluate the LB and UB. The computation time is limited to 7200s. For the node
59 evaluation we use first $LB1$, if the node is not eliminated, we use $LB5$. The B&B performances are
60

presented in Tables 7-10, where the following parameters are reported:

- *Copt*: average of the optimal makespan values
- *UB*: average of the best upper bound values
- *CPU*: average of the B&B computation time values in seconds
- *N_d*: average number of eliminated nodes in the *preliminary elimination*
- *N_{ex}*: average number of explored nodes
- *N_{el}*: average number of eliminated nodes
- *Gap (%)*: average value of the gap between *UB* and *Copt*

//Please insert Table 7 about here

//Please insert Table 8 about here

//Please insert Table 9 about here

//Please insert Table 10 about here

From Tables 7-10, we can make the following remarks:

- The average value of the gap is about 3%. This value clearly shows the good performance of the B&B and the tightness of the upper and lower bounds.
- The average value of the gap decreases when n increases.
- Computation time is larger for higher values of n .
- Computation time and average number of the explored nodes are relatively high when the release dates range is independent of n (r_j in $[1,100]$ and in $[1,200]$) and the non-availability interval is in the middle of the horizon. Consequently, the problem is more difficult for lower range of release dates and when the non-availability is in the middle of the horizon. Indeed, if the non-availability is in the beginning of the horizon, it is skipped early in the horizon (after sequencing some jobs), then the two machines become available.
- The B&B can solve instances with up to 20 jobs, except for instances of groups 1 to 4 (r_j in $[1,100]$ and in $[1,200]$). Indeed, the low range of release dates in these groups increases the search space. Consequently, the search tree is bigger and the problem is difficult to solve.
- The preliminary elimination is more efficient when n is large.
- The lower bounds and, in particular, *LB1* and *LB5* allow for an extensive elimination of nodes.

Thus, they enhance the B&B effectiveness.

For larger values of n (30, 40 and 50) we tested the performance of the Greedy Search heuristic by reporting the following parameters in Table 8:

- LB : average of the best lower bound values
- t_{GS} : average of the GS computation times
- $G_H(\%)$: average value of the gap between H (H can be $H1$, $H2$, GA and GS) and LB .

The average values of the computation times required by $H1$, $H2$ and GA are close to zero second.

The obtained performances are reported in Table 11.

//Please insert Table 11 about here

From Table 11, we can make the following conclusions:

- The maximum value of the gap obtained for the different methods is in average of 11% and the minimum is of 1%. Thus, the methods that we developed as well as the elaborated lower bounds yield good results.
- The value of G_{GS} is of 1% whereas $G_{GA} \approx 8\%$, $G_{H1} \approx 4\%$ and $G_{H2} \approx 11\%$. Hence, the Greedy Search clearly outperforms all the other algorithms. The GS has a larger computation time (about 0,4s in average) than the other algorithms. Nevertheless, such a value is very small in practical situations.
- In conclusion, our GS algorithm seems to be a very interesting method in order to obtain a near-optimal solution (gap of 1%) in a short computation time (in less than 1s) despite the strong NP -Hardness of the studied problem.

5. Conclusion

This paper studied the two-machine no-wait flow-shop scheduling problem, when every machine is subject to one non-availability constraint and jobs have different release dates. The aim is to minimize the makespan. Several lower and upper bounds are proposed and incorporated in a branch-and-bound algorithm. Numerical experiments were carried out on a large set of instances. The obtained results showed that we can find the optimal solution for problems with up to 20 jobs within a reasonable amount of computation time. Moreover, the branch-and-bound algorithm can be converted into a greedy search heuristic, GS , that has a good performance. Such a heuristic is able to give a near-optimal solution (gap of 1%) in a short computation time (in less than 1s) despite the

1
2
3 strong *NP-Hardness* of the studied problem.

4
5 In a future work, we aim to develop new lower bounds with possibly other types of relaxations, as
6 well as a branch-and-cut method to optimally solve instances of larger sizes. Moreover, the study of
7 new metaheuristic algorithms and representations seems to be interesting in order to improve the
8 B&B performances.
9

10 11 12 **Acknowledgements:**

13
14 We thank referees for their constructive comments which significantly improved the paper.
15
16

17 18 19 **References**

- 20
21 Aggoune, R., Portmann, M.C (2006). Flow shop scheduling problem with limited machine
22 availability: A heuristic approach. *International Journal of Production Economics* 99, 4-16.
23
24 Allaoui, A., Artiba, A., Elmaghraby, S.E., Riane, F (2006). Scheduling of a two-machine flowshop
25 with availability constraints on the first machine. *International Journal of Production*
26 *Economics* 99, 16-27.
27
28 Allahverdi, A., Aldowaisan, T. (2002). No-Wait Flowshops with Bicriteria of Makespan and Total
29 Completion Time. *Journal of the Operational Research Society* 53:9, 1004-1015.
30
31 Bagchi, T.P., Gupta, J.N.D., Sriskandarajah, C (2006). A review of TSP based approaches for
32 flowshop scheduling. *European Journal of Operational Research* 169, 816-854.
33
34 Bianco, L., Dell'Olmo, P., Giordani, S (1999). Flow shop no-wait scheduling with sequence
35 dependent setup times and release dates. *INFOR* 37:1, 3-19.
36
37 Carlier, J (1982). The one-machine sequencing problem. *European Journal of Operational*
38 *Research* 11, 42-47.
39
40 Carlier, J., Hermès, F., Moukrim, A., and Ghédira, K (2010). Exact resolution of the one-machine
41 sequencing problem with no machine idle time. *Computers & Industrial Engineering*
42 DOI:10.1016/j.cie.2010.03.007.
43
44 Cheng, J., Steiner, G., Stephenson, P (2001). A computational study with a new algorithm for three-
45 machine permutation flow shop problem with release times. *European Journal of*
46 *Operational Research* 130, 559–575.
47
48 Cheng, T.C. E., Liu, Z (2003). $3/2$ approximation for two-machine no-wait flowshop scheduling
49 with availability constraints. *Information Processing Letters* 88, 161-165.
50
51 Ben Chihaoui, F., Dridi, N., Ben Hadj-Alouane, A (2009). Two-machine No-wait Flowshop
52 Scheduling with Availability Constraints and Release Dates. *Proceedings of the 2009*
53 *International Conference on Computers & Industrial Engineering, 6-8 July 2009, Troyes,*
54
55
56
57
58
59
60

1
2
3 France.

- 4
5 Espinouse, M.L., Formanowicz, P., Penz, B (1999). Minimizing the makespan in the two-machine
6 no-wait flow-shop with limited machine availability. *Computers & Industrial Engineering*
7 37, 497-500.
8
9
10 Espinouse, M.L., Formanowicz, P., Penz, B (2001). Complexity results and approximation for the
11 two-machine no-wait flow-shop with limited machine availability. *Journal of the*
12 *Operational Research Society* 52, 116-121.
13
14
15 Finke, G., Espinouse, M.L., Jiang, H (1997). Flowshops and extensions. In: *Proceedings:*
16 *Conference on Management and Control of Production and Logistics (MCPL'97)*,
17 Campinas, Brazil, 275-280.
18
19
20 França, P. M., Tin, G., Buriol, L.S (2006). Genetic algorithms for the no-wait flowshop sequencing
21 problem with time restrictions. *International Journal of Production Research* 44, 939-957.
22
23
24 Gilmore, P.C., Gomory, R.E (1964). Sequencing a one-state variable machine: a solvable case of the
25 travelling salesman problem. *Operations Research* 12, 655-679.
26
27
28 Hall, L.A (1994). A polynomial approximation scheme for a constrained flow-shop scheduling
29 problem. *Mathematics of Operations Research* 19, 68-85.
30
31
32 Hall, L.A (1995). Approximability of flowshop scheduling. *Proceedings of the 36th Annual*
33 *Symposium on Foundations of Computer Science, IEEE Comput. Soc. press*, Los Alamitos,
34 82-91.
35
36
37 Hall, N.G., Sriskandarajah, C (1996). A survey of machine scheduling problems with blocking and
38 no-wait in process. *Operations Research* 44:3, 510-525.
39
40
41 Kacem, I., M. Haouari, M (2009). Approximation algorithms for single machine scheduling with
42 one unavailability period. *4OR: A Quarterly Journal of Operations Research* 7:1, 79-92.
43
44 Kashyrskikh, K.N., Potts, C.N., Sevastianov, S.V (2001). A (3/2)-approximation algorithm for two-
45 machine flwo-shop sequencing subject to release dates. *Discrete Applied Mathematics* 114,
46 255-271.
47
48
49 Kubzin, M.A., Strusevich, V.A (2004). Approximation Algorithms for Two-Machine Flow Shop No-
50 Wait Scheduling with a Non-Availability Interval. *Naval Research Logistics* 51:4, 613 – 631.
51
52
53 Kovalyov, M.Y., Werner, F (1997). A polynomial approximation scheme for problem $F2/r_j/C_{max}$.
54 *Operations Research Letters* 20, 75-79.
55
56
57 Lee, C. Y (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with
58 an availability constraint. *Operations Research Letters* 20, 129-139.
59
60 Lee, C. Y (1999). Two-machine flowshop scheduling with availability constraints. *European*
Journal of Operational Research 114, 420-429.

- 1
2
3 Lenstra, J.K., Rinnooy Kan, A.H.G., P., Brucker (1977). Complexity of machine scheduling
4 problems”, in: *Studies in Integer Programming*, eds. P.L. Hammer, E.L. Johnson, B.H.
5 Korter and G.L. Nemhauser, *Ann. Discrete Math.*, North-Holland, Amsterdam 1, 343–362.
6
7
8
9 Ma, Y., Chu, C., Zuo, C (2010). A survey of scheduling with deterministic machine availability
10 constraints. *Computers & Industrial Engineering* 58:2, 199-211.
11
12 Potts, C.N (1985). Analysis of heuristics for two-machine flow-shop sequencing subject to release
13 dates. *Mathematics of Operations Research* 10, 576–584.
14
15
16 Ronconi, D. P (2005). A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop
17 with Blocking. *Annals of Operations Research* 138, 53–65.
18
19 Röck, H (1980). The three machine-machine no-wait flow shop problem is NP-complete. *Journal of*
20 *ACM* 31, 336-345.
21
22
23 Schmidt, G (2000). Scheduling with limited machine availability. *European Journal of Operational*
24 *Research* 121, 1-15.
25
26 Wang, G., Cheng, T.C.E (2001). Heuristics for two-machine no-wait flowshop scheduling with
27 availability constraints. *Information Processing Letters* 80, 305-309.
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

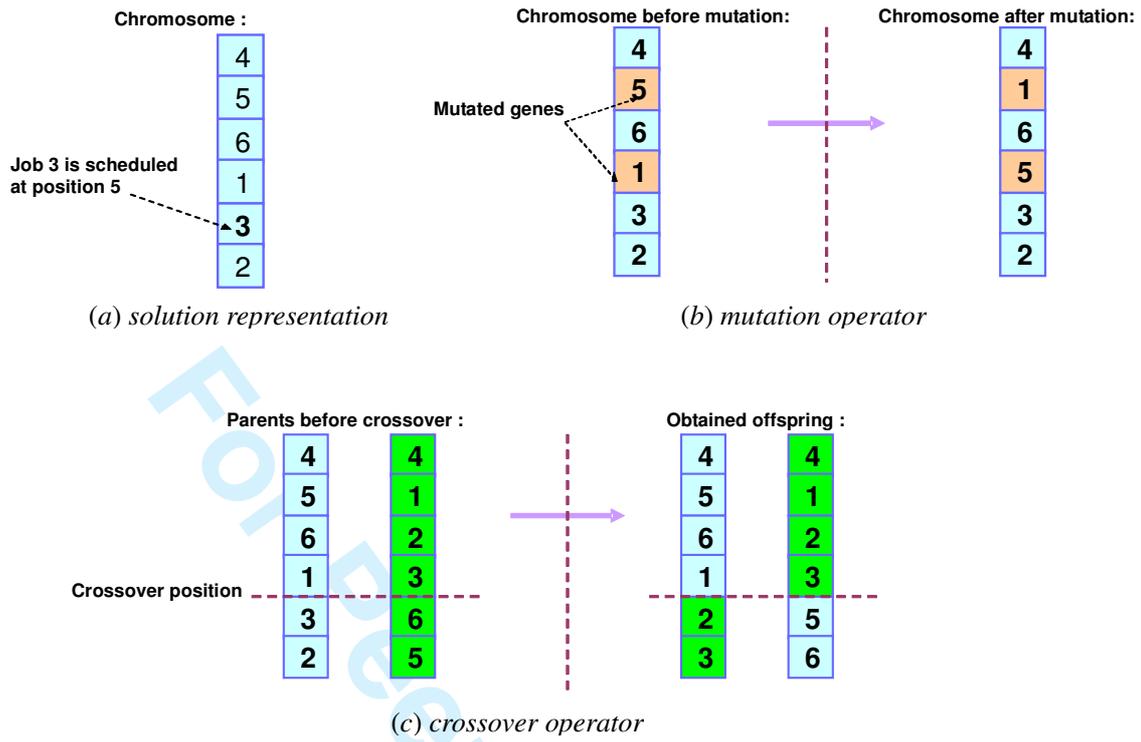


Fig. 1. Representation and operators

Table1. Group identification

Parameter R	$s_1 = 0.25 \times T$	$s_1 = 0.5 \times T$
1	1	2
2	3	4
n	5	6
$2n$	7	8

Table 2. Results of lower and upper bounds for n=5

Groups	Average values of LB				Gap of UB (in %)			
	LB1	LB2	LB3	LB5	H1	H2	GA	GS
1	430	399	411	451	22	11	17	22
2	464	415	440	473	18	2	12	16
3	511	420	432	531	3	2	5	6
4	448	395	414	478	15	7	13	18
5	563	565	567	563	8	18	17	11
6	570	564	586	582	12	16	12	8
7	792	792	792	792	0	2	9	0
8	868	868	868	868	4	8	10	1

Table 3. Results of lower and upper bounds for n=10

Groups	Average values of LB				Gap of UB (in %)			
	LB1	LB2	LB3	LB5	H1	H2	GA	GS
1	693	651	651	693	27	15	12	30
2	632	535	654	633	30	18	15	28
3	708	634	626	713	15	2	9	20
4	758	548	668	761	15	12	7	18
5	1039	1036	1036	1039	10	24	5	3
6	1060	1073	1072	1061	5	9	4	3
7	1889	1884	1884	1889	2	4	3	0
8	1963	1939	1939	1963	2	3	3	1

Table 4. Results of lower and upper bounds for n=15

Groups	Average values of LB				Gap of UB (in %)			
	LB1	LB2	LB3	LB5	H1	H2	GA	GS
1	1002	895	904	1002	28	20	11	21
2	962	815	938	962	23	6	9	14
3	930	906	906	930	33	14	14	27
4	925	665	851	927	25	16	7	22
5	1564	1563	1563	1564	3	9	5	2
6	1542	1507	1507	1542	8	14	4	4
7	2962	2948	2948	2962	6	10	1	1
8	2826	2810	2817	2826	7	15	3	0

Table 5. Results of lower and upper bounds for $n=20$

Groups	Average values of LB				Gap of UB (in %)			
	LB1	LB2	LB3	LB5	H1	H2	GA	GS
1	1126	1140	1128	1138	20	20	13	32
2	1199	1195	1195	1199	8	16	13	30
3	1221	1109	1099	1231	13	18	8	27
4	1244	1154	1154	1250	12	19	12	22
5	2034	2034	2029	2034	0	6	3	0
6	2060	2068	2068	2060	2	6	5	1
7	4022	4022	4022	4022	2	4	2	0
8	4016	4015	4015	4016	1	2	2	1

Table 6. Summary of computational times of LB and UB (in seconds)

n	LB1	LB2	LB3	LB5	H1	H2	GA	GS
5	0	0	0	0	0	0	0,15	0,02
10	0	0	0	0	0	0	0,15	0,02
15	0	0	0	2	0	0	0,31	0,05
20	0	0	0	120	0	0	0,31	0,05

Table 7. B&B performance for $n=5$

Groups	UB	Copt	CPU	N_d	N_{el}	N_{ex}	Gap
1	453	443	0	3	12	33	2
2	484	467	0	0	23	45	3
3	520	489	0	6	7	25	6
4	594	562	0	1	19	33	6
5	592	553	0	7	10	15	7
6	644	628	0	5	10	19	2
7	906	902	0	2	5	7	0
8	1009	1009	0	2	4	3	0

Table 8. B&B performance for $n=10$

Groups	UB	Copt	CPU	N_d	N_{el}	N_{ex}	Gap
1	748	708	0	4	146217	134475	6
2	749	713	1	0	287549	143090	5
3	766	736	0	6	131878	147749	4
4	780	745	0	2	95430	42601	5
5	1094	1084	0	10	2220	1760	1
6	1052	1034	0	17	11246	9482	2
7	1899	1899	0	3	2333	1571	0
8	1968	1957	0	5	4424	2518	1

Table 9. B&B performance for n=15

Groups	UB	Copt	CPU	N_d	N_{el}	N_{ex}	Gap
1	1045	1010	1154	3	247862574	316505246	3
2	1059	1019	3483	0	1623109776	495159902	4
3	1065	1019	952	9	170304340	254320785	5
4	1024	974	2247	5	1039966545	404465743	5
5	1616	1604	320	14	4941954	3579316	1
6	1611	1550	3600	24	24620459	16281991	4
7	2937	2935	180	6	4771824	2984638	0
8	2925	2920	1600	17	91077335	130196286	0

Table 10. B&B performance for n=20

Groups	UB	Copt	CPU	N_d	N_{el}	N_{ex}	Gap
1,2,3,4	The B&B failed to improve the upper bound before the time limit						
5	2089	2065	1356	18	10440088	16649816	1
6	2225	2145	7200	57	218131733	192697602	4
7	3899	3887	5360	12	138114617	58012023	0
8	3810	3809	6200	5	152033322	232010045	0

Table 11. Performance of algorithms H1, H2, GA and GS for n=30 to 50

n	LB	GA	GS	H1	H2	G_H1	G_H2	G_GA	G_GS	t_GS
30	3087	3245	3104	3195	3403	3%	10%	5%	1%	0,1
40	4042	4318	4077	4154	4441	3%	10%	7%	1%	0,3
50	4971	5524	4999	5189	5521	4%	11%	11%	1%	0,6