This is an Accepted Manuscript of an article published by Elsevier in Omega on June 27, 2017

available online: https://doi.org/10.1016/j.omega.2017.05.010

The non-permutation flow-shop scheduling problem: a literature review

Rossit, Daniel Alejandro[1,2], Tohmé, Fernando[2,3], Frutos, Mariano[1,4]*

[1]Department of Engineering, Universidad Nacional del Sur, Av. Alem 1253, Bahía Blanca (8000), Argentina.
[2]INMABB-UNS-CONICET, Av. Alem 1253, Bahía Blanca (8000), Argentina.
[3]Department of Economics, Universidad Nacional del Sur, San San Andrés 800, Bahía Blanca (8000), Argentina.
[4]IIESS UNS CONICET, San Andrés 800, Bahía Blanca (8000), Argentina.
**corresponding author*

Abstract

The Non-Permutation Flow-Shop scheduling problem (NPFS) is a generalization of the traditional Permutation Flow-Shop scheduling problem (PFS) that allows changes in the job order on different machines. The flexibility that NPFS provides in models for industrial applications justifies its use despite its combinatorial complexity. The literature on this problem has expanded largely in the last decade, indicating that the topic is an active research area. This review is a contribution towards the rationalization of the developments in the field, organizing them in terms of the objective functions in the different variants of the problem. A schematic presentation of both theoretical and experimental results summarizes many of the main advances in the study of NPFS. Finally, we include a bibliometric analysis, showing the most promising lines of future development.

Keywords: Non-permutation Flow-shop; Scheduling; Flow-Shop; Review

Highlights

- An exhaustive review of the Non-Permutation Flow-Shop Scheduling problem.
- A comprehensive classification in terms of objective functions and the solution methods employed in the literature.
- A compilation of problems that, modeled as PFS, do not ensure optimality.
- A revision of the main experimental results.

- A detailed description of future research lines and literature gaps.

1. Introduction

Scheduling problems of production systems have been extensively analyzed and worked out under different approaches (Błazewicz, et al (1996, [8]); Allahverdi et al (1999, [3]); Kis (2003, [37]); Allahverdi et al (2008, [4]); Kis and Kovacs (2012, [39]) and Allahverdi (2015) [5]). The results in this field have contributed to the improvement of manufacturing systems (Błazewicz, et al (2007, [11])).

Flow-shop configurations are commonplace in manufacturing settings where a set of jobs $N = \{1, 2, \ldots, n\}$ are processed by a set of machines $M = \{1, 2, \ldots, m\}$. Each job goes through the machines in the same technological order, i.e. it starts at machine 1, then goes to machine 2, … up to machine m. The decision to make is to choose the order on which the different jobs will pass through the machines. If the job sequence is the same for all the machines, the schedule is called a *permutation* and the problem of choosing the best one is known as the Permutation Flow-Shop problem (PFS). If instead the processing sequence can change from one machine to the next, the permutation condition is relaxed and the problem is known as Non-Permutation Flow-Shop (NPFS). The standard description of the NPFS problem considers the following specifications:

1. Each machine can process only one job at a time.
2. Each job j has a processing time p_{ij} on machine $i = 1, 2, \ldots, m$.
3. The capacity of intermediate buffers must be large enough to allow the reordering of the job sequence.

The standard settings of NPFS and PFS are very similar, being the third item the most relevant potential difference between them. In some cases, PFS problems assume also intermediate buffers with unlimited capacity, being so perfectly compatible with NPFS. On the other hand, in the absence of intermediate buffers, the NPFS approach is not applicable to obtain a feasible scheduling scheme (the same happens with the “no-wait” flow-shop case [48]). Besides the aforementioned three main specifications in the standard NPFS form, there are other requirements: all the jobs and machines must be available from the beginning; preemption is not allowed; machines can be idle during the planning horizon; each job can be processed by only one machine at a time; and the problem data is deterministic and known in advance. This description does not encompass the entire realm of NPFS problems, but serves as a template for them. With minor changes (such as adding or removing constraints), all the different NPFS variants can be obtained.

In the last decade, the researchers in the scheduling community have shown a growing interest in the analysis NPFS problems. Among the important issues that have since been considered, one is the detection of the manufacturing conditions for which NPFS is more promising than PFS, since the solutions that are obtained under the latter approach can be inferior to those of NPFS problems. This approach has been extensively analyzed in the literature on flow shop systems, yielding new views on the schedule of production activities. This is in particular the case of environments in which the optimizing criterion is related to due-dates. Liao et al (2006, [43]) indicates that solving flow shop problems minimizing total tardiness under the PFS approach leads to efficiency losses of around a 10% of the objective in comparison to the solutions obtained under NPFS. Moreover, Lin et al (2009, [46]) shows that in flow shop systems organized in manufacturing cells with due-date related objectives the gains in efficiency obtained with NPFS schedules are, for some cases, of 30%, while in average of around 10%. In the case of completion time-related objective functions the gains are of 5% to 6%. Some of these results were extended by Ying et al (2010, [97]), showing empirically that if set up and processing times have larger dispersion the improvements are even larger. For instance, if the range of set up times increases, the average improvement of NPFS schedules over PFS ones for completion time objectives, when the range of set up times increases, grows from 0,5% to 1,5%, reaching in some cases up to 13%. For objective functions related to delivery dates, the average improvements grow from 0.5% under PFS to 7% with NPFS, with many instances above 30% reaching even 40%.

This is extremely relevant since flow-shop settings are very common in actual industrial plants, representing nearly a quarter of manufacturing systems, assembly lines and information service facilities (Pan et al., 2011, [59]). Therefore, the possibility of improving the performance of manufacturing systems by means of a better scheduling approach can have a huge impact on a number of industry and organizations.

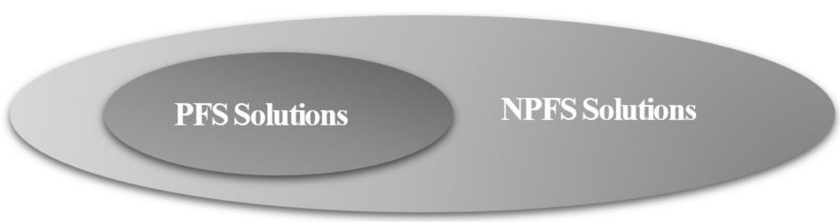

Figure 1. Solution spaces of NPFS and PFS

The main reason for the late concern with NPFS problems is their hardness: while the PFS approach searches its optimal solution among *n!* feasible schedules, being *n* the number of jobs, the NPFS approach has to consider $n!^m$ possibilities. The increase of hardware computational power in the last decade has nevertheless fueled the interest in finding efficient algorithms for NPFS problems. In fact, more than the 65% of the papers reviewed for this paper, have been published after 2006. This shows that NPFS is currently a fashionable topic in the flow-shop scheduling literature. Furthermore, the results obtained indicate that this approach has large potential benefits superseding those of the classic PFS one. Moreover, as shown in Figure 1, since the class of solutions of PFS is a subset of those of NPFS.

We conceive our survey as a contribution to the systematization of the literature on NPFS problem, highlighting important results and outlining future research lines. This review gathers, to the best of our knowledge, all the NPFS literature, describing the NPFS problems discussed there, classifying them in terms of the objective functions and commenting on the solution methods applied. The organization of the paper is as follows. Section 2 presents the classification and notation used in the paper. Once laid out the basis for the review, section 3 presents a description of the literature, classified according to the objective functions considered there. In Section 4, we present a statistical analysis of the problems and solutions methods developed in the literature, obtaining interesting bibliometric data and results. Finally, section 5, presents the conclusions of this work and an outline of promising future lines of research.

2. Non-Permutation Flow-Shop problems: classification, notation and other considerations

To represent the different NPFS variants we have to consider some modifications of the standard form presented in the previous section, namely removing or adding assumptions and constraints. To denote them we adopt the classification and nomenclature proposed by Graham et al. (1979, [29]) and implemented by Pinedo (2012, [62]). The resulting NPFS variants are characterized as a triplet $\alpha\,|\,\beta\,|\,\gamma$. The first field, α, describes the machine environment or shop configuration and contains only one entry. The β field provides details of the processing characteristics and constraints and may contain no entry at all, a single entry, or multiple entries. The γ field describes the objective function, usually in a single entry (more than one entry indicates a multi-objective case).

With regard to the α field, the possible entries could be either F (for pure flow-shop settings with m stages and only one machine or processor per stage), or J (for job-shop with m stages). Despite this potential variety, in this paper we consider only the pure flow-shop settings. The reason of this is that the other settings have been already described in the literature. For instance, the hybrid flow-shop has been reviewed by Linn et al. (1999, [49]); Ruiz et al. (2010, [81]); Ribas et al. (2010, [74]) and Li et al. (2015, [42]). Thus, for us, the only possible entry in the α field is F.

With respect to the β field, multiple entries are possible, enumerating the constraints and assumptions considered for the specific cases. The appearance of an entry implies that the corresponding condition applies. The possible entries are:

- r_j: indicates that jobs cannot start their processing before their release date. If r_j is not present in the β field, jobs can start their processing at any time. In contrast to release dates, due dates are not specified in this field. The objective function gives sufficient indication whether or not there are due dates.
- *Prmp*: means that preemption is allowed, while its absence indicates that they are not allowed.
- s_{jk}: denotes the sequence-dependent setup time of job k after finishing job j. If this setup time depends on the machine, the machine subscript i is included, i.e., s_{ijk}. If no s_{jk} appears in the β field, all setup times are supposed to be sequence independent (included in the processing times) or 0.
- *prmu*: indicates that the job ordering is the same order for every machine.
- *block*: implies that buffer capacities between machines are limited. Jobs must wait in the previous stage until sufficient space is free. This condition is not enough to prevent NPFS schedules, since the buffer capacity may be enough to reorder at least one job. This topic will be thoroughly discussed in the next section.
- *unavail*: states that machines are not available at some times.

In the case of stochastic parametrizations, we will indicate it with the same notation but in capital letters. For instance, if the release date of job j is an uncertain parameter, the entry at the β field will be denoted R_j, while the regular non-stochastic entry is r_j. This notation is adopted from Pinedo (2012, [63]). Other possible entries for β exist, but do not apply in our study, as for instance no-wait (it does not work for NPFS) and precedence (it is redundant for flow-shop settings). Nevertheless, if some other entry appears in our review, its denotation will be self-explanatory.

Let C_{ij} represent the completion time of the operation of job j on machine i, and C_{mj} the completion time on the last machine (that is, when j exits the system). The flowtime of job j is denoted by F_j, and indicates the time spent by the job in the system, which can be calculated as: $F_j = C_{mj} - r_j$. The lateness of job j is defined as L_j, and *is* $L_j = C_{mj} - d_j$. So expressed, L_j can be negative. The tardiness of job j is $T_j = \max\{ C_{mj} - d_j, 0\}$ and the earliness $E_j = \max\{ d_j - C_{mj} , 0\}$. Both of them are nonnegative by definition. If there exists a penalty for each tardy job, then the unit penalty is used,

U_j, which is 1 if $C_{mj} > d_j$ and 0 otherwise. Many objective functions associate a weight to each job, w_j. These weights gauge the importance of each job respect to the others, representing different costs, volume, priorities or other special features consider relevant by the decision-maker.

To illustrate how this notation is used, let us consider the standard version of NPFS presented in the introduction of the paper, which will be denotated as $F \mid \mid C_{max}$ (notice that the β field is empty). This means that the production setting is a flow-shop system of m machines and the optimization criterion is captured by makespan. For another example, suppose that the number of machines is limited to 10, the jobs have release dates, the setups are sequence dependent for each machine, and the optimality criterion is maximal tardiness. This problem is represented as $F \mid r_j, s_{ijk} \mid T_{max}$.

2.1. How buffers influence policies

As already mentioned, a NPFS treatment requires the existence of intermediate buffers that smooth out the production system. There exist various kinds of intermediate buffers, depending on their capacity. The most common in the literature satisfies the condition of *unlimited intermediate storage* (UIS). On the other extreme of the range of possibilities, we find the case in which no intermediate buffers exist, corresponding to the *no intermediate storage* (NIS) condition. Between them, we have the cases of finite *intermediate storage* (FIS). We have also the case in which products must go immediately from a workstation to another, the *zero wait* (ZW) case. Finally, the *mixed intermediate storage* (MIS) case obtains as a combination of two or more of the previous cases.

The UIS condition covers the cases in which the buffering capacity is at least $n - 1$, where n is the number of jobs. This ensures the absence of deadlocks in the production system, since each machine has a buffer that allows it to store all the intermediate products except the one that is being processed. Rossi and Lanzetta (2013, [75]) reduce the storing capacity of the buffer, in this case, to $n - 2$ since the previous machine can keep on hold the result of the last processing job without interrupting the flow of the rest of jobs. However, the usual minimal bound for the capacity of UIS buffers in the literature is, as said, $n - 1$.

The opposite is the case of the NIS condition. When a job finishes its process on machine i, if machine $i + 1$ is busy processing another job, the former must stay on machine i generation a deadlock in the flow of the system, since there is no buffering facility that could be used to store it. This kind of production system does not lend itself to a NPFS treatment and admits only PFS solutions. The FIS condition, in turn, allows for the use buffers able to store $|b_i|$ units[1] after machine i has finished its operation, with $|b_i|$ less than $n - 1$ units. This implies that if job j finishes on machine

[1] By a slight abuse of language, we denote with $|b_i|$ the capacity of buffer b_i.

i, and b_i situated between i and i + 1, is full while machine i + 1 is processing job k, the result of job j must wait until it finds a place in b_i obstructing machine i. b_i will be able to free space once machine i + 1 finishes job k and transfers the result to i + 2 or to buffer b_{i+1} between i + 1 and i + 2, i = 1, 2, …, m – 2.

The complexity of the problem with intermediate buffers with limited capacity is analyzed in Papadimitriou and Kanellakis (1980, [61]), showing that even with only two machines is NP-hard. If only schedules that do not generate deadlocks are considered feasible, the number of NPFS feasible schedules depends on the capacity of each b_i. To see this, consider on one hand the case in which each b_i has capacity 0, not allowing NPFS solutions, being the number of feasible schedules $n!$ (corresponding to PFS solutions), where n is the number of jobs. On the other hand, if each b_i has at least a capacity of n – 1, no deadlock can arise and thus each NPFS schedule is a feasible solution, implying that the number of feasible solutions is $n!^m$. In turn, if the capacity of each b_i strictly larger than 0 but also less than n – 1, not all NPFS schedules will be feasible since some of them will generate deadlocks. Brucker et al (2003, [16]) analyzed this case, showing that the cardinality of the set of feasible schedule Ω grows with the capacity of each buffer b_i according to the following expression:

$$\Omega = n! \prod_{i=1}^{m-1} |bi|! \,(|bi| + 1)^{n-|bi|}$$

The ZW case focuses on jobs that, after finishing on a machine i have to transfer immediately its output to machine i + 1. It is immediate that this condition can be only satisfied by PFS schedules and thus it does not allow NPFS feasible schedules. Finally, the MIS case mixes UIS and FIS buffers with instances of NIS or ZW. Thus, NPFS feasible schedules can only exist for some parts of the system where the storing policies satisfy UIS or FIS.

3. The Literature on NPFS

The notation presented above will be applied to characterize 72 papers. The resulting information is presented in Table 1 (at the end of Section 3.5), which indicates in its first column the year of the publication, in the second the reference and in the third the characterization of the problem addressed in that publication. The last column includes some comments about the publications, such as the approach used and other aspects of the paper. This table follows a similar format to the one presented in Ruiz and Vázquez-Rodríguez (2010, [81]). We encourage the reader to examine the different solution methods that have been proposed for flow shop systems: in the case of exact solutions see Kis and Pesch (2005, [38]), for the late work criterion Błażewicz, et al (2005, [10]) and for meta-heuristics with sequence-dependent setups Ruiz, et al (2005, [80]).

In order to organize the review, we will divide the papers according to the type of objective used in each work. Among the objectives we will consider are completion-time, cost and due-date. On the other hand, we devote a particular interest to makespan (by far the most popular completion-time objective) as a category in itself. Finally, we have two special "portmanteau" cases, one of the papers that consider multi-objective problems and the other covering those concerned with all other single-objective cases.

3.1. Completion-time based objective

3.1.1. Makespan

Makespan is the most frequently considered objective function. In fact, around 55% of the papers under review consider makespan as a single objective. Thus, we separate this objective from the rest of the completion-time ones. The first work dealing with a makespan NPFS problem was Janiak (1988, [36]). In that paper, the duration of each operation depends linearly on the fraction of a limited resource allotted to each machine (for instance fuel), and the decision is twofold, involving the choice of the job sequence and the allocation of the resource to the different machines. To solve the problem, a Branch & Bound procedure is applied. Potts et al. (1991, [64]) quantified for the first time the impact of enforcing permutation schedules. They found a set of instances for which the worst case of PFS makespan is $1/2\sqrt{m}$ times the NPFS makespan. Tandon et al. (1991, [90]) compared empirically PFS against NPFS schedules. For small instances, they adopted an enumerative procedure while for bigger ones they used simulated annealing. They showed that, for wider ranges of processing times and bigger instances, NPFS becomes more advantageous than PFS. Strusevich and Zwaneveld (1994, [86]) addressed two-machine cases, considering separately the setup, processing and removal times. In this case, PFS cannot ensure optimality, and in the worst case the makespan of PFS is 3/2 of the NPFS makespan. They also analyzed the two-machine case with finite buffer capacity, to show again that PFS does not ensure optimality. Both cases analyzed by Strusevich and Zwaneveld are NP-hard. Deal, et al (1994, [21]) analyze problems of petrochemical plants for which NPFS schedules are feasible, using FIS buffering. To solve the problem these authors use a heuristic method identifying critical jobs, balancing the job load among processing stations and avoiding bottlenecks. Grau et al (1996, [31]) study the scheduling of multipurpose batch plants with a finite wait inter-stage policy (after finishing the processing a job in a machine, the time that the next job can wait is restricted). To face this NPFS problem they implemented recursive procedures. Koulamas

(1998, [40]) presented a heuristic (HFC) capable of generating non-permutation schedules when it deems appropriate. This heuristic has a similar performance as the NEH heuristic (Nawaz et al 1989, [57]), with the advantage of yielding NPFS solutions while the NEH algorithm does not. Schwindt & Trautmann (2000, [84]) analyze scheduling in batch production systems seen as an instance of resource–constrained project scheduling by incorporating sequence–dependent facility setup times and finite intermediate storage constraints. They also take into consideration possible production shutdowns and time–varying work force. Jain and Meeran (2002, [35]) propose a multi-level hybrid meta-heuristic enabling an efficient interaction between strategies of intensification and diversification, based on scatter search and path relinking techniques. Liu and Ong (2002, [50]) propose three meta-heuristics for PFS and NPFS problems based on the neighborhood structure of insertions. The meta-heuristic for NPFS problems has a critical-path neighborhood structure. Méndez & Cerdá (2003, [53]) formulates a mathematical model of operational strategies changing the precedences in the production line, also assuming that decisions can be made on the use of intermediary buffers shared by several stages of the process. Pugazhendhi et al. (2003, [65]) consider the NPFS problem assuming skipping or missing operations. A heuristic procedure (called NPS) that inserts a job in the sequence whenever it improves the makespan. Brucker et al. (2003, [16]) handle the NPFS problem with limited buffer capacity, which can eventually lead to blockings (when the buffer is complete, the job must wait occupying the machine after its processing has finished). To solve the problem, they implement a Tabu Search algorithm. Aggoune (2004, [66]) addresses the NPFS problem considering availability constraints due to maintenance activities. Two types of maintenance activities are considered separately, one of a fixed type, and the other of a time-window kind. In the fixed case, the tasks must be carried out according to a fixed timetable, while in the time-window case, there exists a time interval to perform the maintenance tasks. The solution is obtained using a combination of a genetic algorithm and Tabu Search. Pugazhendhi et al. (2004, [66]) tackle the NPFS problem with missing operations and sequence-dependent setup times. The optimizing procedure consists in a new recursive formulation that gives a good permutation solution, followed by the NPS heuristic ([65]) improving the solution by yielding non-permutation schedules. This paper, also, deals with the objective function of minimizing the total weighted flow time. Rebaine (2005, [73]) studies the worst-case performance ratio between the solutions of NPFS and PFS problems with time delays. For the two-machine case, the solution of the PFS version does not ensure optimality yielding a worst-case makespan ratio of 2. But if the operation times are just of one unit of execution time, the makespan ratio is reduced to (2-(3/n+2)). For the *m*-machine case, the makespan ratio is bounded by *m*. Haq et al. (2007, [32]) address the NPFS problem with a Scatter Search algorithm. The algorithm is based on joining solutions and exploiting the adaptive memory to avoid generating or incorporating duplicate solutions at various stages of the problem. Ying and Lin (2007, [96]) present a Multi-Heuristic Desirability Ant Colony system (MHD-ACS) for NPFS problems. They show the benefits of ant colony optimization for the solution of NPFS problems. Ying (2008, [97]) proposed an iterated greedy heuristic for NPFS problems. This heuristic is compared to other simple constructive heuristics and state-of –the-art meta-heuristics. As a conclusion, the author indicates that iterated greedy methods are promising for NPFS problems. Rayward-Smith and Rebaine (2008, [72]) present two heuristics for the two-machine unit execution time operations with time delays. The heuristics are based on ordering jobs in terms of a non-increasing time delays order. Sadjadi et al. (2008, [82]) analyze three NPFS problems, two of them with makespan as the objective function and the other one with total weighted tardiness. In the makespan cases different features are considered, one of them involves including time lags while another assumes sequence-dependent setup times. Both of these cases consider missing operations. Mixed-Integer linear programming formulations are presented for both cases. Sadjadi et al. (2008, [83]) consider two NPFS problems with different objectives: one with makespan and the other with total flow time as goals. To solve this problem, they implement a two-step procedure. Initially, an Ant Colony optimization algorithm is used to obtain a good permutation solution. Then, this solution is improved by means of a local search procedure that yields a non-permutation solution. Lin and Ying (2009, [46]) present a hybrid Simulated Annealing and Tabu Search algorithm for the NPFS problem also yielding a non-permutation solution. Nagarajan and Sviridenko (2009, [58]) present a bound for the PFS and the NPFS solutions to the general case, showing that the makespan of the PFS optimal solution can be at most $2\sqrt{\min\{m,n\}}$ times the makespan of the NPFS optimal solution.

Zheng and Yamashiro (2010, [100]) propose a quantum differential evolutionary algorithm (QDEA) for the NPFS problem. The algorithm is based on running differential operations and local search over a so-called Q-bit representation. Färber et al. (2010, [26]) address a scheduling problem in which resequencing is permitted when workstations have access to intermediate or centralized resequencing buffers, although this access is restricted by the number of available buffer places and the physical size of the products. To solve this problem, the authors apply a hybrid approach, based on constraint logic programming (CLP). Brucker and Shakhlevich (2011, [17]) study the inverse scheduling version of the flow-shop problem, i.e. one in which, firstly, a job sequence is given, and then, to make it optimal, processing times are restricted as to satisfy certain boundaries. They deduce necessary and sufficient conditions for both PFS and NPFS problems. Ramezanian et al. (2011, [70]) study the NPFS problem with missing operations, solving it with a genetic algorithm and Tabu Search. Rudek (2011, [79]) prove that in the two-machine case with learning effects, PFS does not ensure optimality, and both approaches (PFS and NPFS) are NP-hard, even if the learning effect is assumed for only one of the machines (in a form of steep learning curve). Cheng et al. (2012, [18]) analyze the process of tearing-down and reconstructing buildings as a two-machine flow-shop with resource-constrained problem. The authors provide MIP problem formulations and discuss their complexity, developing polynomial algorithms for special cases. Rossi and Lanzetta (2013, [75]) address the NPFS problem with an ACO algorithm, establishing that the minimum buffer capacity to avoid blockings is (*n-2*). Rossi and Lanzetta (2013, [76]) deal with the same problem. A particular feature of the ACO algorithm is that from the beginning it explores non-permutation solutions. In [76] the authors tested the ACO algorithm on Taillard's (1993, [89]) benchmarks, but in Rossi and Lanzetta (2014, [77]) they use the benchmarks of Demirkol et al. (1998, [22]) benchmarks. For these instances, their ACO algorithm outperforms other variants also used to solve NPFS problems. Shen et al. (2014, [85]) tackle the NPFS batching problem with sequence-dependent family setup time. These authors develop a Tabu Search algorithm, including double tabu lists and multilevel diversification. The group technology assumption is relaxed, allowing the family of jobs to be split. Gharbi et al. (2014, [27]) present lower and upper bounds for several single-machine adjustment procedures. Moukrim et al. (2014, [56]) introduce a Branch & Bound algorithm for the problem described in Rebaine (2005, [73]). They present both new bounding procedures for this B&B algorithm as well as new dominance rules. Benavides et al. (2014, [13]) deal with heterogeneous NPFS problems for which two simultaneous issues need to be addressed: the assignment of workers to workstations and the scheduling problem itself. The motivation comes from cases in which workers are disabled people, and thus, their skills are not homogeneous. To solve this optimizing problem, a Scatter Search and a Path Relinking algorithm are proposed. In Nikjo and Zarook (2014, [59]) the problem analyzed is a NPFS in the context of a manufacturing cell with agreeable release dates and setup times dependent on the sequence of parts of related products. Genetic algorithms and Tabu Search yield the solutions. Zhang et al. (2014, [99]) approach the NPFS problem with periodical maintenance activities. The method used for its solution is a hybrid genetic algorithm and a heuristic based on NEH theory. Rossit et al. (2016, [78]) deals with NPFS problem under lot streaming considerations. Benavides and Ritt (2016, [15]) propose a constructive iterated local search heuristic for the NPFS problem. The algorithm is based on the observation that permutation and non-permutation schedules are similar enough as to facilitate finding a non-permutation solution after obtaining a good permutation one. Cui et al. (2016, [20]) deal with NPFs problems with availability constraints. The availability of machines depends on two kinds of extra-production tasks, one involves fixed tasks while the other refers to tasks with flexible time intervals with the continuous working time assigned to machines cannot surpass a maximum allowed time. The optimization is carried out running a hybrid incremental genetic algorithm combining local refinements and a population diversity supervision scheme.

3.1.2. Other completion-time based objectives

We will review here the literature on NPFS problems with other completion-time based objectives. In particular, we will focus on the following objective functions: total completion time, total weighted completion time, total flow time and total weighted flow time.

Rajendran and Ziegler (2001, [69]) study the NPFS problem with missing operations when the objective function is the minimization of total flow time. The authors solve it using dispatching rules combined with a heuristic rule. Pugazhendhi et al. (2004, [67]) deal with two NPFS problems with missing operations, the first one minimizing the total flow time, and the second, minimizing the total weighted flow time. They present a heuristic (NPS-set), which works by improving a permutation schedule. Färber and Coves Moreno (2006, [24]) propose a genetic algorithm for NPFS problems when intermediate buffers are not available for every station or machine, each of which is assumed to be capacitated. Färber et al. (2007, [25]) tackle a NPFS problem in which the demand is semi-dynamic and the resequencing is restricted (similarly to [24]). The objective function is total weighted completion time. The authors solve the problem by applying two approaches: the first a Constraint Logic Programming analysis and the second a genetic algorithm. Li et al. (2010, [41]) address a two-machine robotic NPFS problem with total weighted completion as the performance criterion. Robots take care of loading, unloading and translating jobs from a station to another. These robots can handle only one job at a time. Optimal solutions arise from the application of a genetic algorithm. Vahedi-Nouri et al. (2013, [91]) address the NPFS problem with learning effects and machine availability constraints under the minimization of total flow time. The authors present a MIP formulation and propose an improvement heuristic. Isenberg and Scholz-Reiter (2013, [34]) deal with a batching NPFS problem, where batches are built at each stage. This results in a stage-interdependent batching and scheduling problem. These authors consider three different objective functions: total flow time, total completion time and makespan. Vahedi-Nouri et al. (2014, [93]) present a heuristic method and a Simulated Annealing algorithm for a NPFS problem with learning effects, availability constraints and release dates. The objective function optimize is total flow time. Benavides and Ritt (2015, [14]) study the advantages of NPFS over PFS schedules. They use a two-phase heuristics and consider the case of total completion time as objective function. In the first phase, an iterated local search algorithm seeks a good permutation solution, and in the second phase, an effective insertion neighborhood improves that solution by exploring close non-permutation solutions. Henneberg and Neufeld (2016, [33]) study a NPFS with missing operations when the objective is total completion time. They solve it with a modification of the NPS-set heuristic presented in [22], based on a two-phase version of Simulated Annealing.

3.2. Due-date based objectives

Here we will focus on papers in which the objective functions represent a due-date concept. These problems are known for being computationally hard, being even “binary NP-hard” in two-machine cases (Błazewicz, et al 2005, [10]). Nevertheless, these problems have been extensively studied in the PFS setting (Błażewicz , et al (2008, [12]); Pesch and Sterna (2009, [62]))

The objective functions that will be contemplated in this section are: maximum tardiness, total tardiness and total weighted tardiness.

Swaminathan et al. (2007, [88]) study the impact of the enforcement permutation condition on the general flow shop (non-permutation) problem. The goal analyzed is total weighted tardiness. To obtain the solution they use three approaches: pure permutation, shift-based and pure dispatching. The latter is the one able to yield non-permutation schedules. Their results show that PFS provides an inefficient approach to this problem. Swaminathan et al. (2004, [87]) study the same problem in a simplified version. Liao and Huang (2010, [44]) study the NPFS problem with total tardiness as a goal, presenting and evaluating three different MIP formulations. Then, they present also two Tabu Search algorithms. The comparison of NPFS to PFS indicates that NPFS is much more suitable for these types of problems. Ziaee (2013, [101]) addresses the NPFS problem with sequence dependent setup times with the minimization of total weighted tardiness as objective. This author proposes a two-phase heuristic with the usual pattern. Namely, the first phase looks for a good permutation solution, and second one, improves it through a non-permutation local search. Xiao et al. (2015, [94]) analyze flow-shop scheduling with order acceptance under weighted tardiness. The authors present two different formulations of the problem. The first is a MIP formulation, which CPLEX can solve for small instances. The second one, is a NIP (non-linear integer programming) formulation that can be solved, in particular its medium and large size instances, by a two-phase genetic algorithm.

3.3. Experimental mono-objective studies

In this subsection, we present a group of papers comparing the quality of the solutions of the PFS and NPFS problems in experimental analyses. These papers consider different given mono-objective manufacturing settings, in order to assess the extra computational effort required by the NPFS problems. The validity of the comparisons of these papers comes from the fact that the problems are tested under the same parametrization and same instances while the solutions are obtained running the same algorithms. In this way, these papers provide valuable experimental insights to the non-permutation literature. The objectives analyzed in all the cases are the six more common ones used in scheduling: three are completion-time based criteria (makespan, total completion time and total weighted completion time), and the other three are due-date based criteria (maximum tardiness, total tardiness and total weighted tardiness).

Liao et al. (2006, [43]) were the first to carry out this type of research. They tested a classic flow-shop system under six objective functions. Their results indicate that, in general, NPFS schedules improve very little over the PFS ones the value of completion-time based objectives. However, for due-date based criteria the improvement is significant, especially for problems with more than thirty jobs. They used as optimization tools a Genetic Algorithm and a Tabu Search algorithm. Lin et al. (2009, [47]) presents a similar study, with the same objective functions but for a flow line manufacturing cell with a sequence-dependent family of setups. Again, the conclusion for completion-time based objectives is that non-permutation and permutation schedules have a similar performance, being non-permutation a little better. But for due-date based objectives, non-permutation schedules clearly outperform permutation ones. The authors solve the problems using a Genetic Algorithm, Simulated Annealing and Tabu Search. The Simulated Annealing procedure outperforms the other two meta-heuristics. Ying et al. (2010) [98] revisit [47], testing different setup ranges, concluding that, for larger setup ranges NPFS overtakes PFS for most of the cases yielding larger improvements. They find that NPFS performs better, in general, under the six objective functions, but for due-date based ones, its performance is much better than that of PFS. In this case, all the solutions are found running a Simulated Annealing algorithm.

3.4. Multi-objective versions

A promising area of study for non-permutation scheduling involves the optimization of several objectives, mainly because the non-permutation case allows for a dearth of new solutions that do not arise in the permutation setting. The papers that analyze multiple-objective instances of the NPFS problems will be reviewed next.

Mehravaran and Logendran (2012, [51]) were the first to study multi-objective problems under non-permutation schemes. They consider a flow-shop setting with sequence-dependent setup times assuming machine availability constraints, job releasing and missing operations. They use a bi-objective function. The goal is the minimization of the normalized sum of weighted completion time and weighted tardiness. The authors present a MIP formulation and a Tabu Search algorithm. Mehravaran and Logendran (2013, [52]) address the NPFS problem considering dual resources: machines and labor. The goal is the minimization of the total weighted completion time and the total weighted tardiness. As in [51] they use a weighted sum combining the two objectives. The specification of the problem includes different skill levels, sequence-dependent setups, machine availability constraints and job release dates. A two-layered procedure yields the solution. The outer layer solves the traditional flow-shop problem (considering only job sequencing), and the inner layer, finds an assignation of jobs to labor in agreement to the machine schedule. Three different search algorithms are developed. These authors, the first ones to investigate flow-shop scheduling with two resources problem, emphasize on the superiority of non-permutation schedules over permutation ones. Rahmani et al. (2014, [68]) study a stochastic NPFS problem. Processing times and release date are stochastic parameters that have a normal distribution. Three different objectives are minimized: makespan, total flow time and tardiness. To deal with uncertainty they apply both a chance constrained programming and a fuzzy goal programming approach. They also adapt a genetic algorithm to handle large-size problem. Amirian and Sahraeian (2015, [6]) analyze a NPFS problem minimizing simultaneously the makespan, the sum of flow time and maximum tardiness. The setting includes release dates, past sequence-dependent set-up times, learning effects and machine availability constraints. The authors use, as solution methods, Augmented ε-constraint and a heuristic based on it.

3.5. Economic objective functions

In this section, we review works that evaluate objective functions from an economic point of view, trying either to minimize operation costs or to maximize profits. In particular, we review papers that study NPFS problems in which the cost is the objective function. In these five contributions, the specification of which cost has to be minimized varies.

Grau et al. (1995, [30]) study a NPFS problem seeking to minimize the product changeover cost of the production plan. This cost is incurred each time the production is set to produce a different product. The authors develop a Branch and Bound procedure to solve the problem. Doganis et al (2005, [23]) analyzes flow shop lubricant production processes. A MILP model is used to generate schedules that are potentially NPFS, but not allowing Schedule changes at all stages since between some of them buffering is of NIS type. The objective is the maximization of the income accrued by the firm. Liberopoulos et al (2010, [45]) study problems of production plants of PET resins with intermediate storage facilities specific to each product. The objective is the minimization of costs of set up of intermediate buffers, in order to adapt products to alternative buffers, a costly activity, without hampering the operational capacity of the system. Mohammadi et al. (2010, [55]) address both the lot sizing and the scheduling problem in a NPFS system. They develop a MIP formulation for the problem and present five MIP-based heuristics to minimize setup, storage and production costs. Some of these heuristics are only capable of solving the PFS version of the problem. Vahedi-

Nouri et al. (2013, [92]) analyze a NPFS problem with learning effects and flexible maintenance activities. The objective is the minimization of the sum of tardiness and maintenance costs. The authors develop a hybrid of a Firefly algorithm and Simulated Annealing to solve a MIP formulation of the problem. Ramezanian and Saidi-Mehrabad (2013, [71]) investigate the lot sizing and scheduling flow-shop problem, considering sequence-dependent setups, capacity constraints, uncertain processing times and uncertain multiproduct and multi-period demand. A MIP model joint with a big bucket time approach represents the problem. Two MIP-based heuristics with a rolling horizon framework are applied. The authors also develop a hybrid meta-heuristic based on a combination of Simulated Annealing, a Firefly algorithm and an ad-hoc heuristic for scheduling. Babaei et al. (2014, [7]) also analyze the lot sizing and scheduling problem under slightly different constraints, namely sequence-dependent setups, setup carryover and backlogging. They propose a MIP formulation solved by the application of a genetic algorithm.

T**able 1.**
Summary of the reviewed literature
References: for the β field: *rc*: resource constrained, *skip*: skipping operations, *avail*: machine availability conditions, *fmls*: family group products, *learn*: learning effect, *hr*: heterogeneous resources, *rp*: relocation, *dr*: dual resources. A β entry in capital letters means a stochastic parameter. In the Comments column, B&B: Branch and Bound, SA: Simulated Annealing, MPF: Mathematical Programming Formulation, SS: Scatter Search, PR: Path Relinking, TS: Tabu Search, OM: Other Metaheuristics, GA: Genetic Algorithm, ACO: Ant Colony Optimization, IG: Iterated Greedy, CLP: Constraint Logic Programming, CCP: Chance Constrained Programming, FGP: Fuzzy Goal Programming.

Reference	**Problem**	**Comments**
Janiak (1988) [36]	$F\|rc\|C_{max}$	B&B procedure
Potts, et al. (1991) [64]	$F\|\|C_{max}$	bound between NPFS C_{max} and PFS C_{max} for special instances
Tandon, et al. (1991) [90]	$F\|\|C_{max}$	Enumerative for small instances and SA big instances
Strusevich, et al (1994) [86]	$F2\|s_{ijk}, removal\ times\|C_{max}$	PFS is not optimal and the problem is NP-hard
	$F2\|block\|C_{max}$	PFS is not optimal and the problem is NP-hard
Deal et al 1994 [21]	$F\|b_i\|C_{max}$	heuristic for balancing resources usage
Grau, et al. (1995) [30]	$F\|batch\|Costs$	B&B procedure
Grau, et al. (1996) [31]	$F\|batch, finite\ wait\|C_{max}$	tailored recursive procedure
Koulamas (1998) [40]	$F\|\|C_{max}$	HFC heuristic
Schwindt et al (2000) [84]	$F\|time\ lags\|C_{max}$	MPF
Rajendran, et al (2001) [69]	$F\|skip\|\Sigma C_j$	dispatching rules & heuristic
Jain, et al (2002) [35]	$F\|\|C_{max}$	Meta-heuristic, based on SS and PR, and TS
Liu, et al (2002) [50]	$F\|\|C_{max}$	OM

Pugazhendhi, et al. (2003) [65]	$F \mid skip \mid C_{max}$	heuristic
Brucker, et al. (2003) [16]	$F \mid block \mid C_{max}$	TS
Méndez et al (2003) [53]	$F \mid b_i \mid C_{max}$	MPF
Aggoune (2004) [1]	$F \mid avail \mid C_{max}$	GA and TS
Pugazhendhi, et al. (2004) [66]	$F \mid skip \mid \gamma$	$\gamma \in \{\Sigma\, w_j F_j, \Sigma\, F_j\}$ Heuristic: NPS set
Pugazhendhi, et al. (2004) [67]	$F \mid skip, s_{ijk} \mid \gamma$	$\gamma \in \{\Sigma\, w_j F_j, C_{max}\}$ Tailored heuristic and NPS-set
Swaminathan, et al. (2004) [87]	$F \mid stochastic \mid Costs$	GA and ATC heuristic
Doganis et al (2005) [23]	$F \mid b_i \mid Revenue$	MPF
Rebaine (2005) [76]	$F \mid time\ delays \mid C_{max}$	NP-hard, for 2 machines PFS not optimal
Liao, et al. (2006) [43]	$F \mid \mid \gamma$	$\gamma \in \{C_{max}, \Sigma\, C_j, \Sigma\, w_j C_j, T_{max}, \Sigma\, T_j, \Sigma\, w_j T_j\}$ TS and GA, compares all the six objective functions
Färber, et al (2006) [24]	$F \mid block \mid \Sigma w_j C_j$	GA
Haq, et al. (2007) [32]	$F \mid \mid C_{max}$	SS
Ying, et al (2007) [96]	$F \mid \mid C_{max}$	ACO
Färber, et al. (2007) [25]	$F \mid block \mid \Sigma w_j C_j$	GA and CLP
Swaminathan, et al. (2007) [88]	$F \mid \mid \Sigma\, w_j T_j$	ATC heuristics and GA
Ying (2008) [97]	$F \mid \mid C_{max}$	IG
Rayward-Smith, et al (2008) [72]	$F2 \mid p_{ij}=p, time\ delays \mid C_{max}$	heuristic - (uet: unit execution time)
Sadjadi, et al. (2008) [82]	$F \mid \mid \Sigma\, w_j T_j$ $F \mid time\ lags \mid C_{max}$ $F \mid s_{ijk} \mid C_{max}$	MPF
Sadjadi (2008) [83]	$F \mid \mid \gamma$	$\gamma \in \{\Sigma\, F_j, C_{max}\}$ ACO and local search
Lin, et al. (2009) [47]	$F \mid fmls, s_{ijk} \mid \gamma$	$\gamma \in \{C_{max}, \Sigma\, C_j, \Sigma\, w_j C_j, T_{max}, \Sigma\, T_j, \Sigma\, w_j T_j\}$ SA, TS and GA
Lin, et al (2009) [46]	$F \mid \mid C_{max}$	SA and TS
Nagarajan, et al (2009) [58]	$F \mid \mid C_{max}$	Comparison of PFS and NPFS makespan for the general case
Ying, et al. (2010) [98]	$F \mid fmls, setup \mid \gamma$	$\gamma \in \{C_{max}, \Sigma\, C_j, \Sigma\, w_j C_j, T_{max}, \Sigma\, T_j, \Sigma\, w_j T_j\}$ SA, setup depends on the family sequence
Liao, et al. (2010) [44]	$F \mid \mid \Sigma\, T_j$	TS
Li, et al. (2010) [41]	$F2 \mid block \mid \Sigma w_j C_j$	GA

Liberopoulos et al (2010) [45]	$F\mid b_i \mid Costs$	MPF
Mohammadi, et al. (2010) [55]	$F\mid s_{ijk} \mid Costs$	MPF based heuristic
Zheng, et al (2010) [100]	$F\mid\mid C_{max}$	Quantum Differential Evolutionary Algorithm (QDEA)
Farber, et al. (2010) [26]	$F\mid\mid C_{max}$	hybrid CLP and GA
Brucker, et al (2011) [17]	*inverse scheduling* - C_{max}	sufficient conditions for optimal sequence
Ramezanian, et al. (2011) [70]	$F\mid skip \mid C_{max}$	GA and TS
Rudek (2011) [79]	$F2\mid learn \mid C_{max}$	NEH-based heuristic
Mehravaran, et al (2012) [51]	$F\mid\mid \Sigma w_j C_j \ \& \ \Sigma \ w_j T_j$	TS with progressive perturbation
Cheng, et al. (2012) [18]	$F2\mid rp \mid C_{max}$	complexity analysis, is NP-hard
Vahedi-Nouri, et al. (2013) [91]	$F\mid learn, avail \mid \Sigma \ F_j$	heuristic: VFR
Vahedi-Nouri, et al. (2013) [92]	$F\mid learn, avail \mid Costs$	hybrid firefly-SA
Ziaee (2013) [101]	$F\mid s_{ijk} \mid \Sigma \ w_j T_j$	local search heuristic
Isenberg, et al (2013) [34]	$F\mid batch, fmls, r_j \mid \gamma$	$\gamma \ \in \{\Sigma \ F_j, \Sigma \ C_j, C_{max}\}$ MPF
Mehravaran, et al (2013) [52]	$F\mid skip, dr, s_{ijk}, avail, r_j \mid \Sigma w_j C_j \ \& \ \Sigma \ w_j T_j$	OM
Rossi, et al (2013) [41]	$F\mid\mid C_{max}$	ACO
Rossi, et al (2013) [40]	$F\mid bi= n\text{-}2 \mid C_{max}$	ACO
Ramezanian, et al (2013) [71]	$F\mid s_{ijk}, P_{ij} \mid Costs$	MPF-Heuristics and OM, uncertain demands
Shen, et al. (2014) [85]	$F\mid batch, setup \mid C_{max}$	TS
Gharbi, et al. (2014) [27]	$Fm\mid\mid C_{max}$	bounding procedures
Moukrim, et al. (2014) [56]	$F2\mid uet, time \ delays \mid C_{max}$	B&B
Rossi, et al (2014) [77]	$F\mid\mid C_{max}$	ACO
Benavides, et al. (2014) [13]	$F\mid hr \mid C_{max}$	Heuristic: SS and PR
Nikjo, et al (2014) [59]	$Fm\mid s_{ijk}, r_j \mid C_{max}$	GA and TS
Vahedi-Nouri, et al. (2014) [93]	$F\mid learn, avail, r_j \mid \Sigma \ F_j$	Heuristic and SA
Babaei, et al. (2014) [7]	$F\mid backlog \mid Costs$	GA
Zhang, et al. (2014) [99]	$F\mid setup, avail \mid C_{max}$	ACO
Rahmani et al. (2014) [68]	$F\mid R_j, P_{ij} \mid C_{max} \ \& \ \Sigma \ F_j \ \& \ \Sigma \ T_j$	CCP and FGP
Xiao, et al. (2015) [94]	$F\mid OA = order \ acceptance \mid \Sigma \ w_j T_j$	TS-GA
Amirian, et al (2015) [6]	$F\mid learn, s_{ijk} \mid C_{max} \ \& \ \Sigma \ F_j \ \& \ T_{max}$	Augmented ε-constraint, heuristic

Benavides, et al (2015) [14]	$F \mid \mid \Sigma C_j$	IG
Benavides, et al (2016) [15]	$F \mid \mid C_{max}$	OM
Cui, et al. (2016) [20]	$F \mid avail \mid C_{max}$	OM
Henneberg, et al (2016) [33]	$F \mid skip \mid \Sigma F_j$	SA
Rossit, et al. (2016) [78]	$F \mid lot\text{-}streaming \mid C_{max}$	MPF

4. A quantitative analysis of the literature

This review has analyzed 72 papers, representing, as far as we know, the whole NPFS literature (not including Hybrid Flow-Shop variants). Our analysis follows closely other reviews, as for instance Yenisey and Yagmahan (2014, [95]) on multi-objective flow-shop formulations and Ruiz and Vásquez-Rodríguez (2010, [81]) on hybrid flow-shop problems.

A remarkable aspect of this scheduling literature is that more than the 65% of the papers have been published after 2007. This is can be seen in Figure 2, in which for clarity papers are grouped in terms of their publication in five-year periods. Given the clear trend to an increasing number of publications, while still low compared to those devoted to other well-developed scheduling issues, we can infer that NPFS is a promising area for further developments.

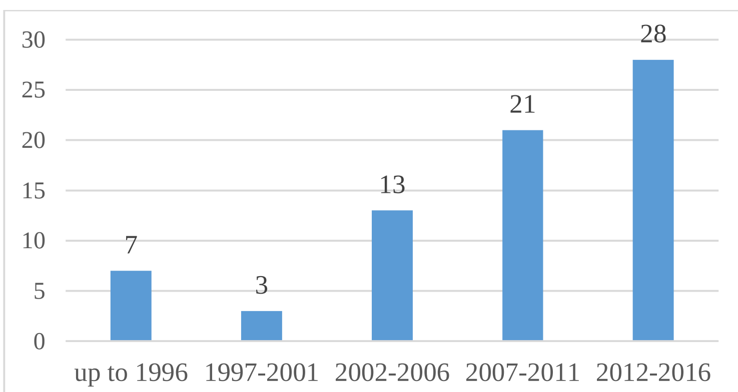

Figure 2. Number of papers published in five-year periods.

Figure 3 shows the different NPFS problems that have been analyzed in the literature, indicating the proportion of papers devoted to each kind of objective function. As was already mentioned, completion-time based are by far the most frequent objectives functions: 73% of the papers focus on them. A special case of completion-time objective is makespan, covered by 56% of the papers. Other kinds of completion-time objectives are analyzed in 17% of the publications. This is not surprising, giving the primacy of makespan over other objective functions in the literature on scheduling, as indicated in [81]. The other types of objectives functions are considered in the remaining 27% of the literature. From them, due-date based objectives functions represents only the 8% of the publications, indicating that these important objective functions are under-represented, requiring further and deeper attention. This has been emphasized in particular in [43], [47] and [98].

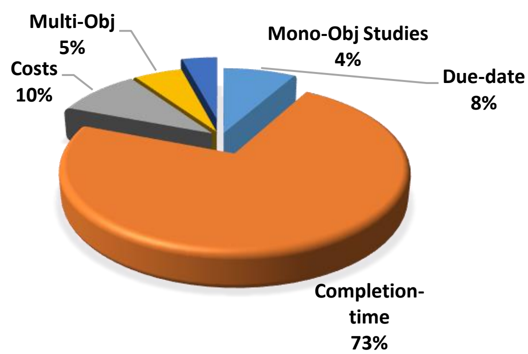

Figure 3. Distribution of objective functions considered in the literature.

The distribution of the different optimization techniques applied in the literature is presented in Figure 4. This shows that in general, exact approaches (mathematical programming and Branch and Bound) are not frequently applied, representing only 22% of the literature. In contrast, heuristics are used in 28% of the publications. Particular cases of meta-heuristic, Simulated Annealing, Tabu Search, Genetic Algorithms and Ant Colony Optimization algorithms are the most frequently applied methods of solution.

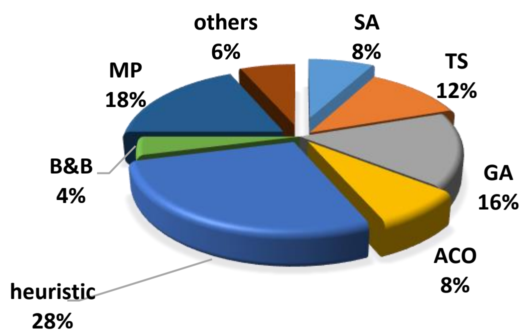

Figure 4. Distribution of optimization tools used. ACO: Ant Colony Optimization, GA: Genetic Algorithm, TS: Tabu Search, SA: Simulated Annealing, MP: Mathematical Programming, B&B: Branch and Bound.

To conclude, we can point out that there does not exist yet a consensus on the state-of-the art optimization methods for NPFS methods. We can state that exact methods seem not to be (currently) the most adequate for the solution of problems of intermediate and large size, while heuristic and meta-heuristic methods have shown to be able to yield solutions for them of good and very good quality. The downside of this is that heuristic methods are not yet able to handle general cases. On the other hand, among meta-heuristic methods, those based on Tabu Search yield better results than others to which they have been compared. Those comparisons, it must be noted, are not exhaustive and thus Tabu Search cannot be deemed yet as the best possible approach to solving NPFS problems. The natural similarities between NPFS and PFS problems have led some authors ([16]) to develop sequential improvement procedures that start by solving, in a standard way, the PFS problem. The result of such procedures is, at the very least, a very good PFS solution but sometimes yielding a NPFS one. On the other hand, Rossi and Lanzetta (2014, [77]) applied meta-heuristics (ACO) to NPFS problems just from the start, instead of finding a previous PFS solution. This allows to search directly the space of NPFS solutions. The proviso is that this approach is more adequate in the cases in which the optimal NPFS and PFS solutions differ markedly. When those solutions are rather similar, starting from PFS solutions seems a better approach to reach the optimal NPFS ones. Both approaches profit form the flow shop structure, in which the sequence is the same for all jobs.

4.1. Bibliometric analysis

Also is of interest to provide some bibliometric information about the literature on NPFS. We follow the approach of other reviews, such as Aguezzoul (2014, [2]), Merigó et al (2016, [54]) and Gorman (2016 [28]), who showed that bibliometric information can be very useful for the evaluation of the research on a new topic. The relevant information includes the list of journals were papers on the topic have been published, the frequency of publication and their impact. On the latter, [28] centers its attention in the number of citations reported by Google Scholar at the time the article was retrieved. This means, in our case, August 2016.

Table 2.

List of journals that have published two or more articles on NPFS. Note: the percentage is over the total of papers reviewed.

Publication name	No. of Papers	Percentage
International Journal of Production Research	8	11%
Inter. Jour. of Advanced Manufacturing Technology	7	10%
Computers & Operations Research	6	8%
Proceedings	6	8%
Computers & Chemical Engineering	5	7%
European Journal of Operational Research	4	5%
Computers & Industrial Engineering	3	4%
Journal of Scheduling	3	4%

OR-Spectrum	2	3%
Applied Mathematics and Computation	2	3%
Expert Systems with Applications	2	3%
Journal of Applied Sciences	2	3%

Table 2 is the list of all the journals that have published two or more papers reviewed in this work. We can see that the International Journal of Production Research has been the outlet for 11% of all the papers in the field. It is closely followed by the International Journal of Advanced Manufacturing Technology and Computers & Operations Research, that have published 7 and 6 of the papers, respectively. With respect to conference proceedings, we only consider those indexed in Scopus and Google Scholar and are written, at least its abstract, in English. The journals listed in Table 2 have published 68% of the papers on NPFS reviewed here.

Journals other than those listed in Table 2 that have published at least one article on NPFS, are Information Sciences, Journal of Manufacturing Systems, International Journal of Production Economics and Applied Mathematical Modelling.

The impact of the work on NPFS is assessed in terms of the number of citations reported by Google Scholar. Table 3 presents this information. We can see there the high impact of these articles, totaling more than 1,400 citations. That means, in average, 20 citations per NPFS article while the most cited one is Koulamas (1998 [40]) with 138 cites. On the other hand, we have to note that more than half of the papers, 37 of them, have 10 or more cites.

Table 3.
Citations of NPFS papers drawn from Google Scholar, August 2016.

Bibliometric analysis	
Numbers of total cites of NPFS papers	1452
Average number of cites per paper	20
Most cited paper (Koulamas 1998 [40]*)*	138
Papers with ≥10 cites	37 (50%)

4.2. Special cases

Since NPFS is far from being an extensively researched topic, we collect some important results that may serve as guidelines for beginners or as a state-of-the-art reference for advanced researchers or practitioners in the field. The first point to make is that NPFS schemes must yield the same or better results than PFS ones for the same problem instance since the former includes all the solutions of the latter and more. On the other hand, a highly relevant topic is the extra computational effort required to solve NPFS problems in comparison to PFS problems. The oldest result in this respect was presented by Conway et al. (1967, [19]) showing that, for the general flow-shop setting (non-permutation for us) and makespan as objective function, the schedule on the first and the second machine can be the same without hampering the optimal solution. The same is true for the last and the second to last machines. Thus, for the case of $F_3||C_{max}$, PFS is optimal. This result is clearly proven in [27]. In consequence, the NPFS approach becomes beneficial for systems with more than three machines. Newer results allow refining this analysis. In table 4, we highlight some of these results. The first row presents the bound on the worst case if the problem is solved by a PFS scheme. The next rows indicate special cases for which PFS cannot ensure optimality, even in the two-machine case, because some of the conditions of [19] do not apply.

Table 4.

Special Non-permutation results, considering makespan as objective.

Problem	Comments	Source
$F\|\|C_{max}$ vs $F\|prmu\|C_{max}$	PFS makespan worst case is: $2\sqrt{\min\{m,n\}}$ times NPFS makespan.	[58]
$F_2\|removal\ times\|C_{max}$	PFS approach does not ensure optimality. PFS makespan worst case is: 3/2 times NPFS makespan.	[86]
$F_2\|block\|C_{max}$	PFS approach does not ensure optimality.	[86]
$F_2\|time\ delays\|C_{max}$	PFS does not ensure optimality. PFS makespan worst case is: 2 times NPFS makespan.	[73]
$F_2\|learning\ effect\|C_{max}$	PFS does not ensure optimality.	[79]

Other relevant experimental results described recently are:

- For a wider range of processing times, the chances that NPFS schemes outperform PFS schedules increase [90].
- In general, environments in which the objective functions are due-date based will benefit more of the NPFS approach than environments in which they are based on completion-time [43], [98] and [94].
- For a wider range of setup times it is more likely that the NPFS approach outperforms the PFS approach [98] and [85]
- For simple flow-shop, the makespan is 2-3% better in the NPFS case [43] and [16].

5. Conclusions and directions for future research

In this paper, we have reviewed 72 articles on NPFS. We have classified these papers according to the variants of the problem considered in them, including the assumptions, constraints, objective functions and solution methods applied by the authors. We think this work may be helpful to other researchers in the field as well as a starting point for new research efforts.

The papers have been analyzed based on the type of objective function considered. Completion-time based criteria are the most frequent among the NPFS problems. Within this group, makespan is the most intensively studied (more than half of the papers have makespan as objective function). The other optimization criteria (due-date based and costs) and multi-objective approaches are covered in a quarter of all the publications. It is clear that these approaches are underrepresented in the literature. A conclusion from this review is that NPFS papers have, in average, 19 citations with more than half of them having been cited over 10 times.

Besides these conclusions, we present also a compendium of some theoretical and experimental results. On the theoretical aspect, we mentioned the problems for which the PFS approach does not ensure optimality, even in two-machine cases. That is, problems for which the NPFS approach becomes necessary to obtain high quality solutions. We also present a concise list of experimental results on the comparison of NPFS against PFS.

The NPFS problem is a recent and under-developed research topic (compared to traditional scheduling problems), and thus a promising area for further developments. The review allows us to suggest that the following are relevant inquiry issues. (1) NPFS problems with due-date based objective functions. (2) NPFS problems with three or more objectives. (3) real world case studies, comparing the costs of using NPFS and PFS approaches. (4) Scheduling under uncertainty is an interesting problem for which rescheduling could help to improve solutions. (5) The implementation of new meta-heuristics to address complex NPFS systems.

Acknowledgment

Thanks are due to the anonymous referees for their comments and criticisms that helped us to improve this paper.

References

[1] Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operational Research*, *153*(3), 534-543.

[2] Aguezzoul, A. (2014). Third-party logistics selection problem: A literature review on criteria and methods. *Omega*, *49*, 69-78.

[3] Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2), 219-239.

[4] Allahverdi, A., Ng, C. T., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3), 985-1032.

[5] Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2), 345-378.

[6] Amirian, H., & Sahraeian, R. (2015). Augmented ε-constraint method in multi-objective flowshop problem with past sequence set-up times and a modified learning effect. *International Journal of Production Research*, *53*(19), 5962-5976.

[7] Babaei, M., Mohammadi, M., & Ghomi, S. F. (2014). A genetic algorithm for the simultaneous lot sizing and scheduling problem in capacitated flow shop with complex setups and backlogging. *The International Journal of Advanced Manufacturing Technology*, *70*(1-4), 125-134.

[8] Błażewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1), 1-33.

[9] Błażewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005). A comparison of solution procedures for two-machine flow shop scheduling with late work criterion. *Computers & Industrial Engineering*, 49(4), 611-624.

[10] Błażewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005). The two-machine flow-shop problem with weighted late work criterion and common due date. *European Journal of Operational Research*, 165(2), 408-415.

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

[12] Błażewicz, J., Pesch, E., Sterna, M., & Werner, F. (2008). Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date. *Computers & Operations Research*, 35(2), 574-599.

[13] Benavides, A. J., Ritt, M., & Miralles, C. (2014). Flow shop scheduling with heterogeneous workers. *European Journal of Operational Research*, *237*(2), 713-720.

[14] Benavides, A. J., & Ritt, M. (2015). Iterated Local Search Heuristics for Minimizing Total Completion Time in Permutation and Non-permutation Flow Shops. In: *ICAPS* (pp. 34-41).

[15] 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.

[16] Brucker, P., Heitmann, S., & Hurink, J. (2003). Flow-shop problems with intermediate buffers. *OR Spectrum*, *25*(4), 549-574.

[17] Brucker, P., & Shakhlevich, N. V. (2011). Inverse scheduling: two-machine flow-shop problem. *Journal of Scheduling*, *14*(3), 239-256.

[18] Cheng, T. E., Lin, B. M., & Huang, H. L. (2012). Resource-constrained flowshop scheduling with separate resource recycling operations. *Computers & Operations Research*, *39*(6), 1206-1212.

[19] Conway, R. W., Maxwell, W. L., & Miller, L. W. (1967). *Theory of scheduling*. Courier Corporation

[20] Cui, W. W., Lu, Z., Zhou, B., Li, C., & Han, X. (2016). A hybrid genetic algorithm for non-permutation flow shop scheduling problems with unavailability constraints. *International Journal of Computer Integrated Manufacturing*, 1-18.

[21] Deal, D. E., Yang, T., & Hallquist, S. (1994). Job scheduling in petrochemical production: two-stage processing with finite intermediate storage. *Computers & Chemical Engineering*, 18(4), 333-344.

[22] Demirkol, E., Mehta, S., & Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, *109*(1), 137-141.

[23] Doganis, P., Sarimveis, H., Bafas, G., & Koufos, D. (2005). An MILP model for optimal scheduling of the lubricant production plant. *Chemical Engineering Communications*, 192(8), 1067-1084.

[24] Färber, G., & Moreno, A. M. C. (2006). Performance study of a genetic algorithm for sequencing in mixed model non-permutation flowshops using constrained buffers. In: *International Conference on Computational Science and Its Applications* (pp. 638-648). Springer Berlin Heidelberg.

[25] Färber, G., Salhi, S., & Moreno, A. M. C. (2007). Semi-dynamic Demand in a Non-permutation Flowshop with Constrained Resequencing Buffers. In: *International Conference on Large-Scale Scientific Computing* (pp. 536-544). Springer Berlin Heidelberg.

[26] Farber, G., Coves Moreno, A. M., & Salhi, S. (2010). Performance evaluation of hybrid-CLP vs. GA: non-permutation flowshop with constrained resequencing buffers. *International Journal of Manufacturing Technology and Management*,*20*(1-4), 242-258.

[27] Gharbi, A., Labidi, M., & Louly, M. A. (2014). The Nonpermutation Flowshop Scheduling Problem: Adjustment and Bounding Procedures. *Journal of Applied Mathematics*, vol. 2014, Article ID 273567, 14 pages, 2014. doi:10.1155/2014/273567

[28] Gorman, M. F. (2016). A "Metasurvey" analysis in Operations Research and Management Science: A survey of literature reviews. *Surveys in Operations Research and Management Science*, *21*(1), 18-28.

[29] 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.

[30] Grau, R., Espuña, A., & Puigjaner, L. (1995). Environmental considerations in batch production scheduling. *Computers & Chemical Engineering*, *19*, 651-656.

[31] Grau, R., Espuña, A., & Puigjaner, L. (1996). Completion times in multipurpose batch plants with set-up, transfer and clean-up times. *Computers & Chemical Engineering*, *20*, S1143-S1148.

[32] Haq, A. N., Saravanan, M., Vivekraj, A. R., & Prasad, T. (2007). A scatter search approach for general flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *31*(7-8), 731-736.

[33] Henneberg, M., & Neufeld, J. S. (2016). A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations. *International Journal of Production Research*, *54*(12), 3534-3550.

[34] Isenberg, M. A., & Scholz-Reiter, B. (2013). The Multiple Batch Processing Machine Problem with Stage Specific Incompatible Job Families. In: *Dynamics in Logistics* (pp. 113-124). Springer Berlin Heidelberg.

[35] Jain, A. S., & Meeran, S. (2002). A multi-level hybrid framework applied to the general flow-shop scheduling problem. *Computers & Operations Research*, *29*(13), 1873-1901.

[36] Janiak, A. (1988). General flow-shop scheduling with resource constraints. *The International Journal of Production Research*, *26*(6), 1089-1103.

[37] Kis, T. (2003). Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, 151(2), 307-332.

[38] 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.

[39] Kis, T., & Kovács, A. (2012). On bilevel machine scheduling problems. *OR Spectrum*, 34(1), 43-68.

[40] Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research*, *105*(1), 66-71.

[41] Li, J., Zhang, L., ShangGuan, C., & Kise, H. (2010). A GA-based heuristic algorithm for non-permutation two-machine robotic flow-shop scheduling problem of minimizing total weighted completion time. In: *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on* (pp. 1281-1285). IEEE.

[42] Li, J. Q., & Pan, Q. K. (2015). Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, *316*, 487-502.

[43] 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.

[44] 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.

[45] Liberopoulos, G., Kozanidis, G., & Hatzikonstantinou, O. (2010). Production scheduling of a multi-grade PET resin plant. *Computers & Chemical Engineering*, 34(3), 387-400.

[46] 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.

[47] Lin, S. W., Ying, K. C., & Lee, Z. J. (2009). Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times. *Computers & Operations Research*, *36*(4), 1110-1121.

[48] Lin, Shih-Wei, and Kuo-Ching Ying. "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics." *Omega* 64 (2016): 115-125.

[49] Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering*, *37*(1), 57-61.

[50] Liu, S., & Ong, H. L. (2002). A comparative study of algorithms for the flowshop scheduling problem. *Asia-Pacific Journal of Operational Research*, *19*(2), 205.

[51] Mehravaran, Y., & Logendran, R. (2012). Non-permutation flowshop scheduling in a supply chain with sequence-dependent setup times. *International Journal of Production Economics*, *135*(2), 953-963.

[52] Mehravaran, Y., & Logendran, R. (2013). Non-permutation flowshop scheduling with dual resources. *Expert Systems with Applications*, *40*(13), 5061-5076.

[53] Méndez, C. A., & Cerdá, J. (2003). An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optimization and Engineering*, 4(1), 7-22.

[54] Merigó, José M., and Jian-Bo Yang. "A Bibliometric Analysis of Operations Research and Management Science." *Omega* (2016).

[55] Mohammadi, M., Fatemi Ghomi, S. M. T., Karimi, B., & Torabi, S. A. (2010). MIP-based heuristics for lotsizing in capacitated pure flow shop with sequence-dependent setups. *International Journal of Production Research*, *48*(10), 2957-2973.

[56] Moukrim, A., Rebaine, D., & Serairi, M. (2014). A branch and bound algorithm for the two-machine flowshop problem with unit-time operations and time delays. *RAIRO-Operations Research*, *48*(2), 235-254.

[57] Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

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

[59] Nikjo, B., & Zarook, Y. (2014). A Non-Permutation Flow Shop Manufacturing Cell Scheduling Problem with Part's Sequence Dependent Family Setup Times. *International Journal of Applied Metaheuristic Computing (IJAMC)*, *5*(4), 70-86.

[60] Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, *181*(12), 2455-2468.

[61] Papadimitriou, C. H., & Kanellakis, P. C. (1980). Flowshop scheduling with limited temporary storage. *Journal of the ACM (JACM)*, *27*(3), 533-549

[62] Pesch, E., & Sterna, M. (2009). Late work minimization in flow shops by a genetic algorithm. *Computers & Industrial Engineering*, *57*(4), 1202-1209.

[63] Pinedo, M. (2012). *Scheduling*. Springer.

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

[65] Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2003). Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems. *Computers & Industrial Engineering*, *44*(1), 133-157.

[66] Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2004). Generating non-permutation schedules in flowline-based manufacturing sytems with sequence-dependent setup times of jobs: a heuristic approach. *The International Journal of Advanced Manufacturing Technology*, *23*(1-2), 64-78.

[67] Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2004) b. Relative performance evaluation of permutation and non-permutation schedules in flowline-based manufacturing systems with flowtime objective. *The International Journal of Advanced Manufacturing Technology*, *23*(11-12), 820-830.

[68] Rahmani, D., Ramezanian, R., & Saidi-Mehrabad, M. (2014). Multi-objective flow shop scheduling problem with stochastic parameters: fuzzy goal programming approach. *International Journal of Operational Research*, *21*(3), 322-340.

[69] Rajendran, C., & Ziegler, H. (2001). A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs. *European Journal of Operational Research*, *131*(3), 622-634.

[70] Ramezanian, R., Saidi-Mehrabad, M., & Rahmani, D. (2011). Flow Shop Scheduling Problem with Missing Operations: Genetic Algorithm and Tabu Search. *International Journal of Applied Operational Research*, *1*(2), 21-30.

[71] Ramezanian, R., & Saidi-Mehrabad, M. (2013). Hybrid simulated annealing and MIP-based heuristics for stochastic lot-sizing and scheduling problem in capacitated multi-stage production system. *Applied Mathematical Modelling*, *37*(7), 5134-5147.

[72] Rayward-Smith, V. J., & Rebaine, D. (2008). Analysis of heuristics for the UET two-machine flow shop problem with time delays. *Computers & Operations Research*, *35*(10), 3298-3310.

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

[74] Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review: Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439-1454.

[75] Rossi, A., & Lanzetta, M. (2013). Nonpermutation flow line scheduling by ant colony optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *27*(04), 349-357.

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

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

[78] 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.

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

[80] Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, 165(1), 34-54.

[81] Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1-18.

[82] Sadjadi, S. J., Aryanezhad, M. B., & Ziaee, M. (2008). The general flowshop scheduling problem: mathematical models. *Journal of Applied Sciences*, *8*(17), 3032-3037.

[83] Sadjadi, S. J., Bouquard, J. L., & Ziaee, M. (2008). An ant colony algorithm for the flowshop scheduling problem. *Journal of Applied Sciences*, *8*(21), 3938-3944.

[84] Schwindt, C., & Trautmann, N. (2000). Batch scheduling in process industries: an application of resource–constrained project scheduling. *OR Spectrum*, *22*(4), 501-524.

[85] 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.

[86] Strusevich, V. A., & Zwaneveld, C. M. (1994). On non-permutation solutions to some two machine flow shop scheduling problems. *Zeitschrift für Operations Research*, *39*(3), 305-319.

[87] Swaminathan, R., Fowler, J. W., Pfund, M. E., & Mason, S. J. (2004) Minimizing total weighted tardiness in a dynamic flowshop with variable processing times. In: *IIE Annual Conference. Proceedings* (p. 1). Institute of Industrial Engineers-Publisher.

[88] Swaminathan, R., Pfund, M. E., Fowler, J. W., Mason, S. J., & Keha, A. (2007). Impact of permutation enforcement when minimizing total weighted tardiness in dynamic flowshops with uncertain processing times. *Computers & Operations Research*, *34*(10), 3055-3068.

[89] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285.

[90] Tandon, M., Cummings, P. T., & LeVan, M. D. (1991). Flowshop sequencing with non-permutation schedules. *Computers & Chemical Engineering*, *15*(8), 601-607.

[91] 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.

[92] Vahedi Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. *International Journal of Production Research*, *51*(12), 3501-3515.

[93] Vahedi-Nouri, B., Fattahi, P., Tavakkoli-Moghaddam, R., & Ramezanian, R. (2014). A general flow shop scheduling problem with consideration of position-based learning effect and multiple availability constraints. *The International Journal of Advanced Manufacturing Technology*, *73*(5-8), 601-611.

[94] 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.

[95] Yenisey, M. M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*,*45*, 119-135.

[96] 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.

[97] 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.

[98] 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.

[99] Zhang, S.-Y., Lu, Z.-Q., Cui,W.-W (2014). Flow shop scheduling optimization algorithm with periodical maintenance. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS.* 20(6), 1379-1387.

[100] Zheng, T., & Yamashiro, M. (2010, September). A novel quantum differential evolutionary algorithm for non-permutation flow shop scheduling problems. In: *Electrical Engineering Computing Science and Automatic Control (CCE), 2010 7th International Conference on* (pp. 357-362). IEEE.

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