Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Critical paths of non-permutation and permutation flow shop scheduling problems

Daniel Alejandro Rossit[a,b*], Fernando Tohmé[b,c], Mariano Frutos[a,d], Martín Safe[b,e] and Óscar C. Vásquez[f]

[a]*Department of Engineering, Universidad Nacional del Sur, Argentina*
[b]*INMABB UNS CONICET - Av. Alem 1253, Bahía Blanca, Buenos Aires, Argentina*
[c]*Department of Economy, Universidad Nacional del Sur, Argentina*
[d]*IIESS UNS CONICET, Argentina*
[e]*Department of Mathematics, Universidad Nacional del Sur, Argentina*
[f]*Department of Industrial Engineering, Universidad de Santiago de Chile, Chile*

C H R O N I C L E

Article history:
Received July 1 2019
Received in Revised Format August 10 2019
Accepted August 10 2019
Available online August 10 2019

Keywords:
Non-permutation flow shop
Scheduling
Makespan
Critical path

A B S T R A C T

The literature on flow shop scheduling has extensively analyzed two classes of problems: permutation and non-permutation ones (PFS and NPFS). Most of the papers in this field have been just devoted on comparing the solutions obtained in both approaches. Our contribution consists of analyzing the structure of the critical paths determining the makespan of both kinds of schedules for the case of 2 jobs and *m* machines. We introduce a new characterization of the critical paths of PFS solutions as well as a decomposition procedure, yielding a representation of NPFS solutions as sequences of partial PFS ones. In structural comparisons we find cases in which NPFS solutions are dominated by PFS solutions. Numerical comparisons indicate that a wider dispersion of processing times improves the chances of obtaining optimal non-permutation schedules, in particular when this dispersion affects only a few machines.

1. Introduction

Flow Shop scheduling problems involve finding job schedules by optimizing some criteria. A flow shop system is such that all its jobs j, $j = 1, 2, \ldots, n$, visit all the machines m, $m = 1, 2, \ldots, M$, following the same technological order (all the jobs must visit the first machine, then the second machine and so on). Each job j performs an operation $O_{j,m}$ on a machine m. Each such operation has a processing time, $p_{j,m}$. The goal is to find a sequence of jobs, optimizing the objective function (here we consider only the most common objective, the makespan criterion C_{max}). One way of dealing with this NP-hard problem (Garey et al., 1976), is by focusing on the permutation flow shop scheduling problem (PFS), where the order of jobs going through a machine is repeated on all machines (in Graham et al. (1979) notation, $F|prmu|C_{max}$). This condition involves considering $n!$ possible schedules, being n the number of jobs. This permutation condition ensures the optimality of the solutions of problems with up to 3 machines (Conway et al., 1967;

* Corresponding author
E-mail: daniel.rossit@uns.edu.ar (D. A. Rossit)

2020 Growing Science Ltd.
doi: 10.5267/j.ijiec.2019.8.001

Pinedo, 2002; Błażewicz et al., 2013) since in an optimal flow shop schedule, the ordering of jobs on the first two machines (m_1 and m_2) and on the last two machines (m_{M-1} and m_M) can be the same. This implies, in turn, that in larger instances the permutation condition can exclude the optimal solution. To ensure optimality we consider the variant known as non-permutation flow shop scheduling problem (NPFS) (in Graham's notation, $F||C_{max}$). The optimal solution of the NPFS is in fact the optimal solution for the entire problem. The main disadvantage of considering the NPFS problem is that the number of possible schedules grows as $n!^M$, being n the number of jobs and M the number of machines. NPFS has thus a huge search space, even for medium size instances. So, for example, for 20 jobs and 5 machines the total number of possible schedules is 8.52×10^{91}. This means that the problem becomes easily intractable. One of the implicit conditions for the feasibility of the NPFS schedules is the existence of intermediate storage steps between successive stages in the flow of activities. If the storage capacity or the number of storage steps is limited, the complexity of the problem increases (Li & Tang 2005). According to Rossi and Lanzetta (2014) "storage facilities for NPFS schedules can either be between, on board or shared among the machines". They also point out that, in the NPFS case with sequence-dependent setups, allowing different jobs to share setups at some stages can reduce the impact of those setups. Potts et al. (1991) and Tandon et al. (1991) make, in the same sense, a stronger claim, namely that PFS yields lower quality solutions.

In this paper, we present a contribution to further understanding of these problems. We restrict our attention to problems with 2 jobs and M machines, with makespan as the objective function. We focus on the internal structure of both kinds of schedules in order to analyze the transition from PFS solutions to NPFS ones. We introduce a novel characterization of their critical paths (i.e. the sequences of activities that support the makespans of the schedules), showing that in the case of 2 jobs the critical path of NPFS schedule can be seen as a sequence of PFS critical paths. This allows us to analyze the time length of the critical path, by identifying the processing times of their component operations. Starting from these analyses we study the dominance relations between both types of schedules. We find some particular cases in which NPFS schedules are worse, in terms of their makespans, than PFS ones. Numerical explorations provide information about the impact of processing times in 2-job flow shop problems. As we will show, the critical paths of NPFS schedules tend to incorporate more operations than those of PFS schedules, but nevertheless may yield better makespans (Rossit et al., 2018). This indicates that the comparison among critical paths depends not only on the number of operations in them but also on their processing times. We design and run experiments to assess the dispersion of processing times. The results are evaluated in terms of the exact solutions of mixed-integer formulation of PFS and NPFS problems. Given the complexity of running this kind of experiments we consider a study case of a flow shop scheduling problem with 2 jobs and m-machines, which is known to be polynomially solvable (Akers, 1956; Błażewicz et al., 2007).

The plan of the paper is as follows. Section 2 reviews the literature on both NPFS and critical paths in PFS flow shop problems. Section 3 describes the critical paths of both PFS and NPFS solutions. Section 4 evaluates critical paths and presents some hypotheses about their relation to parameters of the problem, which in Section 5 are experimentally tested. Finally, Section 6 concludes.

2. Literature review

This section reviews the state of the art in the research on NPFS and PFS problems, providing a framework for our own work.

2.1. NPFS problems[1]

In the last years, with the huge increases in computing power and the development of efficient algorithms, NPFS problems have gained the attention of the scientific community. Several variations have been

[1] Rossit et al. (2018a) presents a comprehensive discussion of all the relevant aspects of NPFS problems, covering some issues that are beyond the scope of this paper.

studied under the NPFS scheme. For instance, Ying et al. (2010) study a flow shop scheduling problem that is solved by simulated annealing, showing that NPFS schedules yield better results than PFS ones. Rudek (2011) shows that for the two-machine case if learning effect is considered, PFS is not optimal anymore, being necessary to analyze NPFS schedules. Vahedi-Nouri et al. (2013) solve NPFS problems with learning effects and availability constraints, solving them with a heuristic (VFR) developed by the authors. Ziaee (2013) addresses NPFS with sequence-dependent setup times through a two-phase heuristic. Rossi and Lanzetta (2013) deal with the NPFS problem with an ACO algorithm. A particular feature of the ACO algorithm is that from the beginning it explores non-permutation solutions. In Rossi & Lanzetta (2014) they use the benchmarks of Demirkol et al. (1998) for the same problem. For these instances, their ACO algorithm outperforms other variants also used to solve NPFS problems. These authors also outline a sequence of logical steps for the passage from a physical layout (the actual production assets) to a mathematical model and its algorithmic treatment. Splitting the lot of units of each product is considered under a NPFS scheme in Shen et al. (2014) where flow shop batching with sequence-dependent family setups is investigated and solved by means of a taboo search algorithm; and in Rossit et al. (2016) a lot streaming strategy is implemented for the optimization of a NPFS problem with the makespan objective. Some preliminary results of this work were presented in Rossit et al. (2018b).

The vast majority of papers dealing with NPFS problems as their main contribution, present algorithms (meta-heuristics, in general) capable of handling them (Liao et al., 2006; Ying & Lin, 2007; Ying, 2008; Lin & Ying, 2009; Liao & Huang, 2010; Vahedi-Nouri et al., 2013; Ziaee, 2013; Rossi & Lanzetta, 2014; Benavides & Ritt 2016). These algorithms generally address the NPFS problem after a first phase in which it produces a near-optimal PFS solution: different types of procedures are applied to improve this initial PFS solution. Alternatively, Kis and Pesch (2005) review exact solution methods for PFS and NPFS problems. Very few papers focus on the structure of the problem and its relation to the solution (Potts et al., 1991; Rebaine, 2005; Nagarajan & Sviridenko, 2009; Xiao et al., 2015). The main contribution of these papers involves the characterization of the difference between the theoretical makespans of NPFS and PFS schedules under different conditions.

2.2. Contributions on critical paths in flow shop problems

The concept of *critical paths* has been closely associated with the analysis of scheduling problems since the inception of these studies, at the end of the 50's (Kelley & Walker, 1959; Kelley, 1961). A critical path is the class of activities that define the makespan of a schedule. This means that a delay in any of those activities extends the makespan of the schedule. While this concept seems most relevant in problems of project scheduling, it is widely used in all the field of scheduling in Operations Research. Nevertheless, in flow shop scheduling the interpretation of "critical path" is not straightforward because jobs, instead of operations, have to be scheduled. Furthermore, job schedules do not translate immediately into operation schedules. For instance, the solution of a flow shop problem with three jobs ($n = 3$) and six machines ($M = 6$) indicates that the jobs must be processed in the 1-2-3 order. But this does not indicate which exact sequence of operations among from 18 ($6*3$) will be the one yielding the optimal makespan. Nevertheless, there is a definite relation between the 8 operations defining the critical path ($M + (n - 1)$, according to Nagarajan et al., 2009) and the schedule of jobs. A first attempt to addressing this issue is due to Nip and Wang (2013), who combine the classical 2-machines flow shop problem with the search for the shortest paths in undirected graphs. More precisely, they combine Johnson's rule (Johnson, 1954) with Dijkstra's shortest path procedure (Dijkstra, 1959) to generate an approximation algorithm for a particular PFS problem. This algorithm implicitly uses the critical path. Nip et al. (2015) extend Nip & Wang (2013) towards 3-machines flow shop problems, applying again a shortest path solution. They describe the critical path structure for the corresponding PFS problems. Furthermore, they show that their approach, when applied to 3 or more machines, is NP-hard and propose an approximation algorithm to solve those cases. Shang et al. (2017) also use critical paths in the minimization of makespan in flow shop problems. Here, the authors introduce a moderately exact exponential algorithm for PFS

problems with 3 machines. The motivation for introducing such variant of a dynamic programming solution is that in many instances an exponential variant can be better than a polynomial one. For example, it is faster for an $O(1.1^n)$ instance than for a $O(n^4)$ for every $n < 224$.[2] This algorithm goes through a sequence of partial critical schedules converging to the final schedule, where the former are schedules in which operations start as soon as possible.

2.3. Comments on the literature

We have presented the main contributions in the area of NPFS problems with makespan as objective as well as those on critical paths in flow shop problems. As indicated, the literature focuses on handling the large computational requirements of NPFS problems, comparing in experiments the PFS and NPFS solutions. On the other hand, the literature on critical paths in flow shop problems focuses only on PFS problems. None of those works compares the structure of critical paths of both problems. This is exactly what we aim to accomplish in this article. This allows us to generate PFS solutions similar to those in Nip et al. (2015), but instead of being restricted to 3 machines, we extend them to m-machines. In turn, unlike Nip et al. (2015), who add operations according to the jobs to which they belong, we index them by the machines. This allows us to extend the description to NPFS problems.

3. Critical paths

In this section, we will present an analysis of the critical path structures of both PFS and NPFS (from now on, *critical path* will be denoted CP). It is worth to mention that both structures are symmetric. The optimal makespan obtained in a forward analysis is equal to the one obtained on a backward one (inverting the routing of the jobs, i.e., jobs are processed first on the last machine m_M, next on m_{M-1} and so on). It is also interesting to recall the properties proposed by Conway et al. (1967). In this sense, Blazewicz et al. (2007), provide direct and simple enunciations as well as their corresponding proofs.

Proposition 1. *For $F||C_{max}$, there exists an optimal solution with the same processing ordering on the first two machines.*

Proof. To see this, consider any solution in which the orders differ on the first two machines. Then, there must exist a pair of adjacent jobs, say a and b, on the first machine permutation, appearing in reverse order in the permutation on the second machine. But these two jobs can be reversed on the first machine without increasing the starting time (and thus the completion time) of any job on the second machine. Inductively, we can repeat this pairwise switching sequence until the permutation in the first machine agrees with the (original) order on the second. □

An immediate consequence of Proposition 1 is that $F2||C_{max}$ and $F2|prmu|C_{max}$ are equivalent, where "$F2||C_{max}$" means a 2-machines flow shop problem in which makespan is minimized. That is, in the case of a 2-machines flow shop, the optimal schedule is a permutation. Consequently, the $F2||C_{max}$ is solvable in polynomial time using Johnson's algorithm (Johnson 1954). Moreover, due to the symmetry of the $F||C_{max}$, the following property holds:

Proposition 2. *For $F||C_{max}$, there exists an optimal solution having the same processing ordering on the last two machines.*

Proof. Analogous to the proof of Proposition 1. □

Propositions 1 and 2 are well known in the literature on NPFS schedules (Rossit et al., 2018). They basically correspond to Theorems 5.1 and 5.2 of Conway et al. (1967), which in turn extend and make more precise Lemma 2 of (Johnson, 1954). Our contribution aims to find *new* results in the context of

[2] For more on this, see Woeginger (2003) and Fomin & Kratsch (2010).

the classical Flow Shop problem. As a motivation, we present in Example 1, a numerical examination of the smallest case in which Propositions 1 and 2 no longer ensure optimality. The analysis of this example will yield new concepts that will be defined precisely in the rest of this paper jointly with novel results involving them

Example 1. Let us consider a numerical example of 2 jobs and 4 machines. This is the smallest possible case in which PFS is not optimal for makespan ($M > 3$). The instance selected is described in Table 1. 2 jobs (j_1 and j_2) have to be scheduled on 4 machines (m_1, m_2, m_3 and m_4) in order to minimize makespan.

Table 1
Processing times of Example 1

	m_1	m_2	m_3	m_4
j_1	4	1	1	4
j_2	1	4	4	1

The possible PFS schedules are two: j_1-j_2 and j_2 - j_1, both schedules yielding a makespan of 14. For NPFS schedules, we have other alternatives (remember that switching the sequence between the first and the last two machines does not improve the PFS makespan): sequencing j_1-j_2 on machines m_1 and m_2, and j_2 - j_1 for m_3 and m_4; and j_2 - j_1 for m_1 and m_2, and j_1-j_2 for m_3 and m_4. The latter sequence (j_2- j_1; j_1-j_2) is the optimal one, with a makespan of 12. The Gantt representation of the optimal NPFS schedule is illustrated in Fig. 1, as follows:

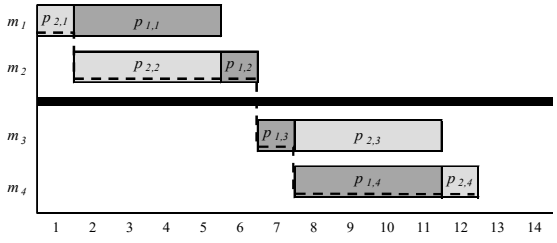

Fig. 1. Optimal schedule, which is a NPFS solution

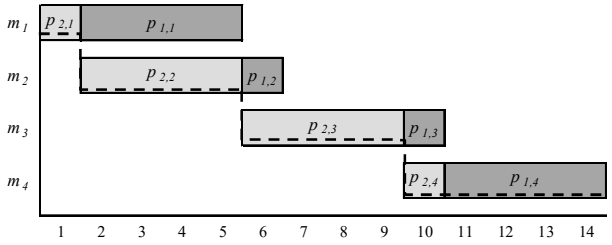

Fig. 2. Best possible PFS solution

From Fig. 1, we can deduce that a switch in the sequence happens in the passage from m_2 to m_3, indicated with a thick black horizontal line. m_3 is considered a *switching machine* since it switches the job ordering of the previous machine. Let us compare this with Fig. 2, which shows the best PFS schedule for the problem. A simple inspection of both figures indicates that the NPFS solution has a smaller makespan than the PFS one. In the sequences shown in Fig. 1 and Fig. 2 the critical paths are drawn in dashed lines. It is of interest to notice that these critical paths exhibit indifferences between including or not some activities. So, in Figure 1, the same makespan is obtained by including the second activity of j_2 or the first of j_1, an indifference that can be arbitrarily resolved. If we enumerate the activities in the critical paths of the NPFS schedule (Figure 1) we get the following: i) the first activity of j_2, ii) the second activity of j_2, iii) the second activity of j_1, iv) the third activity of j_1, v) the fourth activity of j_1 and vi) the fourth activity of j_2, a total of 6 activities. An analogous enumeration of the CP of the PFS schedule (Figure 2) yields: i) the first activity of j_2, ii) the second activity of j_2, iii) the third activity of j_2, iv) the fourth activity of j_2, v) the fourth activity of j_1, a total of 5 activities. Comparing both CPs, we can see that the minimal makespan is obtained at the longest CP in terms of activities, something that is not quite intuitive. The processing times have a large impact on the characterization of the makespan and the CP of a schedule since the entries in Table 1 are not homogeneous. We can deepen our analysis by considering the critical paths of solutions to detect the processing times that determine the makespan.

3.1. Critical paths of PFS schedules

The PFS-CP described and illustrated in the previous section provides a first insight into a description of CPs. For a more thorough characterization consider Fig. 3, which represents a generic schedule of 2 jobs

(j_1 and j_2) on 4 machines (m_1, m_2, m_3 and m_4) flow shop. The black bars marked with "x" indicate that the beginning of a new activity depends on a disjunctive constraint, either the end of a job at the previous machine or the release of the current machine from the previous job. For instance, considering "x1" in Fig. 3, activity $O_{2,2}$ will begin at the maximum between the completion of $O_{2,1}$ and the completion of $O_{1,2}$. As an initial description of the structure of critical paths consider the following proposition:

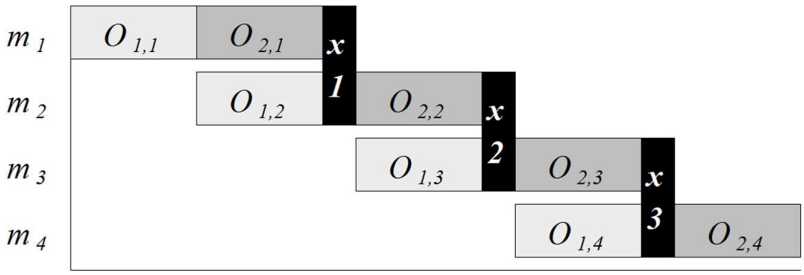

Fig. 3 Generic case of PFS schedule with 2 jobs (j_1 and j_2) and 4 machines

Proposition 3. *Each CP of Fig. 3:*

a) *always starts with the first operation of the first job in the schedule,*
b) *always finishes with the last operation of the last job in the sequence,*
c) *includes sequentially the operations from one job up to a stage, where it starts including the operations of the other job,*
d) *at the aforementioned stage, it includes operations from both jobs (the previous and the next one),*
e) *the rest of the stages only include operations from one of the jobs*.

Proof. The proof follows, without loss of generality, from the careful inspection of Fig. 3. We can see, that $O_{1,1}$ and $O_{2,4}$ will be part of every possible CP (3.a and 3.b). To study all the possible CPs in Figure 3, let us run a backward analysis. $O_{2,4}$ is the last activity included in the CP, at instant "x3". If $O_{1,4}$ is included in the CP, then $O_{2,3}$ finishes before or at the same time as $O_{1,4}$. This implies, that all the other possible paths from $O_{1,1}$ to "x3" that include $O_{2,3}$ are shorter or equal (in terms of makespan) than the ones which include $O_{1,4}$. Furthermore, since the start of $O_{1,4}$ does not depend at all on the activities of job j_2, "x2" will be defined by $O_{1,3}$, and "x1" by $O_{1,2}$, and finally $O_{1,1}$. The CP will be: $O_{1,1}$ - $O_{1,2}$ - $O_{1,3}$ - $O_{1,4}$ - $O_{2,4}$ (five activities). This first insight allows us to state that for the two jobs problem, once a tie (in terms of makespan) at an "x" instant is broken by an activity of the first scheduled job, all the previous ties at "x" instants are also broken by activities of the same job, supporting 3.c.

On the other hand, if "x3" is defined by $O_{2,3}$, then "x2" must be defined by a new comparison. If "x2" is defined by $O_{1,3}$, since $O_{1,3}$ does not depend on activities from the second job, the previous "x" instants are defined by activities from the first job, and the final CP is: $O_{1,1}$ - $O_{1,2}$ - $O_{1,3}$ - $O_{2,3}$ -$O_{2,4}$ (five activities) (3.d and 3.e). The same analysis can be carried out for "x2" if it is defined by $O_{2,2}$, as well as for "x1". Notice that, if from all the points "x" the only one defined by an activity of j_1 is the last one, the CP will consist of all the activities of j_1 and only includes the last activity of j_2 (4.c). This can be verified by considering a Gantt diagram where activity $O_{1,4}$ finishes after the end of $O_{2,3}$, and for the rest of the "x" instants, activities of j_1 finish before their counterparts in j_2. □

Proposition 3 facilitates the identification of the operations (and thus their processing times) included in the critical path. Nip et al. (2015) analyzed the $F|prmu|C_{max}$ problem with n jobs and 3 machines, finding that its makespan is:

$$\sum_{j=1}^{j=u} p_{j,1} + \sum_{k=u}^{k=v} p_{k,2} + \sum_{l=v}^{l=J} p_{l,3} = C_{max} \qquad \text{for some } u, v \in J \tag{1}$$

The difference between theirs and our approach is that Nip and coauthors restrict the number of machines, while we do the same with the number of jobs. That is why expression (1) applies in our framework only for M ≤ 3. Then, up from Proposition 3 we can derive an alternative characterization of the makespan under a similar structure:

Proposition 4. *The makespan of a F|prmu|C_{max} scheduling problem of two jobs obeys the following specification:*

$$\sum_{i=1}^{i=m'} p_{1,i} + \sum_{i'=m'}^{M} p_{2,i'} = C_{max} \qquad \text{for some } m' \epsilon M\,. \tag{2}$$

Proof. If follows immediately from the proof of Proposition 3. □

Expression (2) indicates that the makespan is obtained as a sum of the processing times of the operations in its critical path. It includes the operations of the first job up to a machine m'. After that, Eq. (2) adds the sequence of operations of the second job. Machine m' satisfies Proposition 3.d, being an instance of the following definition:

Definition 1 (*Critical machine*): a machine is *critical* if operations $O_{1,m'}$ and $O_{2,m'}$ belong to the critical path.

Then, all the operations carried out on a critical machine will have a direct impact on *F|prmu|*C_{max} in 2 jobs, while the rest of the machines will impact only on a single operation. Thus, the empirical expression (2) represents all the possible CPs from a PFS schedule of 2 jobs j_1 - j_2. It yields, furthermore, the number of operations expected by (Nagarajan & Sviridenko 2009) for a permutation CP, $t = M + n - 1$, being *t*: number of activities, *m*: number of machines and *n*: number of jobs. In our example of 4 machines and 2 jobs, *t = 5* operations. Another important feature of expression (2) is that it can be extended to a larger number of machines. It can be easily shown that, by adding to Figure 3 one machine, the correspondent activities and another instant "x", yielding a straightforward extension of (2). To represent the set of CPs of schedule j_2 - j_1, it is enough to reorder the terms in (2): first add the activities of j_2 up to *m'*, and then those of j_1 from *m'* up to the last machine *M*.

3.2. Critical path of a NPFS schedule

We intend to find a description of the CP of a NPFS schedule, similar to the one of PFS, in order to compare both. Considering again the numerical example at the beginning of Section 2, we can see that the CPs of both the NPFS and the PFS solutions include activities from the two jobs. However, the comparison of Figures 1 and 2 shows that there exist differences between the two cases. The main ones are: the NPFS-CP has one activity more than the PFS-CP, and in the NPFS-CP, unlike the PFS-CP, the activities from both jobs alternate. This suggests the following result:

Proposition 5. *A F||*C_{max} *scheduling problem of two jobs and four machines can be decomposed into two F|prmu|*C_{max} *scheduling sub-problems of two jobs and two machines each. The makespan is obtained as the sum of the makespans of the sub-problems. The set of CPs of a F||*C_{max} *scheduling problem can be described by the following expression:*

$$\left(\sum_{i=1}^{i=m'} p_{1,i} + \sum_{i'=m'}^{i'=m^*} p_{2,i'}\right) + \left(\sum_{l=m^*}^{l=m''} p_{2,l} + \sum_{l'=m''}^{M} p_{1,l'}\right), for\ some\ m', m^*, m'' \in M | m' < m^* \leq m'' \tag{3}$$

where, m represents the switching machine at which the jobs are switched, while m' and m'' are critical machines.*

Proof. Consider Figure 4. We can see a NPFS schedule of 2 jobs and 4 machines which switches the job ordering from m_2 to m_3.

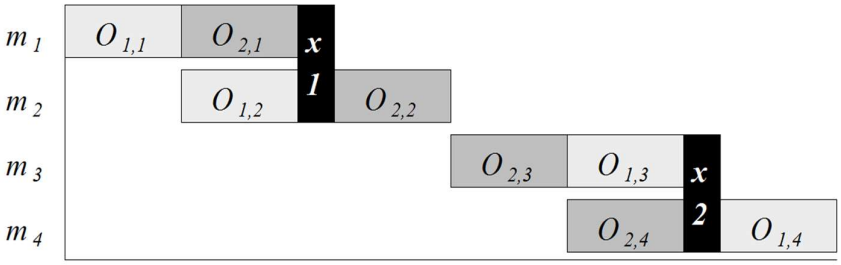

Fig. 4. Generic case of a NPFS schedule of 2 jobs (j_1 and j_2) and 4 machines.

The first feature in Fig. 4 that supports the claim is that $O_{2,3}$ will not start before the completion of $O_{2,2}$. This indicates that the total completion time (makespan) can be decomposed into the sum of the time elapsed up to the completion of $O_{2,2}$ and the rest of the time up to the completion of $O_{1,4}$. Furthermore, a backward analysis shows that the NPFS schedule of Fig. 4, can be decomposed into two minor PFS schedules: one of machines m_1 and m_2, and the other of machines m_3 and m_4. To assess the validity of considering smaller PFS schedules, let us focus on the schedule of m_3 and m_4 in Figure 4, and run a backward analysis. The operation $O_{1,4}$ is included in every possible CP. Then comes instant "x2", which if defined by $O_{2,4}$, indicating that the CP on machines m_3 and m_4 is $O_{2,3}$ - $O_{2,4}$ - $O_{1,4}$. This CP has two aspects that allow considering this smaller system as a PFS schedule: it has three operations, and can be represented by expression (2). For machines m_1 and m_2 a similar analysis shows that a possible CP can be $O_{1,1}$ - $O_{1,2}$ - $O_{2,2}$. Again, the CP corresponds to a PFS-CP. Finally, the CP of the NPFS schedule can be $O_{1,1}$ - $O_{1,2}$ - $O_{2,2}$ - $O_{2,3}$ - $O_{2,4}$ - $O_{1,4}$, which has 6 operations (as in the numerical example of Fig. 1). And it can be represented by expression (2), where m' and m'' are m_2 and m_4, respectively, and the switching machine m^* is m_2. □

Let us note that the switching machine m^* must satisfy Propositions 1 and 2. Consequently, for a 4-machine flow shop configuration, m^* can be only m_2. This clarification raises two new questions: what happens to expression (3) when the number of machines is larger than 4? What happens when m^* can have different or multiple values? Clearly, expression (3) depends on the size of the system and on the type of NPFS schedule: when the number of machines increases, m^* can take different and multiple values (Propositions 1 and 2 restrict the schedules only for the first and the last two machines). An answer is provided by Proposition 6 which considers an instance of Proposition 5, with $M = 5$ and two switching machines. For the sake of clarity, from now on we will denote with S the number of switching machines.

Proposition 6. *A $F||C_{max}$ scheduling problem of two jobs and five machines can be decomposed into two $F|prmu|C_{max}$ scheduling sub-problems, for each switch in the job ordering. The makespan is obtained as the sum of the makespans of the sub-problems.*

Proof. Consider Fig. 5, as a new example of 2 jobs and 5 machines. Applying Propositions 1 and 2, we know that there is no benefit in switching the job ordering from m_1 to m_2 and from m_4 to m_5, thus leaving $n!^{M-2}$ possible schedules. In this new example: $2!^{(5-2)} = 8$ possible schedules. From these 8 possible schedules, 2 are PFS (j_1 - j_2 and j_2 - j_1), and the other 6 are NPFS schedules. Half of the 6 NPFS schedules start with the sequence j_1 - j_2 and the other half with the reverse sequence. Now, consider only NPFS schedules starting with j_1 - j_2, i.e. 3 different schedules. Remember that now m^* can be m_2 or m_3 or both. In the cases where m^* takes a unique value, i.e. m_2 or m_3, expression (3) is valid according to the decomposition procedure shown in Fig. 4. The resulting CPs of the decomposed flow shop subsystems can be described as PFS-CPs, and expression (2) is valid for m machines. But when m^* is both m_2 and m_3, expression (2) is no longer valid.

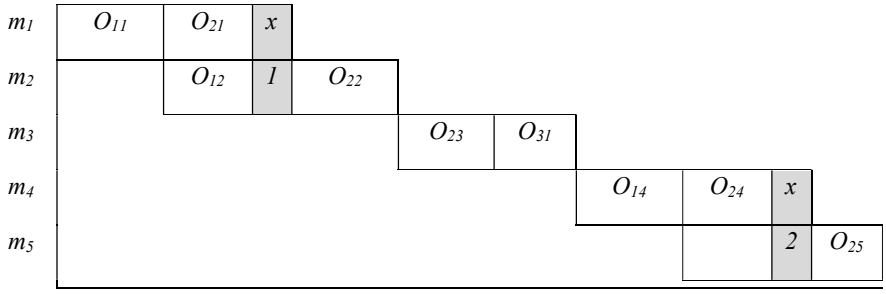

Fig. 5. NPFS schedule of 2 jobs and 5 machines, with two switching machines (m_2 and m_3)

If expression (3) is applied to the schedule in Figure 5 the resulting CP will not represent the actual CP. Expression (3) is valid only for a single switching machine m^*. Consider the case in which the schedule in Figure 5 is forced to pick only one of the switching machines, for instance if m_2 is discarded as switching machine and m_3 becomes m^*. Then, the CP obtained in expression (3) will have 7 activities, unlike the actual optimal schedule. Running a backward analysis, we can see that the actual CP of Figure 5 ends with activity $O_{2,5}$, starting at "x2". Suppose $O_{2,4}$ is included and successively also $O_{1,4}$, $O_{1,3}$, $O_{2,3}$, $O_{2,2}$. Then we get to the switching instant "x1". Assume that $O_{1,2}$ is also included, as well as $O_{1,1}$, yielding $O_{1,1}$ - $O_{1,2}$ - $O_{2,2}$ - $O_{2,3}$ - $O_{1,3}$ - $O_{1,4}$ - $O_{2,4}$ - $O_{2,5}$ as CP, with a total of 8 activities. Consequently, we need a new formulation, with an expression similar to (1) and (2). Based on the decomposition procedure we have been applying, this schedule can be decomposed into a discrete number of flow shop independent systems. More precisely, the schedule in Figure 5 can be decomposed into three smaller systems: i) m_1 and m_2, ii) m_3 and iii) m_4 and m_5. Summing up their *smaller*-makespans we obtain the global makespan (activity $O_{2,3}$ will not start before the completion of $O_{2,2}$, and $O_{1,4}$ will not start before the completion of $O_{1,3}$). Subsequently, adding the CPs of the smaller systems we get the global CP. This is represented by expression (4), where m^* and m^{**} are the switching machines:

$$\left(\sum_{i=1}^{i=m'} p_{1,i} + \sum_{i'=m'}^{i'=m^*} p_{2,i'}\right) + \left(\sum_{l=m^*}^{l=m''} p_{2,l} + \sum_{l'=m''}^{m^{**}} p_{1,l'}\right) + \left(\sum_{k=m^{**}}^{k=m'''} p_{1,k} + \sum_{k'=m'''}^{M} p_{2,k'}\right), \quad for\ some\ m', m^*, m'', m^{**}, m''' \in M | m' < m^* \leq m'' < m^{**} \leq m''' \tag{4}$$

In Fig. 5, m_2 and m_3 are the switching machines. A particular feature of the CP model presented here is that it also applies to the single machine case. The middle summand of expression (4) represents the portion of CP captured by m_3 in Fig. 5.

From the comparison of expressions (3) and (4), we can further specify the decomposition procedure:

Decomposition method. *A $F||C_{max}$ scheduling problem of two jobs can be decomposed into as many $F|prmu|C_{max}$ scheduling sub-problems as the number of switches in the job ordering plus one. The makespan is obtained as the sum of the makespans of the sub-problems. The set of the CPs of the sub-problems can be described by expression* (2)*.*

This enunciation leads to some interesting results for the 2 jobs case. Whenever a switch is present, the decomposition procedure can be applied, and a new activity is added to the CP. Then, expression (2) and its expansions in (3) and (4) can be conceived as a recursive formulation in an algorithm intended to generate flow shop critical paths.

4. Comparing the critical paths of NPFS and PFS for 2 jobs and *m* machines

As mentioned before, we intend to compare NPFS and PFS schedules. Expression (2) and the decomposition method allow the generation of the entire set of possible CPs for each possible schedule. With this set of CPs available, it becomes possible to run comparisons. The idea is to determine the degree of similarity between the CPs. Initially, we know that NPFS-CPs are longer than PFS-CPs for the

same instance. Even more, NPFS-CPs can be much longer than PFS-CPs, since every switch adds extra activities to the CP. Consequently, the length of NPFS-CPs, measured by the number of operations, is longer than that of PFS-CPs. We can hint that NPFS-CPs will yield worse makespans than PFS-CPs for a certain number of operations. The corresponding concept of dominance is expressed as follows:

Definition 2. *If for some NPFS Schedule we have that:*

$$\sum_{p_{j,i} \in NPFS-CP} p_{j,i} \geq \sum_{p_{j,i} \in PFS-CP} p_{j,i}\,, \tag{4}$$

the schedule will be ***dominated*** *by a PFS schedule.*

This definition establishes that, if the sum of the processing times of the operations in the NPFS-CP is as large as that of the operations in the PFS-CP, the former will be dominated by the PFS-CP. Since the critical paths yield the makespans of the schedules, a shorter PFS-CP will indicate that the PFS schedule has a better makespan than NPFS. A particular case of dominance arises when NPFS-CP contains all the operations of the PFS-CP and thus the latter schedule dominates the former one. Figure 6, illustrates this particular dominance case in the case of two jobs and five machines. The CP at the top is a PFS-CP while the CP at the bottom is a NPFS-CP. It is easy to see that the PFS-CP operations are all captured by the NPFS-CP, and that there are two extra activities in the NPFS-CP, indicated by circles. Then, the NPFS-CP is dominated by the PFS-CP.

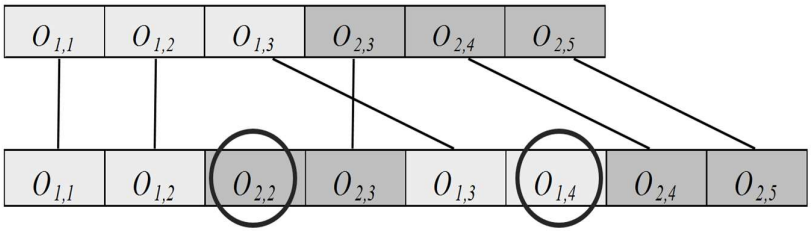

Fig. 6 Critical path dominance

We now present other, less immediate, cases in which Definition 2 applies. Recall that the number of switching machines is denoted S, and that according to Propositions 1 and 2, $S \leq M - 3$.

Proposition 7. *Consider an instance of two jobs and m machines ($M > 4$). If M is odd, the number of switching machines in the NPFS schedule is $M - 3$, and the operations are such that $p_{1,2} > p_{2,1}$ and $p_{1,M} > p_{2,M-1}$. Then, the NPFS schedule is dominated by the PFS schedule.*

Proof. We prove this result by induction. As the base case consider $M = 5$ (the smallest case to which the claim applies) and an NPFS schedule with $S = M - 3 = 2$ switches (illustrated in Figure 3) and a PFS schedule starting with the same operation as the NPFS one. We will show that the PFS solution dominates the NPFS schedule.

By a straightforward application of the decomposition method we know that the makespan of the NPFS schedule with S switches can be rewritten as the sum of the makespans of $S + 1$ PFS sub-problems. These involve: i) operations on m_1 and m_2, ii) operations on m_3 and iii) operations executed in m_4 and m_5, as shown in Figure 3. Each "sub-makespan" can be obtained in terms of expression (2). Since ii) includes operations of the two jobs in m_3, Definition 1 indicates that it is a critical machine and thus the makespan in that case is just the sum of the processing times on those operations.

Since M is an odd number, S is even and thus the schedules for subproblems i) and iii) coincide. In i), the operations $O_{1,1}$ and $O_{2,2}$ are necessarily part of its critical path (Proposition 3, a and b), while only one from $O_{1,2}$ or $O_{2,1}$ will be included. Since, by definition, $p_{1,2} > p_{2,1}$, $O_{2,2}$ can start only after $O_{1,2}$ has finished. Then, the makespan of i) is: $p_{1,1} + p_{1,2} + p_{2,2}$.

For iii) the analysis is similar: $O_{1,4}$ and $O_{2,5}$ are included in the critical path (again according to Proposition 3.a,b) and from the other two, only $O_{1,5}$ belongs to the critical path (by definition $p_{1,M} > p_{2,M-1}$). Then, the makespan of iii) is $p_{1,4} + p_{1,5} + p_{2,5}$.

Therefore, the total makespan of the NPFS schedule is: $p_{1,1} + p_{1,2} + p_{2,2} + p_{2,3} + p_{1,3} + p_{1,4} + p_{1,5} + p_{2,5}$.

We can now turn to the PFS schedules and the possible critical paths describing its makespan. We resort to Definition 1, noting that the simplest case is when the PFS-CP has m_5 as its critical machine. The ensuing makespan would be $p_{1,1} + p_{1,2} + p_{1,3} + p_{1,4} + p_{1,5} + p_{2,5}$, implying that the corresponding operations would be included in the NPFS-CP, and thus that the makespan of NPFS-CP would be longer by adding the processing times of operations $O_{2,2}$ and $O_{2,3}$.

With respect to the rest of the PFS-CP, we know that they cannot have m_1 as critical machine since it would have to include $O_{2,1}$, not triggering the start of $O_{2,2}$ (which, as said, depends on $O_{1,2}$) and thus would not yield the makespan of a PFS solution. The same would happen if m_4 were a critical machine.

In the case that the critical machine of the PFS-CP is m_2, this critical path would include $O_{1,1}$, $O_{1,2}$, $O_{2,2}$, $O_{2,3}$, $O_{2,4}$ and $O_{2,5}$ as follows from equation (2). Other than $O_{2,4}$ all the rest of the operations are in NPFS-CP, contributing to the makespan of both the PFS-CP and the NPFS-CP. Since $p_{2,4} < p_{1,5}$, corresponding to $O_{1,5}$, which belongs to NPFS-CP, it follows that the makespan of NPFS-CP is longer than that of PFS-CP. A similar analysis can be carried out in the case that the critical machine of PFS-CP is m_3.

The inductive step involves the cases in which $M > 5$. It follows from Propositions 1 and 2 that, since $S = M - 3$, the switches will be consecutive. Accordingly, a NPFS-CP can be decomposed in as many as $S + 1$ PFS problems, with as many one-machine PFS problems as $S - 2$. □

Proposition 8. *Given an instance of two jobs and M machines, if M is even, S of the NPFS schedule is M – 3, and* $p_{1,2} > p_{2,1}$, *the NPFS schedule is dominated by the PFS schedule.*

Proof. We will prove this claim by induction. As base case consider an instance with $M = 6$ and a NPFS schedule with $S = M - 3 = 3$ switches. We will compare it to a PFS schedule starting with the same operation as the NPFS schedule. We will show that this PFS schedule dominates the NPFS one.

The decomposition method indicates that the makespan of the NPFS schedule with S switches can be rewritten as the sum of the makespans of $S + 1$ PFS sub-problems. These involve: i) the operations executed on m_1 and m_2, ii) the operations on m_3, iii) the operations on m_4 and iv) the operations carried out on m_5 and m_6. Each of them yields a PFS sub-problem in which the makespan can be obtained from Eq. (2). Each one of problems ii) and iii) involves a single machine that, according to Definition 1, is a critical machine. Then, the makespan of each of these two sub-problems obtains as the sum of the processing times of the two operations executed on the corresponding single machine.

Since M is even, S is odd, and thus the schedules for sub-problems i) and iv) do not coincide. In i) operations $O_{1,1}$ and $O_{2,2}$ are necessarily part of the critical path, while only one of $O_{1,2}$ and $O_{2,1}$ belongs to it. Since $p_{1,2} > p_{2,1}$, $O_{2,2}$ can start only after $O_{1,2}$ has finished its execution. Then, the makespan of i) is: $p_{1,1} + p_{1,2} + p_{2,2}$.

For iv), $O_{2,5}$ and $O_{1,6}$ must both be in the critical path, together with a third operation, the one that takes longer from $O_{1,5}$ and $O_{2,6}$. For example, if $p_{1,5} > p_{2,6}$, the makespan of iv) is $p_{2,5} + p_{1,5} + p_{1,6}$.

Then, the resulting NPFS schedule has makespan: $p_{1,1} + p_{1,2} + p_{2,2} + p_{2,3} + p_{1,3} + p_{1,4} + p_{2,4} + p_{2,5} + p_{1,5}$ (or $p_{2,6}$) + $p_{1,6}$.

Now we turn to the analysis of the PFS schedule and the possible critical paths supporting its makespan. For this we will use again Definition 1. The easiest case is when machine m_6 is critical for the PFS problem, since then the makespan will be given by $p_{1,1} + p_{1,2} + p_{1,3} + p_{1,4} + p_{1,5} + p_{1,6} + p_{2,6}$, implying that

only one of the corresponding operations does not correspond in the NPFS-CP, namely $O_{2,6}$. But this is so because we assumed that $p_{1,5} > p_{2,6}$, i.e. the critical path includes an operation that takes longer than $O_{2,6}$ making the makespan corresponding to the NPFS-CP longer than that of the PFS-CP. Notice that if in problem iv) we had assumed that $p_{1,5} < p_{2,6}$, $O_{2,6}$ would be included in the NPFS-CP excluding $O_{1,5}$, yielding again that the makespan of the NPFS-CP must be longer than that of the PFS-CP.

We have to consider other possible positions for the critical machine in the PFS schedule. There are three possible cases: the critical machine appears before $M - 1$, it is precisely the $M - 1$ machine or it is the last machine, M. In the first case, $1 < m^* < M - 1$, the operations in the PFS-CP up to the $M - 1$ machine will belong also to the NPFS-CP since the NPFS schedule switches jobs consecutively contemplating all the operations executed between machines m_2 and m_{M-2}. With respect to machines m_{M-1} and m_M, the PFS-CP will include operations $O_{2,5}$ and $O_{2,6}$. The NPFS-CP, instead, includes only one of these two operations. $O_{2,5}$ is necessarily included (the first operation of the last sub-problem), but $O_{2,6}$ can or cannot be included. In the case that it is not included, it must be because another one was, with longer processing time ($O_{1,5}$ in the previous case), making again the makespan of the NPFS-CP longer than that of the PFS-CP. In case that $p_{1,5} < p_{2,6}$, $O_{2,6}$ becomes included in the NPFS-CP as well as in the PFS-CP. Since the NPFS-CP incorporates more operations than the PFS-CP, the makespan of the NPFS-CP will be again longer than that of the PFS-CP.

As said, another case arises in the PFS-CP when $m^* = M - 1$. Then, the operations in the PFS-CP on the last two machines are $O_{1,5}$, $O_{2,5}$, and $O_{2,6}$. At most only one of $O_{1,5}$ or $O_{2,6}$ might not be present in the NPFS-CP. The same happens when $m^* = M$. In these two cases the result is similar to the one for $m^* < M - 1$, since the NPFS-CP will include either all the operations of the PFS-CP or only one different but with longer processing time, making the makespan of the former longer than that of the latter.

The claim is also valid in the inductive step, when $M > 6$, since, again as in the proof of Proposition 7, the switches must be consecutive since $S = M - 3$. □

5. Experimental evaluation

In this section we run an experimental analysis of the impact of the distribution of processing times on the optimal solutions of the PFS and NPFS problems. We first present MIP (Mixed-Integer Programming) versions of the permutation and non-permutation formulations of problems. Then, we generate different classes of scenarios for which we will obtain the solutions. Finally, we compare the ensuing schedules.

5.1. MIP formulations

We first present the PFS formulation, which is simpler. Then we will present the NPFS formulation, given in terms of its differences with the PFS one.

PFS model.

- Sets

Jobs $\{j\}$
Machines $\{m\}$

- Parameters

$p_{j,m}$ Processing time of product j on machine m
Ω large positive number

- Variables

$C_{j,m}$ Completion time of job j on machine m
$x_{j',j}$ Binary variable: 1 if job j' is processed before job j

$min\ C_{max}$

subject to:

$$C_{j,m} \geq C_{j,m-1} + p_{j,m}\,, \qquad \forall j, m > 1 \tag{5}$$

$$C_{j,m} \geq C_{j',m} + p_{j,m} - \left(1 - x_{j',j}\right) \cdot \Omega\,, \qquad \forall m, j' \neq j \tag{6}$$

$$C_{j',m} \geq C_{j,m} + p_{j',m} - x_{j',j} \cdot \Omega\,, \qquad \forall m, j' \neq j \tag{7}$$

$$x_{j',j} + x_{j,j'} = 1\,, \qquad j \neq j' \tag{8}$$

$$C_{max} \geq C_{j,m}\,, \qquad \forall j, \forall m \tag{9}$$

$$C_{j,m} > 0;\ x_{j,j'}\ \{0,1\} \tag{10}$$

The objective function to be minimized is the makespan. Constraint (5) indicates the precedence of activities, i.e. a job must stop being processed on a machine before passing to the next one. Constraints (6) and (7) work together indicating the ordering of jobs. If job *j'* is processed before job *j*, then $x_{j',j}$ becomes 1 and constraint (6) becomes active, while constraint (7) turns redundant. In expression (8) the logical order is respected: if job *j'* is processed before job *j*, the converse cannot be valid. The makespan is defined by inequality (9) while (10) expresses the condition of feasibility.

NPFS model

The NPFS model is similar to the PFS model, being the main difference that the job ordering among machines can change. For this, the variable *x* becomes now indexed by the set *m* of machines, as follows:

$x_{j',j,m}$: Binary variable: 1 if job *j'* is processed before job *j* on machine *m*.

Then the equations that are modified are Eq. (6), Eq. (7) and Eq. (8) and we get,

$$C_{j,m} \geq C_{j',m} + p_{j,m} - \left(1 - x_{j',j,m}\right) \cdot \Omega\,, \qquad \forall m, j' \neq j \tag{11}$$

$$C_{j',m} \geq C_{j,m} + p_{j',m} - x_{j',j} \cdot \Omega\,, \qquad \forall m, j' \neq j \tag{12}$$

$$x_{j',j,m} + x_{j,j',m} = 1\,, \qquad j \neq j', \forall m \tag{13}$$

Here constraints (11) and (12) are analogous to (6) and (7), but now the order among machines can change. Constraint (13) is a similar logical condition as (8) but evaluated on every machine.

5.2. Test Scenarios

We intend to evaluate the claims proven in the previous sections. We proceed by generating alternative scenarios, such that one scenario differs from another in the dispersion of values of the processing times $p_{j,m}$. The processing times of the different scenarios are generated by means of the following formula:

$$p_{j,m} = 2^{N(0,\sigma)} \tag{14}$$

The exponent here corresponds to a Gaussian distribution with mean 0 and standard deviation σ. We can thus use σ to vary the range of values of $p_{j,m}$. (Microsoft Excel was used to generate the random samples of the distribution). We designed three experiments. In the first one the processing times of all the operations are generated with the same dispersion: each $p_{j,m}$ obeys Eq. (14). Four different values of σ where considered for each number of machines in the range [4;20]. This range was chosen because M = 4 is the minimal number of machines for which NPFS can be defined, while M = 20 is the largest number

of instances enumerated in Taillard (1993). The four values of σ were 0.5, 1, 1.5 and 2. For each pair σ and M, we run 30 different scenarios.

In the second experiment the settings are the same as in the previous one, except that different standard deviations were considered for the two jobs. For instance, for one job σ was set at 0.5, while for the other job σ was chosen to be 1, 1.5 and 2. M is drawn again from the same range, [4;20]. Again, 30 scenarios were generated for each pair M and σ. Finally, the third experiment made the standard deviation vary among the machines. In this case we considered M to be 10, 15 or 20. Almost all the machines were assigned $p_{j,m}$ values distributed according to $\sigma = 0.5$, except for some machines. In the case $M = 10$, the exceptions were m_4, m_5, and m_6, with processing times drawn from distributions with σ =1, 1.5 and 2. In the case that $M = 15$, for $p_{j,6...9}$ we assumed again the same three values of σ. For $M = 20$ we assumed the same for $p_{j,8...12}$ with, again, σ =1, 1.5 and 2. As in the previous experimental designs we ran 30 scenarios. All these experiments were performed in an Intel Core i5 with 8 GB of main memory using CPLEX 12.5 as MIP solver.

5.3. Results

Tables 2, 3 and 4, summarize the results for the three experimental designs. Each table represents one of those designs, but in all of them the rows present the instances defined by the values of M. The column NPFS indicates the percentage of scenarios in which NPFS schedules solved optimally the problem (over a total of 30 scenarios). Column GAP indicates the average difference between the optimal NPFS solution and that of the best PFS schedule (reported only in the cases that NPFS yields the optimal solutions). Column "Row Avg." presents the average values of the NPFS solution and the GAP for each row. In turn, "Col. Avg." presents the averages per column. The lower right entries of the table report the averages for the entire experiment.

Table 2
Average results of the first experiment: the processing times of all the operations are drawn from the same distribution, and thus have the same dispersion

M	σ = 0.5		σ = 1		σ = 1.5		σ = 2		Row Avg.	
	NPFS	GAP	NPFS	GAP	NPFS	GAP	NPFS	GAP	NPFS	GAP
4	**0,0%**	-	**0,0%**	-	**0,0%**	-	**3,3%**	0,3%	**0,8%**	0,3%
5	**0,0%**	-	**6,7%**	4,2%	**0,0%**	-	**0,0%**	-	**1,7%**	4,2%
6	**0,0%**	-	**10,0%**	3,5%	**0,0%**	-	**3,3%**	1,6%	**3,3%**	2,6%
7	**0,0%**	-	**13,3%**	5,2%	**6,7%**	1,0%	**13,3%**	1,9%	**8,3%**	2,7%
8	**0,0%**	-	**16,7%**	3,1%	**0,0%**	-	**6,7%**	1,3%	**5,8%**	2,2%
9	**0,0%**	-	**16,7%**	3,4%	**10,0%**	3,0%	**16,7%**	3,6%	**10,8%**	3,3%
10	**0,0%**	-	**3,3%**	4,8%	**10,0%**	2,8%	**20,0%**	2,9%	**8,3%**	3,5%
11	**3,3%**	0,3%	**13,3%**	1,5%	**16,7%**	3,0%	**26,7%**	2,7%	**15,0%**	1,9%
12	**6,7%**	2,8%	**16,7%**	1,0%	**26,7%**	4,0%	**23,3%**	4,8%	**18,3%**	3,2%
13	**3,3%**	0,1%	**16,7%**	2,4%	**23,3%**	5,3%	**23,3%**	1,2%	**16,7%**	2,3%
14	**0,0%**	-	**20,0%**	1,2%	**30,0%**	3,6%	**26,7%**	0,9%	**19,2%**	1,9%
15	**3,3%**	0,7%	**10,0%**	1,8%	**33,3%**	3,1%	**40,0%**	1,7%	**21,7%**	1,8%
16	**10,0%**	0,4%	**13,3%**	3,2%	**26,7%**	3,7%	**23,3%**	3,5%	**18,3%**	2,7%
17	**10,0%**	0,7%	**20,0%**	2,0%	**23,3%**	2,4%	**30,0%**	3,5%	**20,8%**	2,2%
18	**6,7%**	0,7%	**16,7%**	1,0%	**20,0%**	5,1%	**33,3%**	2,6%	**19,2%**	2,3%
19	**13,3%**	0,4%	**10,0%**	1,9%	**30,0%**	5,5%	**30,0%**	2,2%	**20,8%**	2,5%
20	**0,0%**	0,8%	**20,0%**	1,8%	**43,3%**	3,0%	**26,7%**	2,7%	**22,5%**	2,1%
Col. Avg.	**3,3%**	0,8%	**13,1%**	2,6%	**17,6%**	3,5%	**20,4%**	2,3%	**13,6%**	**2,5%**

Table 2 summarizes the results of the first experimental design, in which processing times were drawn from a common distribution. The last row shows that with a larger σ the percentage of cases in which NPFS schedules are optimal increases: it goes from 3% of the scenarios for $\sigma = 0.5$ to 20 % for $\sigma = 2$. This can be explained by considering that NPFS schedules incorporate S more operations than PFS ones. Thus, with a low σ the processing times tend to be more similar and any extra operation adds a significant length to the makespan. In the extreme case in which all processing times are the same, NPFS schedules will never support the shortest makespan.

In the same row we can see that the average value of GAP also increases with σ. But the direction of change is not constant: for $\sigma = 2$, the gap is 2.3%, being lower than in the case of $\sigma = 1.5$, which is 3.5%. An interesting fact is that when $\sigma > 0.5$ the gap is larger than 2%. This is particularly relevant, considering that in other contributions to the literature (albeit non-comparable in terms of the size of the problem) as for instance Benavides & Ritt (2016), the values of the gap were around 1 %. With respect to the values of M in Table 2, we can see that for low values of M ($M \leq 10$) the percentage of cases in which a NPFS schedule is optimal never surpasses 10%. In the case of $\sigma = 0.5$, there is no scenario in which NPFS yields the optimal solution. The main reason is again the number of extra operations in NPFS schedules, which have an impact on the makespan. But considering the GAP for $M \leq 10$, we can see that it does not differ by much from the other cases. When $M > 10$, the cases in which NPFS yields the optimal solution tend to increase with M, going from 15 % for $M = 11$ to over 22% when $M = 20$. The GAP oscillates between 2% and 3%.

Table 3
Average results of the second experiment: the processing times of one of the jobs are distributed according a fixed $\sigma = 0.5$, while those of the other job have a variable σ as detailed on the top of the columns.

M	σ_2 = 1,0		σ_2 = 1,5		σ = 2		Row Avg.	
	NPFS	GAP	NPFS	GAP	NPFS	GAP	NPFS	GAP
4	**3,3%**	1,0%	**0,0%**	-	**0,0%**	-	**1,1%**	1,0%
5	**0,0%**	-	**3,3%**	1,2%	**0,0%**	-	**1,1%**	1,2%
6	**6,7%**	2,8%	**0,0%**	-	**0,0%**	-	**2,2%**	2,8%
7	**10,0%**	3,0%	**6,7%**	2,0%	**0,0%**	-	**5,6%**	2,5%
8	**3,3%**	3,7%	**10,0%**	1,5%	**3,3%**	0,2%	**5,6%**	1,8%
9	**6,7%**	1,8%	**6,7%**	4,3%	**13,3%**	2,7%	**8,9%**	3,0%
10	**10,0%**	2,4%	**6,7%**	6,3%	**10,0%**	6,6%	**8,9%**	5,1%
11	**10,0%**	3,1%	**13,3%**	2,6%	**23,3%**	2,9%	**15,6%**	2,9%
12	**16,7%**	2,0%	**10,0%**	2,8%	**16,7%**	2,4%	**14,4%**	2,4%
13	**20,0%**	3,1%	**6,7%**	1,1%	**13,3%**	2,1%	**13,3%**	2,1%
14	**16,7%**	3,5%	**6,7%**	3,0%	**20,0%**	1,5%	**14,4%**	2,7%
15	**16,7%**	2,1%	**13,3%**	1,2%	**30,0%**	1,4%	**20,0%**	1,6%
16	**16,7%**	2,4%	**16,7%**	2,1%	**16,7%**	1,1%	**16,7%**	1,9%
17	**16,7%**	2,8%	**16,7%**	2,2%	**16,7%**	1,3%	**16,7%**	2,1%
18	**10,0%**	2,6%	**16,7%**	1,5%	**20,0%**	0,9%	**15,6%**	1,7%
19	**6,7%**	5,5%	**16,7%**	3,0%	**20,0%**	1,4%	**14,4%**	3,3%
20	**16,7%**	1,4%	**23,3%**	1,9%	**26,7%**	0,6%	**22,2%**	1,3%
Col. Avg.	**11,0%**	2,7%	**10,2%**	2,4%	**13,5%**	1,9%	**11,6%**	**2,3%**

Table 3 presents the results for the second experiment. The last row indicates that, in rough terms, the percentage of scenarios in which NPFS yield the optimal solution is similar, for $\sigma = 1$, $\sigma = 1.5$ and $\sigma = 2$, namely 11 %, 10.2% and 13.5% respectively. With respect to the GAP, it decreases with increases in the dispersion of processing times: it goes down from 2.7% for $\sigma = 1$ to 1.9% for $\sigma = 2$. With respect to the impact of M, the results are similar as in the case of similar dispersion: with larger M the number of scenarios in which NPFS yields the optimal solution increases. The GAP in cases that NPFS schedules are optimal remains, again, stable for different values of M.

Table 4
Average results of the third experiment: the processing times of most of the operations are distributed according a fixed $\sigma = 0.5$ while the processing times of the middle machines operations have a variable σ as detailed on the columns

M	σ = 0,5 σ = 1		σ = 0,5 σ = 1,5		σ = 0,5 σ = 2		Row Avg.	
	NPFS	GAP	NPFS	GAP	NPFS	GAP	NPFS	GAP
10	**20,0%**	1,7%	**20,0%**	0,6%	**16,7%**	3,8%	**18,9%**	2,0%
15	**13,3%**	2,3%	**33,3%**	0,9%	**30,0%**	2,2%	**25,6%**	1,8%
20	**13,3%**	1,8%	**20,0%**	3,0%	**30,0%**	2,3%	**21,1%**	2,3%
Col. Avg.	**15,6%**	1,9%	**24,4%**	1,5%	**25,6%**	2,8%	**21,9%**	**2,1%**

Finally, Table 4 presents the results of experiments in which σ = 0.5 except for machines near M / 2, which have processing times drawn from distributions with a larger deviation. We see that with larger

dispersion in those machines, the number of cases in which NPFS yields optimality increases markedly from 15.6%, for σ = 1, to 25.6%, for σ = 2. We can also see that the GAP for σ = 2 is considerably larger than in the other cases. In turn, *M* does not show such a linear trend: the number of cases in which NPFS is optimal grows with the number of machines, until at *M* = 15 it reaches 25.6% and afterwards it drops to 21.1%, for *M* = 20. The GAP, again, is stable for different values of *M*.

In order to compare the first and the second experiments, we have to look at the lower right cells of Table 2 and Table 3. The global percentages of success of NPFS schedules are very similar: 13.6% in the first and 11.6% in the second experiment. But notice that the first experiment generated 2040 results (17 values of *M*, 4 of σ and 30 scenarios) while the second 1530 results (only 3 values of σ). If in the first experiment we drop the results for σ = 0.5, the global percentage (for the remaining 1530 results) of optimality of NPFS schedules is 17.1%, significantly higher than 11.6% of experiment 2. We can thus conclude that when the operations of *both* jobs have processing times with high dispersion (instead of just one of them) the optimality of NPFS schedules is more probable.

The comparison of the three experiments requires considering the same number of results for each of them. Table 5 puts together the last two columns of Tables (2-4) for *M* = 10, 15 and 20. We can see that, in average, the NPFS schedule is optimal at a higher rate in experiment 3: 21.9% against 17.5% for the other two. Nevertheless, the values of GAP are the lowest in the same experiment. That is, it seems that in scenarios in which processing times of a few machines are very dispersed, the non-permutation approach has higher chances of yield optimal results, albeit with a relatively small difference with respect to PFS schedules.

Table 4
Summary of the results of the three experiments for $M \in \{10, 15, 20\}$.

M	1° Experiment		2° Experiment		3° Experiment	
	NPFS	**GAP**	**NPFS**	**GAP**	**NPFS**	**GAP**
10	**8,3%**	3,5%	**8,9%**	5,1%	**18,9%**	2,0%
15	**21,7%**	1,8%	**20,0%**	1,6%	**25,6%**	1,8%
20	**22,5%**	2,1%	**22,2%**	1,3%	**21,1%**	2,3%
Avg.	**17,5%**	2,5%	**17,5%**	2,7%	**21,9%**	2,1%

6. Conclusions

In this work we studied the structure of the critical paths of both PFS and NPFS schedules. Besides introducing novel characterizations of critical PFS and NPFS paths, we presented expressions summarizing the critical paths of PFS schedules. In the case of the NPFS schedules we have shown that their makespans can be obtained in terms of a decomposition of the NPFS schedules as sequences of PFS sub-schedules. The sum of the makespans of these sub-schedules yields the makespan of the NPFS schedule. We compared the NPFS and PFS schedules, both structurally and numerically. In the former case we found cases in which NPFS schedules are dominated by PFS ones. The numerical analysis shows that with a higher dispersion of processing times (in particular on a few machines) the chances of finding optimal NPFS schedules are higher. The same happens with a larger number of machines.

Acknowledgments

The authors are grateful for partial support from the following sources: Proyecto DICYT 061817VP, Universidad de Santiago de Chile (Ó. C. Vasquez), ´ CYTED Ciencia y Tecnología para el Desarrollo [P318RT0165]; Consejo Nacional de Investigaciones Científicas y Técnicas [PIP: 11220150100777]; Universidad Nacional del Sur [PGI: 24/ZJ35] (D. Rossit, F. Tohmé, Mariano Frutos); M.D. Safe acknowledges partial support from ANPCyT Grant PICT-2017-1315 and Universidad Nacional del Sur Grants PGI 24/L103 and 24/L115

References

Akers Jr, S. B. (1956). Letter to the editor—A graphical approach to production scheduling problems. *Operations Research*, *4*(2), 244-245.

Benavides, A. J., & Ritt, M. (2016). Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops. *Computers & Operations Research*, *66*, 160-169.

Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer Science & Business Media.

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Weglarz, J. (2013). *Scheduling computer and manufacturing processes*. springer science & Business media.

Conway, R. W., Maxwell, W. L., & Miller, L. W. (2003). *Theory of scheduling*. Courier Corporation.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269-271.

Fomin, F. V., & Kratsch, D. (2010). *Exact exponential algorithms*. Springer Science & Business Media.

Garey, M. R., Johnson, D. S., & Sethi. R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*. *1*(2). 117-129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287-326.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics (NRL)*, *1*(1), 61-68.

Kelley Jr, J. E., & Walker, M. R. (1959, December). Critical-path planning and scheduling. In *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference* (pp. 160-173). ACM.

Kelley Jr, J. E. (1961). Critical-path planning and scheduling: Mathematical basis. *Operations Research*, *9*(3), 296-320.

Kis, T., & Pesch, E. (2005). A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research*, *164*(3), 592-608.

Li, S., & Tang, L. (2005). A tabu search algorithm based on new block properties and speed-up method for permutation flow-shop with finite intermediate storage. *Journal of Intelligent Manufacturing*, *16*(4-5), 463-477.

Liao, C. J., Liao, L. M., & Tseng. C. T. (2006). A performance evaluation of permutation vs. non-permutation schedules in a flowshop. *International Journal of Production Research*. *44*(20). 4297-4309.

Liao, L. M.. & Huang, C. J. (2010). Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. *Applied Mathematics and Computation*, *217*(2). 557-567.

Lin, S. W.. & Ying, K. C. (2009). Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems. *International Journal of Production Research*, *47*(5). 1411-1424.

Nagarajan, V.. & Sviridenko, M. (2009). Tight bounds for permutation flow shop scheduling. *Mathematics of Operations Research*. *34*(2). 417-427.

Nip, K., & Wang, Z. (2013, June). Combination of Two-Machine Flow Shop Scheduling and Shortest Path Problems. In *COCOON* (pp. 680-687).

Nip, K., Wang, Z., Nobibon, F. T., & Leus, R. (2015). A combination of flow shop scheduling and the shortest path problem. *Journal of Combinatorial Optimization*, *29*(1), 36-52.

Pinedo, M. L. (2002). *Scheduling: theory. algorithms. and systems*. Springer Science & Business Media.

Potts, C. N., Shmoys, D. B., & Williamson, D. P. (1991). Permutation vs. non-permutation flow shop schedules. *Operations Research Letters*. *10*(5). 281-284.

Rebaine, D. (2005). Flow shop vs. permutation shop with time delays. *Computers & Industrial Engineering*. *48*(2). 357-362.

Rossi, A., & Lanzetta, M. (2013). Scheduling flow lines with buffers by ant colony digraph. *Expert Systems with Applications*, *40*(9), 3328-3340.

Rossi, A., & Lanzetta, M. (2014). Native metaheuristics for non-permutation flowshop scheduling. *Journal of Intelligent Manufacturing*, *25*(6), 1221-1233.

Rossit, D., Tohmé, F., Frutos, M., Bard, J., & Broz, D. (2016). A non-permutation flowshop scheduling problem with lot streaming: A Mathematical model. *International Journal of Industrial Engineering Computations*, *7*(3), 507-516.

Rossit, D. A., Tohmé, F., & Frutos, M. (2018a). The non-permutation flow-shop scheduling problem: a literature review. *Omega*. 77, 143-153.

Rossit, D. A., Vásquez, Ó. C., Tohmé, F., Frutos, M., & Safe, M. D. (2018b). The dominance flow shop scheduling problem. *Electronic Notes in Discrete Mathematics*, *69*, 21-28.

Rudek, R. (2011). Computational complexity and solution algorithms for flowshop scheduling problems with the learning effect. *Computers & Industrial Engineering*, *61*(1), 20-31.

Shen, L., Gupta, J. N., & Buscher. U. (2014). Flow shop batching and scheduling with sequence-dependent setup times. *Journal of Scheduling*, *17*(4), 353-370.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European journal of operational research*, *64*(2), 278-285.

Tandon, M., Cummings, P. T., & LeVan, M. D. (1991). Flowshop sequencing with non-permutation schedules. *Computers & chemical engineering*, *15*(8), 601-607.

Vahedi-Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints. *Journal of Manufacturing Systems*, *32*(1), 167-173.

Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: A survey. *Lecture Notes in Computer Science*, *2570*(2003), 185-207.

Xiao, Y., Yuan, Y., Zhang, R. Q., & Konak. A. (2015). Non-permutation flow shop scheduling with order acceptance and weighted tardiness. *Applied Mathematics and Computation*, *270*, 312-333.

Ying, K. C., & Lin, S. W. (2007). Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *33*(7-8), 793-802.

Ying, K. C. (2008). Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. *The International Journal of Advanced Manufacturing Technology*, *38*(3-4), 348-354.

Ying, K. C.. Gupta, J. N., Lin. S. W., & Lee, Z. J. (2010). Permutation and non-permutation schedules for the flowline manufacturing cell with sequence dependent family setups. *International Journal of Production Research*, *48*(8), 2169-2184.

Ziaee, M. (2013). General flowshop scheduling problem with the sequence dependent setup times: A heuristic approach. *Information Sciences*, *251*, 126-135.