A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem

Article *in* INFORMS Journal on Computing · August 2008

DOI: 10.1287/ijoc.1070.0258 · Source: DBLP

CITATIONS
175

READS
1,177

3 authors:

Gerardo Minella
Instituto Tecnológico de Informática Valencia
7 PUBLICATIONS **560** CITATIONS
SEE PROFILE

Rubén Ruiz
Universitat Politècnica de València
192 PUBLICATIONS **12,036** CITATIONS
SEE PROFILE

Michele Ciavotta
Università degli Studi di Milano-Bicocca
82 PUBLICATIONS **1,391** CITATIONS
SEE PROFILE

INFORMS Journal on Computing
Vol. 20, No. 3, Summer 2008, pp. 451–471
ISSN 1091-9856 | EISSN 1526-5528 | 08 | 2003 | 0451

DOI 10.1287/ijoc.1070.0258

A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem

Gerardo Minella, Rubén Ruiz
Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Valencia, Spain {mgerar@iti.upv.es, rruiz@eio.upv.es}

Michele Ciavotta
Dipartimento di Informatica e Automazione, Università degli studi Roma Tre, Roma, Italy, ciavotta@dia.uniroma3.it

This paper contains a complete and updated review of the literature for multiobjective flowshop problems, which are among the most studied environments in the scheduling research area. No previous comprehensive reviews exist in the literature. Papers about lexicographical, goal programming, objective weighting, and Pareto approaches have been reviewed. Exact, heuristic, and metaheuristic methods have been surveyed. Furthermore, a complete computational evaluation is also carried out. A total of 23 different algorithms including both flowshop-specific methods as well as general multiobjective optimization approaches have been tested under three different two-criteria combinations with a comprehensive benchmark. All methods have been studied under recent state-of-the-art quality measures. Parametric and nonparametric statistical testing is profusely employed to support the observed performance of the compared methods. As a result, we have identified the best-performing methods from the literature, which along with the review, constitutes a reference work for further research.

Key words: scheduling; flowshop; multiobjective; review; evaluation
History: Accepted by John Hooker, Area Editor for Constraint Programming and Optimization; accepted November 2007. Published online in *Articles in Advance* April 21, 2008.

1. Introduction

In the field of scheduling, the flowshop problem has been thoroughly studied for decades. The flowshop problem is defined by a set of $N = 1, 2, \ldots, n$ jobs that have to be processed on a set of $M = 1, 2, \ldots, m$ machines. The processing time of each job $j \in N$ on each machine $i \in M$ is known in advance and is denoted by p_{ij}. All N jobs visit the machines in the same order, which, without loss of generality, can be assumed to be $1, 2, \ldots, m$. The objective is to find a processing sequence of the jobs so that a given criterion is optimized. In general, the number of possible solutions results from the product of all possible job permutations across all machines, i.e., $(n!)^m$ solutions or "schedules." However, in the flowshop literature, it is very common to restrict the solution space by having the same permutation of jobs for all machines. The resulting problem is referred to as the *permutation flowshop problem* (PFSP), where $n!$ schedules are possible.

The majority of the literature for the PFSP is centered around a single optimization criterion or objective. However, a single criterion is deemed as insufficient for real and practical applications. Multiobjective optimization is without a doubt a very important research topic not only because of the multiobjective nature of most real-world problems, but also because there are still many open questions in this area. Over the last decade, multiobjective optimization has received a big impulse in operations research. Some new techniques have been developed to deal with functions and real-world problems that have multiple objectives, and many approaches have been proposed.

The easiest way of dealing with a multiobjective problem is the so-called "a priori" approach, where two or more objectives are weighted and combined into a single measure, usually linear. For example, given two optimization criteria F_1 and F_2, a single-objective problem is derived with a combined $\alpha F_1 + (1 - \alpha)F_2$ function, where $0 \leq \alpha \leq 1$. However, the major drawback in this approach is that α must be given a priori. If α is not known, an alternative is to obtain several solutions with varying values of α but even in this case, if F_1 and F_2 are measured in different scales, this approach might be challenging.

A more desirable approach is the "a posteriori" method. In this case, the aim is to obtain many solutions with as many associated values as objectives. In such cases, the traditional concept of "optimum" solution does not apply. A given solution A might have a better F_1 value than another solution B, but

at the same time, worse F_2 value. In this context, a set of solutions is obtained where their objective values form what is referred to as the *Pareto front*. In the Pareto front, all solutions are equally good because there is no way of telling which one is better or worse. In other words, all solutions belonging to a Pareto front are the "best" solutions for the problem in a multiobjective sense.

There are no comprehensive reviews in the literature about multimachine flowshops with several objectives. In the past years, a number of interesting algorithms have been proposed. However, new proposals are hardly validated against the existing literature, and when done, the quality indicators used are not appropriate. Additionally, the multiobjective literature is rich on advanced methods that have not been applied to the PFSP before. Therefore, the objective of this paper is to give an up-to-date review and evaluation of many existing metaheuristics for solving the multiobjective PFSP. A second objective is to adapt proposed methods from the general multiobjective optimization field to the PFSP. As we will see, the literature is marred with different multiobjective proposals, many of which have not been tested for scheduling problems. As a result, we identify the most promising methods for the most common criteria combinations. We evaluate a total of 23 methods from local search metaheuristics such as tabu search or simulated annealing to evolutionary approaches like genetic algorithms (GAs). Furthermore, we use the latest Pareto-compliant quality measures for assessing the effectiveness of the tested methods. Careful and comprehensive statistical testing is employed to ensure the accuracy of the conclusions. The remainder of this paper is organized as follows. In §2, we further introduce the single and multiobjective PFSP, along with some notation and complexity results. In §3, we review the general literature on multiobjective optimization as well as the existing results for the PFSP. Section 4 examines the different available quality measures for comparing multiobjective methods. Section 5 deals with the comparative evaluation of all the studied algorithms. Finally, in §6 some conclusions and further research topics are given.

2. Single and Multiobjective Optimization

Single optimization criteria for the PFSP are mainly based on the completion times for the jobs at the different machines, which are denoted by C_{ij}, $i \in M$, $j \in N$. Given a permutation π of n jobs, where $\pi_{(j)}$ denotes the job in the jth position of the sequence, the completion times are calculated with the following expression:

$$C_{i,\pi_{(j)}} = \max\{C_{i-1,\pi_{(j)}}, C_{i,\pi_{(j-1)}}\} + p_{i\pi_{(j)}}, \qquad (1)$$

where $C_{0,\pi_{(j)}} = 0$ and $C_{i,\pi_{(0)}} = 0 \ \forall i \in M, \forall j \in N$. Additionally, the completion time of job j equals C_{mj} and is commonly denoted as C_j for short.

By far, the most thoroughly studied single criterion is the minimization of the maximum completion time or makespan, denoted as $C_{\max} = C_{m,\pi_{(n)}}$. Under this objective, the PFSP is referred to as $F/prmu/C_{\max}$ according to Graham et al. (1979), and was shown by Garey et al. (1976) to be $\mathcal{NP}$-hard in the strong sense for more than two machines ($m > 2$). Recent reviews and comparative evaluations of heuristics and metaheuristics for this problem are given in Framinan et al. (2004), Ruiz and Maroto (2005), and Hejazi and Saghafian (2005). The second-most studied objective is the total completion time or $\text{TCT} = \sum_{j=1}^{n} C_j$. The PFSP with this objective ($F/prmu/\sum C_j$) is already $\mathcal{NP}$-hard for $m \geq 2$ according to Gonzalez and Sahni (1978). Some recent results for this problem can be found in El-Bouri et al. (2005) or Rajendran and Ziegler (2005). If there are no release times for the jobs, i.e., $r_j = 0 \ \forall j \in N$, then the total or average completion time equals the total or average flowtime, denoted as F in the literature.

Probably the third-most studied criterion is the total tardiness minimization. Given a due date d_j for job j, we denote by T_j the measure of tardiness of job j, which is defined as $T_j = \max\{C_j - d_j, 0\}$. As with the other objectives, total tardiness minimization results in an $\mathcal{NP}$-hard problem in the strong sense for $m \geq 2$ as shown in Du and Leung (1990). A recent review for the total tardiness version of the PFSP (the $F/prmu/\sum T_j$ problem) can be found in Vallada et al. (2008).

Single and multiobjective scheduling problems have been studied extensively. However, in the multiobjective case, the majority of studies use the simpler "a priori" approach where multiple objectives are weighted into a single one. As mentioned, the main problem in this method is that the weights for each objective must be given. The "a posteriori" multiobjective approach is more complex because in this case, there is no single optimum solution, but rather lots of "optimum" solutions. For example, given two solutions x_1 and x_2 for a given problem with two minimization objectives f_1 and f_2, and $f_1(\cdot)$ and $f_2(\cdot)$ being the objective values for a given solution. Is x_1 better than x_2 if $f_1(x_1) < f_1(x_2)$ but at the same time $f_2(x_1) > f_2(x_2)$? It is clear than in a multiobjective scenario, neither solution is better than the other. However, given a third solution x_3, we can say than x_3 is worse than x_1 if $f_1(x_1) < f_1(x_3)$ and $f_2(x_1) < f_2(x_3)$. To properly compare two solutions in a multiobjective optimization problem (MOOP), some definitions are needed. Without loss of generality, let us suppose that there are M minimization objectives in a MOOP. We use the operator $\lhd$ as "better than," so that the relation $x_1 \lhd x_2$

implies that x_1 is better than x_2 for any minimization objective.

Zitzler et al. (2003) present a much more extensive notation that is later extended in Paquete (2005) and more recently in Knowles et al. (2006). For the sake of completeness, some of this notation is also introduced here:

Strong (or strict) domination. A solution x_1 is said to strongly dominate a solution x_2 ($x_1 \lessdot x_2$) if

$$f_j(x_1) \lhd f_j(x_2) \quad \forall j = 1, 2, \ldots, M;$$

x_1 is better than x_2 for all the objective values.

Domination. A solution x_1 is said to dominate a solution x_2 ($x_1 \prec x_2$) if the following conditions are satisfied:

(1) $f_j(x_1) \not\rhd f_j(x_2)$ for $j = 1, 2, \ldots, M$; x_1 is not worse than x_2 for all the objective values.

(2) $f_j(x_1) \lhd f_j(x_2)$ for at least one $j = 1, 2, \ldots, M$.

Weak domination. A solution x_1 is said to weakly dominate a solution x_2 ($x_1 \preceq x_2$) if

$$f_j(x_1) \not\rhd f_j(x_2) \quad \text{for all } j = 1, 2, \ldots, M;$$

x_1 is not worse than x_2 for all the objective values.

Incomparable solutions. Solutions x_1 and x_2 are incomparable ($x_1 \| x_2$ or $x_2 \| x_1$) if

$$f_j(x_1) \not\trianglelefteq f_j(x_2) \quad \text{nor} \quad f_j(x_2) \not\trianglelefteq f_j(x_1) \quad \text{for all } j = 1, 2, \ldots, M.$$

These definitions can be extended to sets of solutions. A and B being two sets of solutions for a given MOOP, we further define:

Strong (or strict) domination. Set A strongly dominates set B ($A \lessdot B$) if

$$\text{every } x_i \in B \ggg \text{by at least one } x_j \in A.$$

Domination. Set A dominates set B ($A \prec B$) if

$$\text{every } x_i \in B \succ \text{by at least one } x_j \in A.$$

Better. Set A is better than set B ($A \lhd B$) if

$$\text{every } x_i \in B \trianglerighteq \text{by at least one } x_j \in A \text{ and } A \neq B.$$

Weakly domination. A weakly dominates B ($A \trianglerighteq B$) if

$$\text{every } x_i \in B \trianglerighteq \text{by at least one } x_j \in A.$$

Incomparability. Set A is incomparable with set B ($A \| B$) if

$$\text{neither} \quad A \trianglerighteq B \quad \text{nor} \quad B \trianglerighteq A.$$

Nondominated set. Among a set of solutions A, we refer to the nondominated subset A' such as $x^{A'} \in A' \prec x^A$ with $x^{A'} \neq x^A$ and $A' \subset A$.

Pareto global optimum solution. A solution $x \in A$ (A being a set of all feasible solutions of a problem) is a Pareto global optimum solution if and only if there is no $x' \in A$ such that $f(x') \prec f(x)$.

Pareto global optimum set. A set $A' \in A$ (A being a set of solutions of a problem) is a Pareto global optimum set if and only if it contains only and all Pareto global optimum solutions This set is commonly referred to as Pareto front.

In multiobjective optimization, two goals are usually aimed at. An approximation of the Pareto global optimum set is deemed good if it is close to this set. Additionally, a good spread of solutions is also desirable; i.e., an approximation set is good if the whole Pareto global optimum front is sufficiently covered.

One important question is concerned about the complexity of multiobjective flowshop scheduling problems. As mentioned above, the PFSP is already $\mathcal{NP}$-hard under any of three commented single objectives (makespan, total completion time, or total tardiness). Therefore, in a multiobjective PFSP, no matter if the approach is a priori or a posteriori, the resulting problem is also $\mathcal{NP}$-hard if it contains one or more $\mathcal{NP}$-hard objectives.

3. Literature Review on Multiobjective Optimization

The literature on multiobjective optimization is plentiful. However, the multiobjective PFSP field is relatively scarce, especially when compared against the number of papers published for this problem that consider one single objective. The few proposed multiobjective methods for the PFSP are mainly based on evolutionary optimization and some on local search methods like simulated annealing or tabu search.

It could be argued that many reviews have been published about multiobjective scheduling. However, we find that little attention has been paid to the flowshop scheduling problem. For example, the review by Nagar et al. (1995b) is mostly centered around single-machine problems. As a matter of fact, there are only four surveyed papers related with flowshop. In another review by T'kindt and Billaut (2001), we find about 15 flowshop papers reviewed where most of them are about the specific two-machine case. Another review is given by Jones et al. (2002). However, this is more a quantification of papers in multiobjective optimization. Finally, the more recent review of Hoogeveen (2005) mainly contains results for one-machine and parallel-machines scheduling problems. The papers reviewed about flowshop scheduling are all restricted to the two-machine case. For all these reasons, in this paper we provide a complete and comprehensive review about the multiobjective flowshop. However, note that we restrict ourselves to the

pure flowshop setting, i.e., with no additional constraints. In the following, we will use the notation of T'kindt and Billaut (2002) to specify the technique and objectives studied by each reviewed paper. For example, a weighted makespan and total tardiness bicriteria flowshop problem is denoted as $F//F_l(C_{\max}, T)$. For more details, the reader is referred to T'kindt and Billaut (2001, 2002).

3.1. Lexicographical and ε-Constraint Approaches

Lexicographical approaches have been also been explored in the literature. Daniels and Chambers (1990) proposed a constructive heuristic for the m machine flowshop where makespan is minimized subject to a maximum tardiness threshold, a problem denoted by $F/prmu/\varepsilon(C_{\max}/T_{\max})$. This heuristic, along with the one of Chakravarthy and Rajendran (1999) (see the next section), are compared with a method recently proposed in Framinan and Leisten (2006). In this later paper, the newly proposed heuristic is shown to outperform the methods of Daniels and Chambers (1990) and Chakravarthy and Rajendran (1999) both on quality and on the number of feasible solutions found. A different set of objectives is considered in Rajendran (1992), where the authors minimize total flowtime subject to optimum makespan value in a two-machine flowshop. Such an approach is valid for the PFSP problem because the optimum makespan can be obtained by applying the well-known algorithm of Johnson (1954). Rajendran (1992) proposes a branch-and-bound (B&B) method together with some heuristics for the problem. However, the proposed methods are shown to solve 24 jobs maximum.

In Neppalli et al. (1996), two GAs were proposed for solving the two-machine bicriteria flowshop problem also in a lexicographical way as in Rajendran (1992). The first algorithm is based on the vector-evaluated genetic algorithm (VEGA) of Schaffer (1985). In this algorithm, two subpopulations are maintained (one for each objective) and are combined by the selection operator for obtaining new solutions. In the second GA, referred to as the *weighted criteria approach*, a linear combination of the two criteria is considered. This weighted sum of objectives is used as the fitness value. The same problem is studied by Gupta et al. (1999), where a tabu search is employed. This algorithm is finely tuned by means of statistical experiments and shown to outperform some of the earlier existing methods. Gupta et al. (2002) present some local search procedures and three metaheuristics for a two-machine flowshop. The methods developed are simulated annealing, threshold accepting, and tabu search. The criteria to optimize are composed of several lexicographic pairs involving makespan, weighted flowtime, and weighted tardiness. The proposed methods are compared against the GA of Neppalli et al. (1996) and the results discussed.

Gupta et al. (2001) proposed nine heuristics for the two-machine case minimizing flowtime subject to optimum makespan, i.e., $\text{Lex}(C_{\max}, F)$. The authors identify some polynomially solvable cases and carry out a comprehensive analysis of the proposed heuristics. Insertion-based methods are shown to give the best results. The same problem is approached by T'kindt et al. (2002), where the authors propose an ant colony optimization (ACO) algorithm. The method is compared against a heuristic from T'kindt et al. (2003) and against other single-objective methods from the literature. Although in some cases the ACO method is slower, it is shown to give higher-quality results. T'kindt et al. (2003) work with the same problem. The authors propose a B&B method capable of solving instances of up to 35 jobs in a reasonable time. Some heuristics are also provided.

3.2. Weighted Objectives

As mentioned, most studies make use of the a priori approach. This means that objectives are weighted (mostly linearly) into a single combined criterion. After this conversion, most single-objective algorithms can be applied.

Nagar et al. (1995a) proposed a B&B procedure for solving a two-machine flowshop problem with a weighted combination of flowtime and makespan as the objective. The algorithm initializes the branch-and-bound tree with an initial feasible solution and an upper bound, both obtained from a greedy heuristic. This algorithm was able to find the optimal solutions of problems with two machines and up to 500 jobs, but only under some strong assumptions and data distributions. The same authors use this branch and bound in Nagar et al. (1996) as a tool for providing the initial population in a GA. The hybrid B&B + GA approach is tested for the same two-job bicriteria flowshop, and it is shown to outperform the pure B&B and GA algorithms. Another GA is presented in Sridhar and Rajendran (1996) for makespan and flowtime, including also idle time as a third criterion. The algorithm uses effective heuristics for initialization. Cavalieri and Gaiardelli (1998) study a realistic production problem that they model as a flowshop problem with makespan and tardiness criteria. Two GAs are proposed where many of their parameters are adaptive. Yeh (1999) proposes another B&B method that compares favorably against that of Nagar et al. (1995a). For unstructured problems, Yeh's B&B is able to solve up to 14-job instances in less time than the B&B of Nagar et al. (1995a). The same author improved this B&B in Yeh (2001) and finally proposed a hybrid GA in Yeh (2002) showing the best results among all previous works. Note that all these

papers of Yeh deal with the specific two-machine case only. Lee and Chou (1998) proposed heuristic methods and a mixed-integer programming model for the m machine problem combining makespan and flowtime objectives. Their study shows that the integer programming approach is only valid for very small instances. A very similar work and results was given in a paper by the same authors (see Chou and Lee 1999).

Sivrikaya-Şerifoğlu and Ulusoy (1998) presented three B&B algorithms and two heuristics for the two-machine flowshop with makespan and flowtime objectives. All these methods are compared among them in a series of experiments. The largest instances solved by the methods contain 18 jobs. A linear combination of makespan and tardiness is studied in Chakravarthy and Rajendran (1999), but in this case a simulated annealing (SA) algorithm is proposed. Chang et al. (2002) study the gradual-priority weighting approach in place of the variable weight approach for genetic and genetic local search methods. These two methods are related to those of Murata et al. (1996) and Ishibuchi and Murata (1998), respectively. In numerical experiments, the gradual-priority weighting approach is shown to be superior. Framinan et al. (2002) proposed several heuristics along with a comprehensive computational evaluation for the m machine makespan and flowtime flowshop problem. Allahverdi (2003) also studies the same objectives. A total of 10 heuristics are comprehensively studied in a computational experiment. Among the studied methods, three proposed heuristics from the author outperform the others. Several dominance relations for special cases are proposed as well.

A different set of objectives, namely, makespan and maximum tardiness, are studied by Allahverdi (2004). Two variations are tested. In the first one, a weighted combination of the two objectives subject to a maximum tardiness value is studied. In the second, the weighted combination of criteria is examined. The author proposes a heuristic and compares it against the results of Daniels and Chambers (1990) and Chakravarthy and Rajendran (1999). The proposed method is shown to outperform these two according to the results. Ponnambalam et al. (2004) proposed a GA that uses some ideas from the traveling salesman problem (TSP). The implemented GA is a straightforward one that just uses a weighted combination of criteria as the fitness of each individual in the population. The algorithm is not compared against any other method from the literature, and just some results on small flowshop instances are reported. Lin and Wu (2006) focuses on the two-machine case with a weighted combination of makespan and flowtime. The authors present a B&B method that is tested against a set of small instances. The proposed method is able to find optimum solutions to instances of up to 15 jobs in all cases. Lemesre et al. (2007) have studied the m machine problem with makespan and total tardiness criteria. A special methodology based on a B&B implementation, called two-phase method, is employed. Due to performance reasons, the method is parallelized. As a result, some instances of up to 20 jobs and 20 machines are solved to optimality. However, the reported solving times for these cases are of seven days in a cluster of four parallel computers.

3.3. Pareto Approaches

When focusing on the a posteriori approach, the number of existing studies drops significantly. In the previously commented work of Daniels and Chambers (1990), the authors also propose a B&B procedure for the $C_{\max}$ and $T_{\max}$ objectives that computes the Pareto global front for the case of two machines. A GA was proposed by Murata et al. (1996), which was capable of obtaining a Pareto front for makespan and total tardiness. This algorithm, referred to as MOGA, applies elitism by copying a certain number of individuals in the nondominated set to the next generation. The nondominated solutions are kept externally in an archive. The algorithm selection is based on a fitness value given to each solution on the basis of a weighted sum of the objective's values. The weights for each objective are randomly assigned at each iteration of the algorithm. The authors also test their proposed GA with three objectives including flowtime. Later, in Ishibuchi and Murata (1998), the algorithm is extended by using a local search step that is applied to every new solution, after the crossover and mutation procedures.

Sayın and Karabatı (1999) studied a B&B algorithm that generates the optimum Pareto front for a two-machine flowshop with makespan and flowtime objectives. The experimental evaluation compares only against heuristics like those of Johnson (1954) and Rajendran (1992). Some instances of up to 24 jobs are solved to optimality. Liao et al. (1997) proposed a B&B algorithm for the two-machine bicriteria optimization problem, with the objectives of minimizing makespan and number of tardy jobs and also with the objectives of makespan and total tardiness. The lower bound values are obtained by means of the Johnson algorithm for makespan, and Moore's EDD (early due date) algorithm for the number of tardy jobs. For each node of the partial schedules, two lower bounds are calculated using the above heuristics. The accepted nondominated schedules are kept in an external set. At the end of the algorithm, this set contains optimal Pareto front for the problem. Lee and Wu (2001) also studies the two-machine case with B&B methods, but with a combination of flowtime and total tardiness criteria. The authors do not

compare their proposed approach with the literature and just report the results of their algorithm. A new type of GA is shown by Bagchi (2001). This method is based on the NSGA method by Srinivas and Deb (1994). Some brief experiments are given for a single flowshop instance with flowtime and makespan objectives. Murata et al. (2001) improve the earlier MOGA algorithm of Murata et al. (1996). This new method, called CMOGA, refines the weight assignment. A few experiments with makespan and total tardiness criteria are conducted. The new CMOGA outperforms MOGA in the experiments carried out.

Ishibuchi et al. (2003) present a comprehensive study about the effect of adding local search to their previous algorithm (Ishibuchi and Murata 1998). The local search is only applied to good individuals and by specifying search directions. This form of local search was shown to give better solutions for many different multiobjective GAs. In Loukil et al. (2000), many different scheduling problems are solved with different combinations of objectives. The main technique used is a multiobjective tabu search (MOTS). The paper contains a general study involving single- and parallel-machine problems as well. Later, in Loukil et al. (2005), a similar study is carried out, but in this case the multiobjective approach employed is the simulated annealing algorithm (MOSA).

A B&B approach is also shown by Toktaş et al. (2004) for the two-machine case under makespan and maximum earliness criteria. To the best of our knowledge, such a combination of objectives has not been studied in the literature before. The procedure is able to solve problems of up to 25 jobs. The authors also propose a heuristic method. Suresh and Mohanasundaram (2004) propose a Pareto-based simulated annealing algorithm for makespan and total flowtime criteria. The proposed method is compared against that of Ishibuchi et al. (2003) and against an early version of the SA proposed later by Varadharajan and Rajendran (2005). The results, shown only for small problems of up to 20 jobs, show the proposed algorithm to be better on some specific performance metrics. Arroyo and Armentano (2004) studied heuristics for several two and three objective combinations among makespan, flowtime, and maximum tardiness. For two machines, the authors compare the heuristics proposed against the existing B&B methods of Daniels and Chambers (1990) and Liao et al. (1997). For the general m machine case, the authors compare the results against those of Framinan et al. (2002). The results favor the proposed method that is also shown to improve the results of the GA of Murata et al. (1996) if used as a seed sequence. The same authors developed a tabu search for the makespan and maximum tardiness objectives in Armentano and Arroyo (2004). The algorithm includes several advanced features like diversification and local search in several neighborhoods. For the two-machine case, again the proposed method is compared against Daniels and Chambers (1990) and for more than two machines against Ishibuchi and Murata (1998). The proposed method is shown to be competitive in numerical experiments. In a more recent paper, Arroyo and Armentano (2005) carry out a similar study but in this case using GAs as solution tools. Although shown to be better than other approaches, the authors do not compare this GA with their previous methods.

Makespan and total flowtime are studied by Varadharajan and Rajendran (2005) with the help of simulated annealing methods. These algorithms start from heuristic solutions that are further enhanced by improvement schemes. Two versions of these SA (MOSA and MOSA-II) are shown to outperform the GA of Ishibuchi and Murata (1998). Pasupathy et al. (2006) have proposed a Pareto-archived GA with local search and have tested it with the makespan and flowtime objectives. The authors test this approach against Ishibuchi and Murata (1998) and Chang et al. (2002). Apparently, the newly proposed GA performs better under some limited tests. Melab et al. (2006) propose a grid-based parallel GA aimed at obtaining an accurate Pareto front for makespan and total tardiness criteria. While the authors do not test their approach against other existing algorithms, the results appear promising. However, the running days are of 10 days in a set of computers operating as a grid. More recently, Rahimi-Vahed and Mirghorbani (2007) have proposed a complex hybrid multiobjective particle swarm optimization (MOPS) method. The considered criteria are flowtime and total tardiness. In this method, an elite tabu search algorithm is used as an initialization of the swarm. A parallel local search procedure is employed as well to enhance the solution represented by each particle. This complex algorithm is compared against the SPEAII multiobjective GA of Zitzler et al. (2001). MOPS yields better results than SPEAII according to the reported computational experimentation albeit at higher CPU time requirements. Finally, Geiger (2007) has published an interesting study where the topology of the multiobjective flowshop problem search space is examined. Using several local search algorithms, the author analyzes the distribution of several objectives and tests several combinations of criteria.

3.4. Goal Programming and Other Approaches

There are some cases of other multiobjective methodologies like goal programming (GP). For example, Selen and Hott (1986) proposed a mixed-integer goal programming formulation for a biobjective PFSP dealing with makespan and flowtime criteria. As

with every goal programming method, a minimum desired value for each objective has to be introduced. Later, Wilson (1989) proposed a different model with fewer variables but a larger number of constraints. However, both models have the same number of binary variables. The comparison between both models results in the one of Selen and Hott (1986) being better for problems with $n \geq 15$.

Many algorithms in the literature have been proposed that do not explicitly consider many objectives as in previous sections. For example, Ho and Chang (1991) propose a heuristic that is specifically devised for minimizing machine idle time in an m machine flowshop. Although the heuristic does not allow for setting weights or threshold values and does not work with the Pareto approach either, the authors test it against a number of objectives. A similar approach is followed by Gangadharan and Rajendran (1994), where a simulated annealing is proposed for the m machine problem and is evaluated under makespan and flowtime criteria. Along with the SA method, two heuristics are also studied. Rajendran (1995) proposes a heuristic for the same problem dealt with in Ho and Chang (1991). After a comprehensive numerical experimentation, the new proposed heuristic is shown to be superior to that of Ho and Chang's. A very similar study is also presented by the same author in Rajendran (1994). Ravindran et al. (2005) present three heuristics aimed at minimizing makespan and flowtime. The authors test the three proposed methods against the heuristic of Rajendran (1995) but using only very small instances of 20 jobs and 20 machines maximum. The three heuristics appear to outperform Rajendran's albeit slightly. It is difficult to draw a line in these type of papers because many authors test a given proposed heuristic under different objectives. However, the heuristics commented above were designed with several objectives in mind and therefore we have included them in the review.

To sum up, Table 1 contains, in chronological order, the reviewed papers along with the number of machines (2 or m), and the multiobjective approach along with the criteria, as well as the type of method used. In total, 54 papers have been reviewed. Among them, 21 deal with the specific two-machine case. From the remaining 33 that study the more general m machines, a total of 16 use the a posteriori or Pareto-based approach. The results of these methods are not comparable for several reasons. First, the authors do not always deal with the same combination of criteria. Second, comparisons are many times carried out with different benchmarks and against heuristics or older methods. Last and most importantly, the quality measures employed are not appropriate as recent studies have shown. The next section deals with these measures.

4. Multiobjective Quality Measures

As commented in previous sections, comparing the solutions of two different Pareto approximations coming from two algorithms is not straightforward. Two approximation sets A and B can be even incomparable. Recent studies like those of Zitzler et al. (2003) and Paquete (2005), or more recently, Knowles et al. (2006), are an example of the enormous effort being carried out to provide the necessary tools for a better evaluation and comparison of multiobjective algorithms. However, the multiobjective literature for the PFSP frequently uses quality measures that have been shown to be misleading. For example, in the two most recent papers reviewed (Rahimi-Vahed and Mirghorbani 2007, Geiger 2007), some metrics like generational distance or maximum deviation from the best Pareto front are used. These metrics, among other ones, are shown to be nonPareto-compliant in the study of Knowles et al. (2006), meaning that they can give a better metric for a given Pareto approximation front B and worse for another front A even in a case where $A \prec B$. What is worse, in the comprehensive empirical evaluation of quality measures given in Knowles et al. (2006), it is shown that the most frequently used measures are nonPareto-compliant and are demonstrated to give wrong and misleading results more often than not. Therefore, special attention must be given to the choice of quality measures to ensure sound and generalizable results.

Knowles et al. (2006) propose three main approaches that are safe and sound. The first one relies on the Pareto dominance relations among sets of solutions. It is possible to rank a given algorithm over another based on the number of times the resulting Pareto approximation fronts dominate (strong, regular, or weakly) each other. The second approach relies on quality indicators, mainly, the hypervolume I_H and the epsilon indicators that were already introduced in Zitzler and Thiele (1999) and Zitzler et al. (2003), respectively. Quality indicators usually transform a full Pareto approximation set into a real number. Last, the third approach is based on empirical attainment functions. Attainment functions give, in terms of the objective space, the relative frequency that each region is attained by the approximation set given by an algorithm. These three approaches range from straightforward and easy to compute in the case of dominance ranking to the not so easy and computationally intensive attainment functions.

In this paper, we choose the hypervolume (I_H) and the unary multiplicative epsilon (I_ε^1) indicators. The choice is first motivated by the fact that dominance ranking is best observed when comparing one algorithm against another. By doing so, the number of times the solutions given by the first algorithm strongly, regularly, or weakly dominate those given by

Table 1 Reviewed Papers for the Multiobjective Flowshop

Year	Author/s	m	Approach and objectives	Comments
1986	Selen and Hott	m	GP(C_{max}, F)	Mixed-integer goal programming formulation
1989	Wilson	m	GP(C_{max}, F)	Mixed-integer goal programming formulation
1990	Daniels and Chambers	m	$\epsilon(C_{max}/T_{max})$	Heuristics
	Daniels and Chambers	2	$\#(C_{max}, T_{max})$	B&B
1991	Ho and Chang	m	(C_{max}, F)	Heuristics
1992	Rajendran	2	Lex(C_{max}, F)	B&B, heuristics
1994	Gangadharan and Rajendran	m	(C_{max}, F)	Simulated annealing. Heuristics
	Rajendran	m	(C_{max}, F)	Heuristics
1995a	Nagar et al.	2	$F_l(C_{max}, F)$	B&B
	Rajendran	m	(C_{max}, F)	Heuristics
1996	Murata et al.	m	$\#(C_{max}, T)$, $\#(C_{max}, T, F)$	Genetic algorithms
	Nagar et al.	2	$F_l(C_{max}, F)$	Genetic algorithms
	Nepalli et al.	2	Lex(C_{max}, F), $\#(C_{max}, T, F)$	Genetic algorithms
	Sridhar and Rajendran	m	$F_l(C_{max}, F)$	Genetic algorithms
1997	Liao et al.	2	$\#(C_{max}, N_T)$, $\#(C_{max}, T)$	B&B
1998	Cavalieri and Gaiardelli	m	$F_l(C_{max}, T)$	Genetic algorithms
	Ishibuchi and Murata	m	$\#(C_{max}, T)$, $\#(C_{max}, T, F)$	Genetic algorithms
	Lee and Chou	2	$F_l(C_{max}, F)$	B&B
	Sivrikaya-Şerifoğlu and Ulusoy	2	$F_l(C_{max}, F)$	B&B, heuristics
1999	Chakravarthy and Rajendran	m	$F_l(C_{max}, F)$	Heuristics, integer programming
	Chou and Lee	2	$F_l(C_{max}, F)$	B&B
	Gupta et al.	2	Lex(C_{max}, F)	Tabu search
	Sayın and Karabatı	2	$\#(C_{max}, F)$	B&B
	Yeh	2	$F_l(C_{max}, F)$	B&B
2000	Loukil et al.	m	$\#(many)$	Tabu search. Many objectives studied
2001	Bagchi	m	$\#(C_{max}, F)$	Genetic algorithms
	Gupta et al.	2	Lex(C_{max}, F)	Heuristics
	Lee and Wu	2	$\#(F, T)$	B&B
	Murata et al.	m	$\#(C_{max}, T)$	Genetic algorithms
	Yeh	2	$F_l(C_{max}, F)$	B&B
2002	Chang et al.	m	$F_l(C_{max}, T)$, $F_l(C_{max}, T, F)$	Genetic algorithms
	Framinan et al.	m	$F_l(C_{max}, F)$	Heuristics
	Gupta et al.	2	Lex(F, C_{max}), Lex(T, C_{max})	Various methods. Weighted functions
	T'kindt et al.	2	Lex(C_{max}, F)	Ant colony optimization
	Yeh	2	$F_l(C_{max}, F)$	Hybrid genetic algorithm
2003	Allahverdi	m	$F_l(C_{max}, F)$	Heuristics
	Ishibuchi et al.	m	$\#(C_{max}, T)$, $\#(C_{max}, T, F)$	Genetic algorithms and local search
	T'Kindt et al.	2	Lex(C_{max}, F)	B&B, heuristics
2004	Allahverdi	m	$\epsilon(F_l(C_{max}, T_{max}), T_{max})$	
			$F_l(C_{max}, T_{max})$	Heuristics
	Armentano and Arroyo	m	$\#(C_{max}, T_{max})$	Tabu search
	Arroyo and Armentano	m	$\#(C_{max}, T_{max}, F)$	Heuristics, combinations of the three objectives
	Ponnambalam et al.	m	$F_l(C_{max}, F)$	Genetic algorithms
	Suresh and Mohanasundaram	m	$\#(C_{max}, F)$	Simulated annealing
	Toktaş et al.	2	$\#(C_{max}, E_{max})$	B&B, heuristics
2005	Arroyo and Armentano	m	$\#(C_{max}, T_{max})$, $\#(C_{max}, T)$	Genetic algorithms
	Loukil et al.	m	$\#(many)$	Simulated annealing. Many objectives studied
	Ravindran et al.	m	(C_{max}, T)	Heuristics
	Varadharajan and Rajendran	m	$\#(C_{max}, F)$	Simulated annealing
2006	Framinan and Leisten	m	$\epsilon(C_{max}/T_{max})$	Heuristics
	Lin and Wu	2	$F_l(C_{max}, F)$	B&B
	Melab et al.	m	$\#(C_{max}, T)$	Parallel genetic algorithms
	Pasupathy et al.	m	$\#(C_{max}, F)$	Genetic algorithms
2007	Geiger	m	$\#(many)$	Local search
	Lemesre et al.	m	$F_l(C_{max}, T)$	B&B. Parallelism
	Rahimi-Vahed and Mirghorbani	m	$\#(F, T)$	Hybrid particle swarm optimization

the second gives a direct picture of the performance assessment among the two. The problem is that with 23 algorithms compared in this paper (see the next section), the possible two-algorithms pairs is 253, and therefore this type of analysis becomes unpractical. The same conclusion can be reached for the empirical attainment functions because these have to be compared in pairs. Furthermore, the computation of attainment functions is costly and the outcome has to be examined graphically one by one. As a result, such type of analysis is not useful in our case.

According to Knowles et al. (2006), I_H and I_ε^1 are Pareto-compliant and represent the state of the art as far as quality indicators are concerned. Additionally, combining the analysis of these two indicators is a powerful approach because if the two indicators provide contradictory conclusions for two algorithms, it means that they are incomparable. In the following, we give some additional details on how these two indicators are calculated.

The hypervolume indicator I_H, first introduced by Zitzler and Thiele (1999), just measures the area (in the case of two objectives) covered by the approximated Pareto front given by one algorithm. A reference point is used for the two objectives to bound this area. A greater value of I_H indicates both a better convergence to as well as a good coverage of the optimal Pareto front. Calculating the hypervolume can be costly and we use the algorithm proposed in Deb (2001). This algorithm already calculates a normalized and scaled value.

The binary epsilon indicator I_ε proposed initially by Zitzler et al. (2003) is calculated as follows: given two approximation sets A and B produced by two algorithms, the binary multiplicative epsilon indicator $I_\varepsilon(A, B)$ is equal to $\max_{x_B} \min_{x_A} \max_{1\le j\le M} f_j(x_A)/f_j(x_B)$, where x_A and x_B are each of the solutions given by algorithms A and B, respectively. Note that such a binary indicator would require one to calculate all possible pairs of algorithms. However, in Knowles et al. (2006), a unary I_ε^1 version is proposed, where the approximation set B is substituted by the best-known Pareto front. This is an interesting indicator because it tells us how much worse (ε) an approximation set is w.r.t. the best-known Pareto front in the best case. Therefore, ε gives us a direct performance measure. Note, however, that in our case some objectives might take a value of zero (for example, tardiness). Also, objectives must be normalized. Therefore, for the calculation of the I_ε^1 indicator, we first normalize and translate each objective; i.e., in the previous calculation, $f_j(x_A)$ and $f_j(x_B)$ are replaced by $(f_j(x_A) - f_j^-)/(f_j^+ - f_j^-) + 1$ and $(f_j(x_B) - f_j^-)/(f_j^+ - f_j^-) + 1$, respectively, where f_j^+ and f_j^- are the maximum and minimum known values for a given objective j, respectively. As a result, our normalized I_ε^1 indicator will take values between one and two. A value of one for a given algorithm means that its approximation set is not dominated by the best-known one.

5. Computational Evaluation

In this work, we have implemented not only algorithms specifically proposed for the multiobjective PFSP, but also many other multiobjective optimization algorithms. In these cases, some adaptation has been necessary. In the following, we review the algorithms that have been considered.

5.1. Pareto Approaches for the Flowshop Problem

We now detail the algorithms that have been reimplemented and tested among those proposed specifically for the flowshop scheduling problem. These methods have already been reviewed in §3, and here we extend some details about them and their reimplementation.

The MOGA algorithm of Murata et al. (1996) was designed to tackle the multiobjective flowshop problem. It is a simple GA with a modified selection operator. During this selection, a set of weights for the objectives are generated. This way, the algorithm tends to distribute the search toward different directions. The authors also incorporate an elite preservation mechanism that copies several solutions from the actual Pareto front to the next generation. We will refer to our MOGA implementation as MOGA_Murata. Chakravarthy and Rajendran (1999) present a simple simulated annealing algorithm that tries to minimize the weighted sum of two objectives. The best solution between those generated by the earliest due date (EDD), least static slack (LSS), and NEH (from the heuristic of Nawaz et al. 1983) methods is selected to be the initial solution. The adjacent interchange scheme (AIS) is used to generate a neighborhood for the actual solution. Note that this algorithm, referred to as SA_Chakravarty, is not a real Pareto approach because the objectives are weighted. However, we have included it in the comparison to have an idea of how such methods can perform in practice. We "simulate" a Pareto approach by running SA_Chakravarty 100 times with different weight combinations of the objectives. All the 100 resulting solutions are analyzed, and the best nondominated subset is given as a result.

Bagchi (2001) proposed a modification of the well-known NSGA procedure (see the next section) and adapted it to the flowshop problem. This algorithm, referred to as ENGA, differentiates from NSGA in that it incorporates elitism. In particular, the *parent* and *offspring* populations are combined in a unique set, then a nondominated sorting is applied and 50% of the nondominated solutions are copied to the *parent* population of the following generation. Murata et al. (2001) enhanced the original MOGA of Murata et al.

(1996). A different way of distributing the weights during the run of the algorithm is presented. The proposed weight specification method makes use of a cellular structure which permits to better select weights to find a finer approximation of the optimal Pareto front. We refer to this later algorithm as CMOGA.

Suresh and Mohanasundaram (2004) proposed a Pareto archived simulated annealing (PASA) method. A new perturbation mechanism called the "segment-random insertion (SRI)" scheme is used to generate the neighborhood of a given sequence. An archive containing the nondominated solution set is used. A randomly generated sequence is used as an initial solution. The SRI is used to generate a neighborhood set of candidate solutions and each one is used to update the archive set. A fitness function that is a scaled weighted sum of the objective functions is used to select a new current solution. A restart strategy and a reannealing method are also implemented. We refer to this method as MOSA_Suresh. Armentano and Arroyo (2004) developed a multiobjective tabu search method called MOTS. The algorithm works with several paths of solutions in parallel, each with its own tabu list. A set of initial solutions is generated using a heuristic. A local search is applied to the set of current solutions to generate several new solutions. A clustering procedure ensures that the size of the current solution set remains constant. The algorithm also makes use of an external archive for storing all the nondominated solutions found during the execution. After some initial experiments, we found that under the considered stopping criterion (to be detailed later), less than 12 iterations were carried out. This, together with the fact that the diversification method is not sufficiently clear from the original text, has resulted in our implementation not including this procedure. The initialization procedure of MOTS takes most of the allotted CPU time for large values of n. Considering the large neighborhood employed, this all results in extremely lengthy computations for larger n values.

Arroyo and Armentano (2005) proposed a genetic local search algorithm with the following features: preservation of population's diversity, elitism (a subset of the current Pareto front is directly copied to the next generation), and usage of a multiobjective local search. The concept of Pareto dominance is used to assign fitness (using the nondominated sorting procedure and the crowding measure both proposed for the NSGAII) to the solutions and in the local search procedure. We refer to this method as MOGALS_Arroyo. A multiobjective simulated annealing (MOSA) is presented in Varadharajan and Rajendran (2005). The algorithm starts with an initialization procedure that generates two initial solutions using simple and fast heuristics. These sequences are enhanced by three improvement schemes and are later used, alternatively, as the solution of the simulated annealing method. MOSA tries to obtain nondominated solutions through the implementation of a simple probability function that attempts to generate solutions on the Pareto optimal front. The probability function is varied in such a way that the entire objective space is covered uniformly obtaining as many nondominated and well-dispersed solutions as possible. We refer to this algorithm as MOSA_Varadharajan.

Pasupathy et al. (2006) proposed a GA which we refer to as PGA_ALS. This algorithm uses an initialization procedure that generates four good initial solutions that are introduced in a random population. PGA_ALS handles a working population and an external one. The internal one evolves using a Pareto-ranking-based procedure similar to that used in NSGAII. A crowding procedure is also proposed and used as a secondary selection criterion. The nondominated solutions are stored in the external archive, and two different local searches are then applied to half of archive's solutions for improving the quality of the returned Pareto front. Finally, we have also reimplemented PILS from Geiger (2007). This new algorithm is based on iterated local search, which in turn relies on two main principles—*intensification* using a variable neighborhood local search and *diversification* using a perturbation procedure. The Pareto dominance relationship is used to store the nondominated solutions. This scheme is repeated through successive iterations to reach favorable regions of the search space.

Note that among the 16 multiobjective PFSP specific papers reviewed in §3, we are reimplementing a total of 10. We have chosen not to reimplement the GAs of Ishibuchi and Murata (1998) and Ishibuchi et al. (2003) because they were shown to be inferior to the multiobjective tabu search of Armentano and Arroyo (2004) and some others. Loukil et al. (2000, 2005) have presented some rather general methods applied to many scheduling problems. This generality and the lack of details have deterred us from trying a reimplementation. Arroyo and Armentano (2004) proposed just some heuristics, and finally, the hybrid particle swarm optimization (PSO) proposed by Rahimi-Vahed and Mirghorbani (2007) is incredibly complex, making use of parallel programming techniques, and therefore we have chosen not to implement it.

5.2. Other General Pareto Algorithms

The multiobjective literature is marred with many interesting proposals, mainly in the form of evolutionary algorithms, that have not been applied to the PFSP before. Therefore, in this section we review some of these methods that have been reimplemented and adapted to the PFSP.

Srinivas and Deb (1994) proposed the well-known nondominated sorting GA, referred to as NSGA. This method differs from a simple GA only for the way the selection is performed. The nondominated sorting (NDS) procedure iteratively divides the entire population into different Pareto fronts. The individuals are assigned a fitness value that depends on the Pareto front they belong to. Furthermore, this fitness value is modified by a factor that is calculated according to the number of individuals crowding a portion of the objective space. A sharing parameter σ_{share} is used in this case. All other features are similar to a standard GA. Zitzler and Thiele (1999) presented another GA referred to as SPEA. The most important characteristic of this method is that all nondominated solutions are stored in an external population. Fitness evaluation of individuals depends on the number of solutions from the external population they dominate. The algorithm also incorporates a clustering procedure to reduce the size of the nondominated set without destroying its characteristics. Finally, the population's diversity is maintained by using the Pareto dominance relationship. Later, Zitzler et al. (2001) proposed an improved SPEAII version that incorporates a different fine-grained fitness strategy to avoid some drawbacks of the SPEA procedure. Other improvements include a density estimation technique that is an adaptation of the kth nearest neighbor method and a new complex archive truncation procedure.

Knowles and Corne (2000) presented another algorithm called PAES. This method employs local search and a population archive. The algorithm is composed of three parts: the first one is the candidate solution generator, which has an archive of only one solution and generates a new one making use of random mutation. The second part is the candidate solution acceptance function, which has the task of accepting or discarding the new solution. The last part is the nondominated archive, which contains all the nondominated solutions found so far. According to the authors, this algorithm represents the simplest nontrivial approach to a multiobjective local search procedure. In the same paper, the authors present an enhancement of PAES referred to as $(\mu + \lambda)$-PAES. Here, a population of μ candidate solutions is kept. By using a binary tournament, a single solution is selected and λ mutant solutions are created using random mutation. Hence, a $\mu + \lambda$ population is created and a dominance score is calculated for each individual. μ individuals are selected to update the candidate population while an external archive of nondominated solutions is maintained. Another GA is proposed by Corne et al. (2000). This method, called PESA, uses an external population (EP) and an internal population (IP) to pursuit the goal of finding a well-spread Pareto front. A selection and replacement procedure based on the degree of crowding is implemented. A simple genetic scheme is used for the evolution of IP, while EP contains the nondominated solutions found. The size of the EP is upper bounded and a hypergrid-based operator eliminates the individuals in the more crowded zones. Later, in Corne et al. (2001), an enhanced PESAII method is provided. This algorithm differs from the preceding one only in the selection technique in which the fitness value is assigned according to a hyperbox calculation in the objective space. In this technique, instead of assigning a selective fitness to an individual, it is assigned to the hyperboxes in the objective space which are occupied by at least one element. During the selection process, the hyperbox with the best fitness is selected, and an individual is chosen at random among all inside the selected hyperbox.

In Deb (2002), an evolution of the NSGA was presented. This algorithm, called NSGAII, uses a new fast nondominated sorting (FNDS) procedure. Unlike the NSGA, here a rank value is assigned to each individual of the population and there is no need for a parameter to achieve fitness sharing. Also, a crowding value is calculated with a fast procedure and is assigned to each element of the population. The selection operator uses the rank and the crowding values to select the better individuals for the mating pool. An efficient procedure of elitism is implemented by comparing two successive generations and preserving the best individuals. This NSGAII method is extensively used in the multiobjective literature for the most varied problem domains. Later, Deb et al. (2002) introduced yet another GA called CNSGAII. Basically, in this algorithm the *crowding* procedure is replaced by a *clustering* approach. The rationale is that once a generation is completed, the previous generation has a size of P_{size} (*parent set*), and the current one (*offspring set*) is also of the same size. Combining both populations yields a $2P_{\text{size}}$ set, but only half of them are needed for the next generation. To select these solutions, the nondominated sorting procedure is applied first and the *clustering* procedure second.

Deb et al. (2002) studied another different GA. This method, called ε-MOEA, uses two co-evolving populations, the regular one called P and an archive A. At each step, two parent solutions are selected, the first from P and the second from A. An offspring is generated, and it is compared with each element of the population P. If the offspring dominates at least a single individual in P, then it replaces this individual. The offspring is discarded if it is dominated by P. The offspring individual is also checked against the individuals in A. In the archive population, the ε-dominance is used in the same way. For example, and using the

previous notation, a solution x_1 strongly ε-dominates another solution x_2 ($x_1 \prec_\varepsilon x_2$) if $f_j(x_1) - \varepsilon \lhd f_j(x_2)$.

Zitzler and Künzli (2004) proposed another method, called *B*-IBEA. The main idea in this method is defining the optimization goal in terms of a binary quality measure and directly using it in the selection process. *B*-IBEA performs binary tournaments for mating selection and implements environmental selection by iteratively removing the worst individual from the population and updating the fitness values of the remaining individuals. An ε-indicator is used. In the same work, an adaptive variation, called *A*-IBEA, is also presented. An adapted scaling procedure is proposed with the goal of making the algorithm's behavior independent from the tuning of the parameter k used in the basic *B*-IBEA version. Finally, Kollat and Reed (2005) proposed also a NSGAII variation, referred to as ε-NSGAII, by adding ε-dominance archiving and adaptive population sizing. The ε parameter establishes the size of the grid in the objective space. Inside each cell of the grid, no more than one solution is allowed. Furthermore, the algorithm works by alternating two phases. It starts by using a very small population of 10 individuals and several runs of NSGAII are executed. During these runs, all the nondominated solutions are copied to an external set. When there are no further improvements in the current Pareto front, the second phase starts. In this second phase, the ε-dominance procedure is applied on the external archive. The 23 reimplemented algorithms, either specific for the PFSP or general multiobjective proposals, are summarized in Table 2.

5.3. Benchmark and Computational Evaluation Details

Each one of the 23 proposed algorithms is tested against a new benchmark set. There are no known comprehensive benchmarks in the literature for the multiobjective PFSP. The only reference we know of is the work of Basseur (2005), where a small set of 14 instances is proposed. To carry out a comprehensive and sound analysis, a much larger set is needed. We augment the well-known instances of Taillard (1993). This benchmark is organized in 12 groups with 10 instances each. The groups contain different combinations of the number of jobs n and the number of machines m. The $n \times m$ combinations are $\{20, 50, 100\} \times \{5, 10, 20\}$, $200 \times \{10, 20\}$, and 500×20. The processing times (p_{ij}) in Taillard's instances are generated from a uniform distribution in the range $[1, 99]$. We take the first 110 instances and drop the last 10 instances in the 500×20 group because this size is deemed as too large for the experiments. Regarding the due dates for the tardiness criterion, we use the same approach of Hasija and Rajendran (2004). In this work, a tight due date d_j is assigned to each job $j \in N$ following the expression $d_j = P_j \times (1 + random \cdot 3)$, where $P_j = \sum_{i=1}^{m} p_{ij}$ is the sum of the processing times over all machines for job j and *random* is a random number uniformly distributed in $[0, 1]$. This method of generating due dates results in very tight to relatively tight due dates depending on

Table 2 Reimplemented Methods for the Multiobjective Flowshop

Acronym	Year	Author/s	Type
NSGA	1994	Srinivas and Deb	Genetic algorithm. General
MOGA_Murata	1996	Murata et al.	Genetic algorithm. Specific
SPEA	1999	Zitzler and Thiele	Genetic algorithm. General
SA_Chakravarty	1999	Chakravarthy and Rajendran	Simulated annealing. Specific
PAES	2000	Knowles and Corne	Population local search. General
$(\mu + \lambda)$-PAES	2000	Knowles and Corne	Population local search. General
PESA	2000	Corne et al.	Genetic algorithm. General
SPEAII	2001	Zitzler et al.	Genetic algorithm. General
PESAII	2001	Corne et al.	Genetic algorithm. General
ENGA	2001	Bagchi	Genetic algorithm. Specific
CMOGA	2001	Murata et al.	Genetic algorithm. Specific
NSGAII	2002	Deb	Genetic algorithm. General
CNSGAII	2002	Deb et al.	Genetic algorithm. General
ε-MOEA	2002	Deb et al.	Genetic algorithm. General
B-IBEA	2004	Zitzler and Künzli	Genetic algorithm. General
A-IBEA	2004	Zitzler and Künzli	Genetic algorithm. General
MOSA_Suresh	2004	Suresh and Mohanasundaram	Simulated annealing. Specific
MOTS	2004	Armentano and Arroyo	Tabu search. Specific
ε-NSGAII	2005	Kollat and Reed	Genetic algorithm. General
MOGALS_Arroyo	2005	Arroyo and Armentano	Genetic algorithm. Specific
MOSA_Varadharajan	2005	Varadharajan and Rajendran	Simulated annealing. Specific
PGA_ALS	2006	Pasupathy et al.	Genetic algorithm. Specific
PILS	2007	Geiger	Iterated local search. Specific

the actual value of *random* for each job, i.e., if *random* is close to zero, then the due date of the job is going to be really tight because it would be more or less the sum of its processing times. As a result, the job will have to be sequenced very early to avoid any tardiness. These 110 augmented instances can be downloaded from http://soa.iti.es.

Each algorithm has been carefully reimplemented following all the explanations given by the authors in the original papers. We have reimplemented all the algorithms in Delphi, version 2006. It should be noted that all methods share most structures and functions, and the same level of coding has been used; i.e., all of them contain the most common optimizations and speed-ups. FNDS is frequently used for most methods. Unless indicated differently by the authors in the original papers, the crossover and mutation operators used for the genetic methods are the two-point order crossover and insertion mutation, respectively. Unless explicitly stated, all algorithms incorporate a duplicate-deletion procedure in the populations, as well as in the nondominated archives.

The stopping criterion for most algorithms is the same and is given by a time limit depending on the size of the instance. The algorithms are stopped after a CPU running time of $n \cdot m/2 \cdot t$ milliseconds, where t is an input parameter. Giving more time to larger instances is a natural way of separating the results from the lurking "total CPU time" variable. Otherwise, if worse results are obtained for large instances, it would not be possible to tell if it is due to the limited CPU time or due to the instance size. Every algorithm is run 10 different independent times (replicates) on each instance with three different stopping criteria: $t = 100$, 150, and 200 milliseconds. This means that for the largest instances of 200×20, a maximum of 400 seconds of real CPU time (not wall time) is allowed. For every instance, stopping time, and replicate we use the same random seed as a common variance reduction technique.

We run every algorithm on a cluster of 12 identical computers with Intel Core 2 Duo E6600 processors running at 2.4 GHz with 1 GB of RAM. For the tests, each algorithm and instance replicate is randomly assigned to a single computer, and the results are collected at the end. According to §2, the three most-common criteria for the PFSP are makespan, total completion time, and total tardiness. All these criteria are of the minimization type. Therefore, all experiments are conducted for the three following criteria combinations: (1) makespan and total tardiness, (2) total completion time and total tardiness, and finally, (3) makespan and total completion time.

A total of 75,900 data points are collected per criteria combination if we consider the 23 algorithms, 110 instances, 10 replicates per instance, and three different stopping time criteria. In reality, each data point is an approximated Pareto front containing a set of vectors with the objective values. In total, there are $75,900 \cdot 3 = 227,700$ Pareto fronts, taking into account the three criteria combinations. The experiments have required approximately 5,100 CPU hours.

From the $23 \cdot 10 \cdot 3 = 690$ available Pareto front approximations for each instance and criteria combination, a FNDS is carried out and the best nondominated Pareto front is stored. These "best" 110 Pareto fronts for each criteria combination are available for future use of the research community and are downloadable from http://soa.iti.es. Additionally, a set of best Pareto fronts are available for the three different stopping time criteria. These last Pareto fronts are also used for obtaining the reference points for the hypervolume (I_H) indicator and are fixed to 1.2 times the worst-known value for each objective. Also, these best Pareto fronts are also used as the reference set in the multiplicative epsilon indicator (I_ε^1).

5.4. Makespan and Total Tardiness Results

According to the review carried out in the previous sections, makespan and total tardiness are two common criteria. Furthermore, a low makespan increases machine utilization and throughput. However, the best-possible makespan might sacrifice due dates, and therefore both objectives are not correlated. We test all 23 reimplemented methods for these two criteria. Table 3 shows the average hypervolume (I_H) and epsilon indicator (I_ε^1) values for all the algorithms. Note that the results are divided into the three different stopping criteria. Also, the methods are sorted in descending order of hypervolume value. Although later we will conduct several statistical experiments, we proceed now to comment on the results. First and foremost, both quality indicators are contradictory in very few cases. We can observe that as hypervolume values decrease, the epsilon indicator increases. There are some exceptions and these occur between consecutive algorithms with very similar hypervolume values, for example, SPEA and CNSGAII in positions 10 and 11 in the 200-ms time columns. Another interesting result is that the ranking of the algorithms does not practically change as the allowed CPU time is increased, and when it does it is motivated by small differences to start with. However, these are the observed average values across all instances, and as we will mention later, there are more pronounced differences when focusing on specific instance sizes.

Because the objective values are normalized and the worst solutions are multiplied by 1.2, the maximum hypothetical hypervolume is $1.2^2 = 1.44$. As we can see, MOSA_Varadharajan is very close to this

Table 3 Results for the Makespan and Total Tardiness Criteria

	Time	100			150			200	
Number	Method	I_H	I_ε^1	Method	I_H	I_ε^1	Method	I_H	I_ε^1
1	MOSA_Varadharajan	1.366	1.049	MOSA_Varadharajan	1.360	1.053	MOSA_Varadharajan	1.357	1.056
2	MOGALS_Arroyo	1.290	1.089	MOGALS_Arroyo	1.299	1.082	MOGALS_Arroyo	1.301	1.080
3	PESA	1.285	1.086	PESA	1.287	1.084	PESA	1.291	1.081
4	PESAII	1.280	1.089	PESAII	1.284	1.086	PESAII	1.288	1.084
5	PGA_ALS	1.273	1.116	PGA_ALS	1.267	1.116	PGA_ALS	1.255	1.118
6	MOTS	1.224	1.135	MOTS	1.232	1.131	MOTS	1.235	1.129
7	MOGA_Murata	1.178	1.148	MOGA_Murata	1.187	1.143	MOGA_Murata	1.195	1.138
8	CMOGA	1.150	1.162	CMOGA	1.169	1.150	CMOGA	1.181	1.143
9	NSGAII	1.142	1.168	NSGAII	1.156	1.160	NSGAII	1.165	1.155
10	SPEA	1.141	1.169	SPEA	1.154	1.161	SPEA	1.164	1.155
11	CNSGAII	1.137	1.169	CNSGAII	1.153	1.160	CNSGAII	1.162	1.154
12	ε-NSGAII	1.091	1.195	ε-NSGAII	1.106	1.187	ε-NSGAII	1.115	1.182
13	$(\mu+\lambda)$-PAES	1.086	1.197	$(\mu+\lambda)$-PAES	1.101	1.188	$(\mu+\lambda)$-PAES	1.110	1.183
14	PAES	1.049	1.219	ε-MOEA	1.051	1.222	ε-MOEA	1.061	1.216
15	ε-MOEA	1.045	1.225	PAES	1.035	1.227	PAES	1.028	1.230
16	MOSA_Suresh	0.976	1.290	MOSA_Suresh	0.960	1.301	MOSA_Suresh	0.955	1.303
17	SA_Chakravarty	0.894	1.395	SA_Chakravarty	0.882	1.402	SA_Chakravarty	0.870	1.410
18	PILS	0.803	1.409	PILS	0.843	1.379	PILS	0.866	1.363
19	ENGA	0.588	1.522	*A*-IBEA	0.592	1.518	*A*-IBEA	0.599	1.514
20	*A*-IBEA	0.578	1.528	ENGA	0.570	1.534	ENGA	0.559	1.542
21	SPEAII	0.537	1.544	SPEAII	0.524	1.550	SPEAII	0.522	1.549
22	NSGA	0.520	1.571	NSGA	0.502	1.584	NSGA	0.490	1.593
23	*B*-IBEA	0.411	1.640	*B*-IBEA	0.411	1.641	*B*-IBEA	0.417	1.637

Notes. Average quality indicator values for the 23 algorithms tested under the three different termination criteria. Each value is averaged across 110 instances and 10 replicates per instance (1,100 values). For each termination criteria level, the methods are sorted according to I_H.

value in all three stopping criteria. Similarly, the minimum possible epsilon indicator is one. Most interestingly, PESA and PESAII algorithms outperform PGA_ALS and MOTS by a noticeable margin. It has to be reminded that PESA and PESAII have just been reimplemented and adapted to the PFSP problem since both methods were proposed in the general multiobjective optimization literature; i.e., they were not built for the PFSP. PGA_ALS and MOTS are PFSP-specific algorithms, and in the case of MOTS, even for the same two objectives that have been tested. It is also interesting how MOGA_Murata is the seventh-best performer in the comparison, although it is more than 10 years old and one of the first multiobjective algorithms proposed for the PFSP. This algorithm manages to outperform CMOGA, proposed also by the same authors, and claimed to be better than MOGA_Murata. It has to be reminded that our reimplementations have been carried out according to the details given in all the reviewed papers. CMOGA might result to be better than MOGA_Murata under different CPU times or under specific optimizations. However, for the careful and comprehensive testing in this paper, this is not the case.

In a much more unfavorable position are the remaining PFSP-specific methods (MOSA_Suresh, SA_Chakravarthy, PILS, and ENGA). The bad performance of SA_Chakravarthy is expected. Recall that this method uses the a priori approach by weighting the objectives, and here we have run it for 100 different times with varying weights. Therefore, using this type of method in such a way for obtaining a Pareto front is not advisable. ENGA is, as indicated by the original authors, better than NSGA, but overall significantly worse than earlier methods like MOGA_Murata. Another striking result is the poor performance of the recent PILS method. Basically, PILS is an iterative improvement procedure and is extremely costly in terms of CPU time. Therefore, in our experimental setting, it is outperformed by most other methods.

It should be specified that not all methods stop by the allotted CPU time as a stopping criterion. Some methods carry out some local searches after completing the iterations and some others just cannot be properly modified to stop at a given point in time. In any case, all CPU times stay within a given acceptable interval. For example, for 100 milliseconds' stopping time and for the largest 200×20 instances tested, all methods should stop at 3.33 minutes. However, SA_Chakravarty stopped at 1.39 minutes, MOGALS_Arroyo at 3.36, MOTS at 3.52, PILS at 3.36, MOSA_Suresh at 12.8, and MOSA_Varadharajan at 2.23. For 200 milliseconds' stopping time, all methods should stop at 6.67 minutes for the largest instances. In this case, MOTS required 7.05 minutes, PILS 6.72, SA_Chakravarty 1.39, MOSA_Suresh 12.92, and MOSA_Varadharajan 2.16. Interestingly,

MOSA_Varadharajan, the best performer of the test, is actually a fast method; other not-so-well performers like MOSA_Suresh take a much longer time. All other methods can be controlled to stop at the specified time. These discrepancies in stopping times are only important for the largest instances because for the smallest ones, more or less all CPU times are similar.

Table 3 contains just average values and many of them are very similar. Although each average is composed of a very large number of data points, it is still necessary to carry out a comprehensive statistical experiment to assess if the observed differences in the average values are indeed statistically significant. A total of 12 different experiments are carried out. We do design of experiments (DOE) and parametric ANOVA analyses as well as nonparametric Friedman rank-based tests on both quality indicators and for the three different stopping criteria. The utility of showing both parametric as well as nonparametric tests is threefold. First, in the operations research and computer science literature, it is common to disregard parametric testing due to the fact that these types of tests are based on assumptions that the data have to satisfy. Nonparametric testing is many times preferred because it is "distribution-free." However, nonparametric testing is nowhere as powerful as parametric testing. Second, in nonparametric testing a lot of information is lost because the data has to be ranked and the differences in the values (be these large or small) are transformed into a rank value. Third, ANOVA techniques allow for a much deeper and richer study of the data. Therefore, we also compare both techniques in this paper to support these claims. For more information, the reader is refereed to Conover (1999) and Montgomery (2004). We carry out six multifactor ANOVAs where the type of instance is a controlled factor with 11 levels (instances from 20×5 to 200×20). The algorithm is another controlled factor with 23 levels. The response variable on each experiment is either the hypervolume or the epsilon indicator. Last, there is one set of experiments for each stopping time. Considering that each experiment contains 25,300 data points, the three main hypotheses of ANOVA (normality, homocedasticity, and independence of the residuals) are easily satisfied. To compare results, a second set of six experiments are performed. In this case, nonparametric Friedman rank-based tests are carried out. Because there are 23 algorithms and 10 different replicates, the results for each instance are ranked between 1 and 230. A rank of one represents the best result for hypervolume or epsilon indicator. We are performing four different statistical tests on each set of results (for example, for 100 CPU time we are testing ANOVA and Friedman on both quality indicators). Therefore, a correction on the confidence levels must be carried out because the same set of data is being used to make more than one inference. We take the most conservative approach, which is the Bonferroni adjustment, and we set the adjusted significance level α_s to $\alpha/4 = 0.05/4 \simeq 0.01$. This means that all the tests are carried out at a 0.01 adjusted confidence level for a real confidence level of 0.05.

Figures 1(a) and 1(b) show the means plot for the factor algorithm in the ANOVA for the epsilon indicator response variable and the means plot for the ranks of the epsilon indicator, respectively. Both figures refer to the 100 CPU time stopping criterion. For the parametric tests, we use Tukey's honest significant difference (HSD) intervals, which counteract the bias in multiple pairwise comparisons. Similar HSD intervals are used for the nonparametric tests. As can be seen, the nonparametric test is less powerful. Not only are the intervals much wider (recall that overlapping intervals indicate a nonstatistically significant difference), but ranking neglects the differences in the response variables. In the Online Supplement to this paper, available at http://joc.pubs.informs.org/ecompanion.html, where full figures are shown, PILS is shown to be better in rank than MOSA_Suresh and SA_Chakravarthy when we have already observed that it has worse average hypervolume and epsilon indicator values. The reason behind this behavior is that PILS is better for many small instances with a small difference in epsilon indicator than MOSA_Suresh and SA_Chakravarthy. However, it is much worse for some other larger instances. When one transforms this to ranks, PILS obtains a better rank more times and hence it appears to be better, when in reality it is marginally better more times but significantly worse many times as well. Concluding the discussion about parametric versus nonparametric, if the parametric hypotheses are satisfied (even if they are not strictly satisfied, as thoroughly explained in Montgomery 2004), it is much better to use parametric statistical testing. Note that we are using a very large data set; with medium or smaller data sets, the power of nonparametric tests drops significantly.

The relative ordering, as well as most observed differences of the algorithms, are statistically significant, as we can see from Figure 1(a). As a matter of fact, the only nonstatistically significant differences are those between $(\mu + \lambda)$-PAES and ε-NSGAII and between the algorithms CNSGAII, SPEA, and NSGAII. MOGALS_Arroyo, PESA, and PESAII are also equivalent.

Figure 2(a) shows the parametric results also for CPU time of 100, but for the hypervolume quality indicator.

Note that the y-axis is now inverted because a larger hypervolume indicates better results. The relative ordering of the algorithms is almost identical to that of Figure 1(a), the only difference being MOGALS_Arroyo PESA and PESAII. This means that

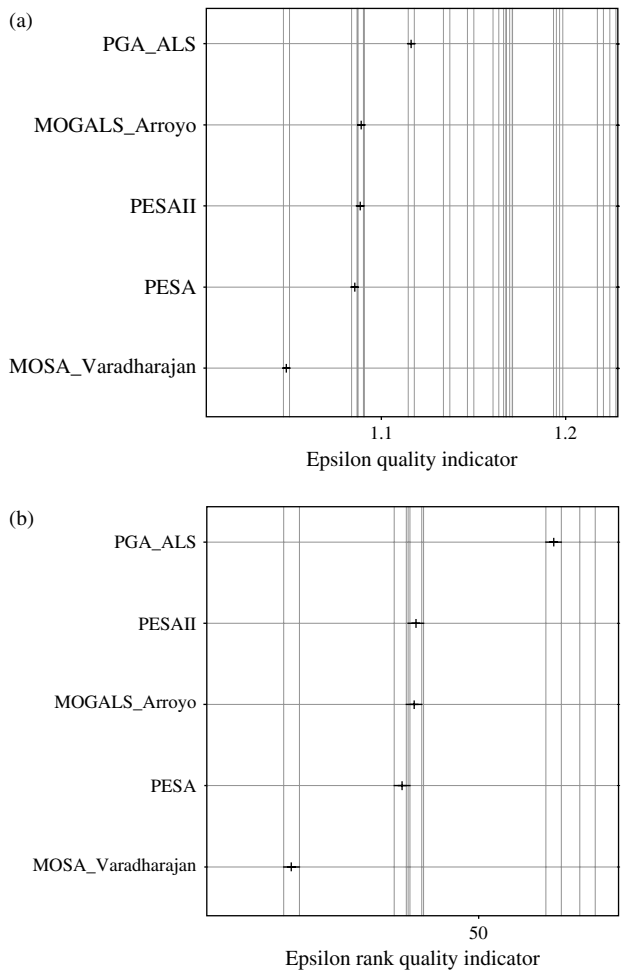

Figure 1 Plots for Epsilon Indicator Response Variable and 100 CPU Time Stopping Criterion

Notes. Makespan and total tardiness criteria. (a) Means plot and Tukey's HSD confidence intervals ($\alpha_s = 0.01$, $\alpha = 0.05$) for the algorithm factor in the ANOVA experiment. (b) Means plot and MSD confidence intervals ($\alpha_s = 0.01$, $\alpha = 0.05$) for the Friedman rank-based test.

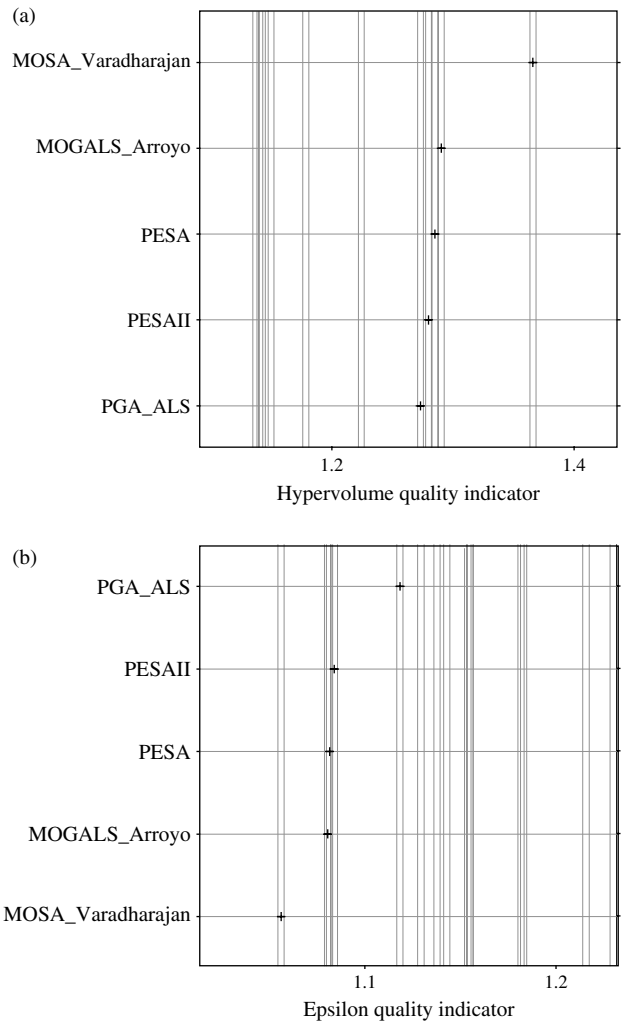

Figure 2 Means Plot and Tukey's HSD Confidence Intervals ($\alpha_s = 0.01$, $\alpha = 0.05$) **for the Algorithm Factor in the ANOVA Experiment**

Notes. Makespan and total tardiness criteria. (a) Hypervolume response variable and 100 CPU time stopping criterion. (b) Epsilon indicator response variable and 200 CPU time stopping criterion.

these algorithms show a very similar performance and on average are incomparable. A more in-depth instance-by-instance analysis would be needed to tell them apart.

As we have shown, increasing CPU time does not change, on average, the relative ordering of the algorithms. Figure 2(b) shows the parametric results for the epsilon indicator and for 200 CPU time.

We can observe the slight improvement on PILS, but although there is an important relative decrease on the average epsilon indicator, it is not enough to improve the results to a significant extent. More or less all other algorithms maintain their relative positions with little differences.

We commented before that the average performance is not constant for all instance sizes. For example, PILS can be a very good algorithm for small problems. Figure 3 shows the parametric results for the epsilon indicator and for 200-ms CPU time, but focuses on the 50×5 instances. As we can see, this group of instances is "easy" in the sense that a large group of algorithms is able to give very low epsilon indicator values. Most notably, PILS statistically outperforms MOSA_Varadharajan albeit by a small margin. This performance, however, is not consistent. For instances of size 50×10, PILS is the third-best performer, and for instances of size 100×5, PILS results to be the worst algorithm in the comparison. A worthwhile venue of research would be to investigate the strengths of PILS and to speed it up so that the performance is maintained for a larger group of instances.

5.5. Total Completion Time and Total Tardiness Results

Total completion time criterion is related to the amount of work-in-progress (WIP). A low total

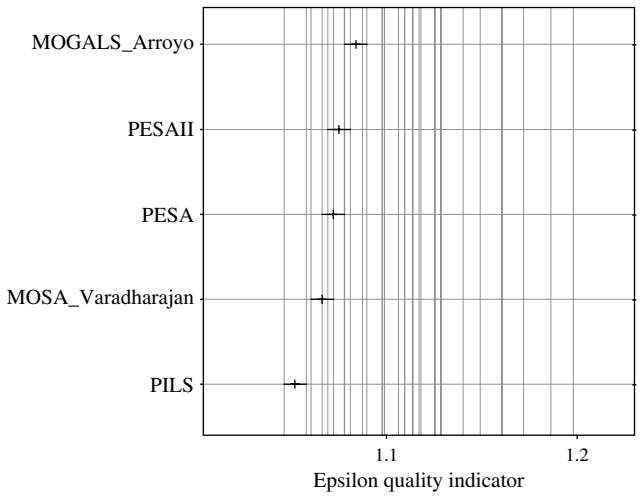

Figure 3 **Means Plot and Tukey's HSD Confidence Intervals** ($\alpha_s = 0.01$, $\alpha = 0.05$) **for the Algorithm Factor in the ANOVA Experiment for the Instance Group** 50×5

Notes. Epsilon indicator response variable and 200 CPU time stopping criterion. Makespan and total tardiness criteria.

completion time minimizes WIP and cycle time as well. As it was the case with the makespan criterion, total completion time is not correlated with total tardiness. Therefore, in this section we report the results for these two objectives.

Table 4 shows the corresponding average hypervolume and epsilon indicator values. One should expect that different criteria combinations should result in different performance for the algorithms tested. However, comparing the results of Tables 3 and 4 gives a different picture. MOSA_Varadhrajan and MOGALS_Arroyo produce the best results with independence of the allowed CPU time. MOGA_Murata as well as many others keep their relative position, and most other PFSP-specific methods are still on the lower positions despite the new criteria combination. The only noteworthy exception is the $(\mu + \lambda)$-PAES method which is ranking seventh, where for the other criteria combinations, it was ranking around 12th. As it was the case with the previous makespan and total tardiness criteria combination, both quality indicators and both parametric as well as nonparametric statistics give consistent results. For example, Figure 4 shows the parametric results for the epsilon indicator and for 200-ms CPU time across all instances. As can be seen, most of the observed differences from Table 4 are indeed statistically significant.

5.6. Makespan and Total Completion Time Results

Finally, we study the combination of makespan and total completion time. Contrary to the two previous criteria combinations, these two objectives are correlated. It is straightforward to see that a low makespan value will also result in a reduced total completion time. However, there are some scheduling

Table 4 **Results for the Total Completion Time and Total Tardiness Criteria**

	Time	100			150			200	
Number	Method	I_H	I_ε^1	Method	I_H	I_ε^1	Method	I_H	I_ε^1
1	MOSA_Varadharajan	1.381	1.034	MOSA_Varadharajan	1.377	1.036	MOSA_Varadharajan	1.375	1.036
2	MOGALS_Arroyo	1.330	1.057	MOGALS_Arroyo	1.326	1.060	MOGALS_Arroyo	1.324	1.061
3	PGA_ALS	1.311	1.076	PGA_ALS	1.313	1.075	PGA_ALS	1.315	1.074
4	MOTS	1.296	1.073	MOTS	1.302	1.071	MOTS	1.309	1.067
5	PESA	1.263	1.088	PESA	1.262	1.088	PESA	1.263	1.088
6	PESAII	1.263	1.088	PESAII	1.261	1.088	PESAII	1.262	1.087
7	$(\mu + \lambda)$-PAES	1.190	1.126	$(\mu + \lambda)$-PAES	1.196	1.123	$(\mu + \lambda)$-PAES	1.200	1.121
8	MOGA_Murata	1.183	1.128	MOGA_Murata	1.191	1.124	MOGA_Murata	1.198	1.120
9	CMOGA	1.163	1.138	CMOGA	1.178	1.130	CMOGA	1.188	1.125
10	NSGAII	1.158	1.143	NSGAII	1.170	1.137	NSGAII	1.176	1.134
11	CNSGAII	1.148	1.146	CNSGAII	1.162	1.139	CNSGAII	1.171	1.135
12	SPEA	1.137	1.151	SPEA	1.149	1.145	SPEA	1.157	1.141
13	ε-NSGAII	1.111	1.165	ε-NSGAII	1.131	1.155	ε-NSGAII	1.144	1.148
14	ε-MOEA	1.100	1.170	ε-MOEA	1.107	1.167	ε-MOEA	1.111	1.164
15	PAES	1.081	1.190	PAES	1.071	1.195	PAES	1.066	1.197
16	MOSA_Suresh	1.065	1.207	MOSA_Suresh	1.053	1.214	MOSA_Suresh	1.048	1.216
17	PILS	0.824	1.388	PILS	0.865	1.357	PILS	0.900	1.334
18	SA_Chakravarty	0.612	1.480	SA_Chakravarty	0.592	1.493	SA_Chakravarty	0.584	1.498
19	SPEAII	0.453	1.572	SPEAII	0.446	1.577	SPEAII	0.445	1.576
20	ENGA	0.426	1.609	ENGA	0.406	1.628	ENGA	0.397	1.635
21	NSGA	0.368	1.658	NSGA	0.348	1.679	*A*-IBEA	0.339	1.670
22	*A*-IBEA	0.333	1.676	*A*-IBEA	0.337	1.674	NSGA	0.338	1.686
23	*B*-IBEA	0.332	1.676	*B*-IBEA	0.337	1.673	*B*-IBEA	0.338	1.671

Notes. Average quality indicator values for the 23 algorithms tested under the three different termination criteria. Each value is averaged across 110 instances and 10 replicates per instance (1,100 values). For each termination criteria level, the methods are sorted according to I_H.

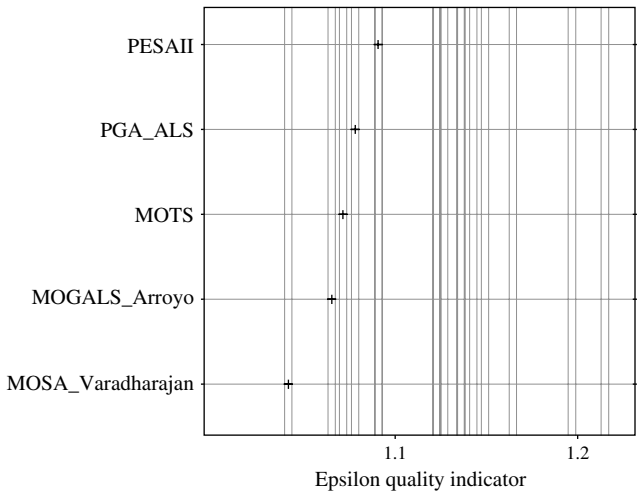

Figure 4 **Means Plot and Tukey's HSD Confidence Intervals** ($\alpha_s = 0.01$, $\alpha = 0.05$) **for the Algorithm Factor in the ANOVA Experiment**

Notes. Epsilon indicator response variable and 200 CPU time stopping criterion. Total completion time and total tardiness criteria.

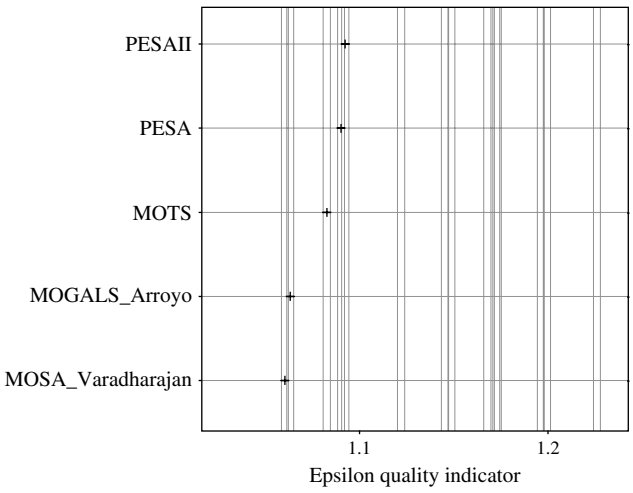

Figure 5 **Means Plot and Tukey's HSD Confidence Intervals** ($\alpha_s = 0.01$, $\alpha = 0.05$) **for the Algorithm Factor in the ANOVA Experiment**

Notes. Epsilon indicator response variable and 200 CPU time stopping criterion. Makespan and total completion time criteria.

scenarios where this is not true. In our case, the best-approximated Pareto fronts obtained throughout the tests contain several points, and in the case of large instances, Pareto fronts are composed of hundreds of points. Therefore, it seems that these two objectives are not as correlated as they seem. Table 5 shows the observed average values. Once again, the results are remarkably similar with respect to the other criteria combinations. MOSA_Varadhrajan and MOGALS_Arroyo are the best performers for the three criteria combinations, and the non-PFSP-specific methods PESA and PESAII are very good performers as well. As in the other cases, most observed differences in the average values are statistically significant as shown in Figure 5. Note that there is a

Table 5 **Results for the Makespan and Total Completion Time**

	Time	100			150			200	
Number	Method	I_H	I_ε^1	Method	I_H	I_ε^1	Method	I_H	I_ε^1
1	MOSA_Varadharajan	1.362	1.054	MOSA_Varadharajan	1.356	1.058	MOSA_Varadharajan	1.350	1.061
2	MOGALS_Arroyo	1.337	1.066	MOGALS_Arroyo	1.337	1.064	MOGALS_Arroyo	1.336	1.064
3	MOTS	1.309	1.087	MOTS	1.312	1.085	MOTS	1.312	1.083
4	PGA_ALS	1.271	1.120	PESA	1.273	1.091	PESA	1.275	1.091
5	PESA	1.269	1.092	PGA_ALS	1.272	1.121	PGA_ALS	1.272	1.122
6	PESAII	1.262	1.095	PESAII	1.267	1.093	PESAII	1.268	1.093
7	MOGA_Murata	1.160	1.154	MOGA_Murata	1.171	1.148	MOGA_Murata	1.176	1.146
8	CMOGA	1.134	1.167	CMOGA	1.155	1.155	CMOGA	1.167	1.149
9	CNSGAII	1.111	1.180	CNSGAII	1.126	1.171	CNSGAII	1.133	1.168
10	NSGAII	1.106	1.184	NSGAII	1.121	1.177	NSGAII	1.128	1.173
11	SPEA	1.099	1.186	SPEA	1.116	1.176	SPEA	1.122	1.173
12	$(\mu+\lambda)$-PAES	1.059	1.206	$(\mu+\lambda)$-PAES	1.071	1.199	$(\mu+\lambda)$-PAES	1.077	1.196
13	ε-NSGAII	1.051	1.213	ε-NSGAII	1.068	1.204	ε-NSGAII	1.077	1.199
14	ε-MOEA	1.027	1.233	ε-MOEA	1.036	1.228	ε-MOEA	1.040	1.226
15	PAES	1.005	1.238	PAES	0.992	1.244	PAES	0.980	1.252
16	MOSA_Suresh	0.950	1.296	MOSA_Suresh	0.936	1.303	MOSA_Suresh	0.922	1.314
17	SA_Chakravarty	0.813	1.410	PILS	0.836	1.383	PILS	0.861	1.367
18	PILS	0.794	1.413	SA_Chakravarty	0.798	1.422	SA_Chakravarty	0.781	1.431
19	ENGA	0.512	1.575	ENGA	0.492	1.590	ENGA	0.477	1.604
20	SPEAII	0.475	1.586	SPEAII	0.467	1.590	SPEAII	0.458	1.600
21	NSGA	0.446	1.625	*A*-IBEA	0.426	1.626	*B*-IBEA	0.425	1.631
22	*A*-IBEA	0.416	1.634	NSGA	0.426	1.642	*A*-IBEA	0.424	1.631
23	*B*-IBEA	0.416	1.634	*B*-IBEA	0.423	1.629	NSGA	0.409	1.658

Notes. Average quality indicator values for the 23 algorithms tested under the three different termination criteria. Each value is averaged across 110 instances and 10 replicates per instance (1,100 values). For each termination criteria level, the methods are sorted according to I_H.

comprehensive list of 36 figures in the Online Supplement to this paper.

6. Conclusions and Future Research

In this paper, we have conducted a comprehensive survey of the multiobjective literature for the flowshop, which is one of the most common and thoroughly studied problems in the scheduling field. This survey complements others such as those of Nagar et al. (1995b), T'kindt and Billaut (2001), or Hoogeveen (2005) that did not deal with multimachine flowshops. The papers surveyed include exact as well as heuristic techniques for many different multiobjective approaches.

Another important contribution in this paper is a complete and comprehensive computational evaluation of 23 different metaheuristics proposed for the Pareto or a posteriori multiobjective approach. Recent and state-of-the-art quality measures have been employed in an extensive experiment where makespan, total completion time, and total tardiness criteria have been studied in three different two-criteria combinations. The comparative evaluation not only includes flowshop-specific algorithms, but also adaptations of other general methods proposed in the multiobjective optimization literature. A new set of benchmark instances, based on the well-known benchmark of Taillard (1993), has been proposed and is available at http://soa.iti.es, along with the best-known Pareto fronts for the tested objectives.

A comprehensive statistical analysis of the results has been conducted with both parametric as well as nonparametric techniques. We have shown the preferred qualities of the parametric tests over the nonparametric counterparts, contrary to what is mainstream in the literature. The array of statistical analyses soundly supports the observed performances of the evaluated algorithms.

As a result, we have identified the best algorithms from the literature, which, along with the survey, constitute an important study and reference work for further research. Overall, the multiobjective simulated algorithm of Varadharajan and Rajendran (2005), MOSA_Varadharajan can be regarded as the best performer under our experimental settings. Another consistent performer is the genetic local search method MOGALS_Arroyo of Arroyo and Armentano (2005). Our adapted versions of PESA and PESAII from Corne et al. (2000, 2001), respectively, have shown a very good performance over many other flowshop-specific algorithms. The recent PILS method from Geiger (2007) has shown a promising performance for small instances. In our study, we have also shown that different stopping criteria as well as different criteria combinations result in little changes; i.e., the algorithms that give best results do so in a wide array of circumstances.

Further research stems from the possibility of devising new methods containing characteristics of the best-observed methods like MOSA_Varadharajan or MOGALS_Arroyo. Finally, there is almost no literature regarding complex multiobjective flowshop problems including additional characteristics like setup times or parallel machines, and new algorithms for such problems are desirable in practice.

Acknowledgments

The authors acknowledge the detailed comments received by the anonymous referees that have helped to improve an earlier version of this paper.

References

Allahverdi, A. 2003. The two- and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime. *Eur. J. Oper. Res.* **147** 373–396.

Allahverdi, A. 2004. A new heuristic for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness. *Comput. Oper. Res.* **31** 157–180.

Armentano, V. A., J. E. C. Arroyo. 2004. An application of a multiobjective tabu search algorithm to a bicriteria flowshop problem. *J. Heuristics* **10** 463–481.

Arroyo, J. E. C., V. A. Armentano. 2004. A partial enumeration heuristic for multiobjective flowshop scheduling problems. *J. Oper. Res. Soc.* **55** 1000–1007.

Arroyo, J. E. C., V. A. Armentano. 2005. Genetic local search for multiobjective flowshop scheduling problems. *Eur. J. Oper. Res.* **167** 717–738.

Bagchi, T. P. 2001. Pareto-optimal solutions for multiobjective production scheduling problems. E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, D. Corne, eds. *Evolutionary Multicriterion Optim., First Internat. Conf., EMO 2001, Zurich (March 7–9), Proc., Lecture Notes in Computer Science*, Vol. 1993. Springer, Heidelberg, 458–471.

Basseur, M. 2005. Conception d'algorithmes coopératifs pour l'optimisation multiobjectif: Application aux problèmes d'ordonnancement de type flow-shop. Ph.D. thesis, Laboratoire d'Informatique Fondamentale de Lille, Lille, France. [In French.]

Cavalieri, S., P. Gaiardelli. 1998. Hybrid genetic algorithms for a multiple-objective scheduling problem. *J. Intelligent Manufacturing* **9** 361–367.

Chakravarthy, K., C. Rajendran. 1999. A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. *Production Planning Control* **10** 707–714.

Chang, P. C., J. C. Hsieh, S. G. Lin. 2002. The development of gradual-priority weighting approach for the multiobjective flowshop scheduling problem. *Internat. J. Production Econom.* **79** 171–183.

Chou, F. D., C. E. Lee. 1999. Two-machine flowshop scheduling with bicriteria problem. *Comput. Indust. Engrg.* **36** 549–564.

Conover, W. 1999. *Practical Nonparametric Statistics*, 3rd ed. John Wiley & Sons, New York.

Corne, D. W., J. D. Knowles, M. J. Oates. 2000. The Pareto envelope-based selection algorithm for multiobjective optimization. M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo Guervós, H.-P. Schwefel, eds. *Parallel Problem Solving from Nature—PPSN VI, 6th Internat. Conf., Paris (September 18–20), Proc., Lecture Notes in Computer Science*, Vol. 1917. Springer, Heidelberg, 839–848.

Corne, D. W., N. R. Jerram, J. D. Knowles, M. J. Oates. 2001. PESA-II: Region-based selection in evolutionary multiobjective optimization. L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke, eds. *Proc. Genetic and Evolutionary Comput. Conf.* (*GECCO-2001*), *San Francisco*, Morgan Kaufmann, San Francisco, 283–290.

Daniels, R. L., R. J. Chambers. 1990. Multiobjective flow-shop scheduling. *Naval Res. Logist.* **37** 981–995.

Deb, K. 2001. *Multiobjective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, West Sussex, UK.

Deb, K. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Comput.* **6** 182–197.

Deb, K., M. Mohan, S. Mishra. 2002. A fast multiobjective evolutionary algorithm for finding well-spread Pareto-optimal solutions. Technical Report 2003002, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, India.

Du, J., J. Y.-T. Leung. 1990. Minimizing total tardiness on one machine is NP-hard. *Math. Oper. Res.* **15** 483–495.

El-Bouri, A., S. Balakrishnan, N. Popplewell. 2005. A neural network to enhance local search in the permutation flowshop. *Comput. Indust. Engrg.* **49** 182–196.

Framinan, J. M., R. Leisten. 2006. A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness. *Internat. J. Production Econom.* **99** 28–40.

Framinan, J. M., J. N. D. Gupta, R. Leisten. 2004. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *J. Oper. Res. Soc.* **55** 1243–1255.

Framinan, J. M., R. Leisten, R. Ruiz-Usano. 2002. Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *Eur. J. Oper. Res.* **141** 559–569.

Gangadharan, R., C. Rajendran. 1994. A simulated annealing heuristic for scheduling in a flowshop with bicriteria. *Comput. Indust. Engrg.* **27** 473–476.

Garey, M. R., D. S. Johnson, R. Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1** 117–129.

Geiger, M. J. 2007. On operators and search space topology in multiobjective flow shop scheduling. *Eur. J. Oper. Res.* **181** 195–206.

Gonzalez, T., S. Sahni. 1978. Flowshop and jobshop schedules: Complexity and approximation. *Oper. Res.* **26** 36–52.

Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.* **5** 287–326.

Gupta, J. N. D., K. Hennig, F. Werner. 2002. Local search heuristics for two-stage flow shop problems with secondary criterion. *Comput. Oper. Res.* **29** 123–149.

Gupta, J. N. D., V. R. Neppalli, F. Werner. 2001. Minimizing total flow time in a two-machine flowshop problem with minimum makespan. *Internat. J. Production Econom.* **69** 323–338.

Gupta, J. N. D., N. Palanimuthu, C. L. Chen. 1999. Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion. *Production Planning Control* **10** 251–265.

Hasija, S., C. Rajendran. 2004. Scheduling in flowshops to minimize total tardiness of jobs. *Internat. J. Production Res.* **42** 2289–2301.

Hejazi, S. R., S. Saghafian. 2005. Flowshop-scheduling problems with makespan criterion: A review. *Internat. J. Production Res.* **43** 2895–2929.

Ho, J. C., Y.-L. Chang. 1991. A new heuristic for the *n*-job, *M*-machine flow-shop problem. *Eur. J. Oper. Res.* **52** 194–202.

Hoogeveen, H. 2005. Multicriteria scheduling. *Eur. J. Oper. Res.* **167** 592–623.

Ishibuchi, H., T. Murata. 1998. A multiobjective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. Systems Man Cybernetics* **28** 392–403.

Ishibuchi, H., T. Yoshida, T. Murata. 2003. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. Evolutionary Comput.* **7** 204–223.

Johnson, S. M. 1954. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.* **1** 61–68.

Jones, D. F., S. K. Mirrazavi, M. Tamiz. 2002. Multiobjective metaheuristics: An overview of the current state-of-the-art. *Eur. J. Oper. Res.* **137** 1–9.

Knowles, J., D. W. Corne. 2000. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Comput.* **8** 149–172.

Knowles, J., L. Thiele, E. Zitzler. 2006. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich. [Revised version.]

Kollat, J. B., P. M. Reed. 2005. The value of online adaptive search: A performance comparison of nsgaii, ε-nsgaii, and ε-moea. C. A. Coello Coello, A. Hernández Aguirre, E. Zitzler, eds. *Evolutionary Multicriterion Optim., Third Internat. Conf., EMO 2005, Guanajuato, Mexico (March 9–11), Proc., Lecture Notes in Computer Science*, Vol 3410. Springer, Heidelberg, 386–398.

Lee, C. E., F. D. Chou. 1998. A two-machine flowshop scheduling heuristic with bicriteria objective. *Internat. J. Indust. Engrg.* **5** 128–139.

Lee, W. C., C. C. Wu. 2001. Minimizing the total flow time and the tardiness in a two-machine flow shop. *Internat. J. Systems Sci.* **32** 365–373.

Lemesre, J., C. Dhaenens, E. G. Talbi. 2007. An exact parallel method for a biobjective permutation flowshop problem. *Eur. J. Oper. Res.* **177** 1641–1655.

Liao, C.-J., W.-C. Yu, C.-B. Joe. 1997. Bicriterion scheduling in the two-machine flowshop. *J. Oper. Res. Soc.* **48** 929–935.

Lin, B. M. T., J. M. Wu. 2006. Bicriteria scheduling in a two-machine permutation flowshop. *Internat. J. Production Res.* **44** 2299–2312.

Loukil, T., J. Teghem, P. Fortemps. 2000. Solving multiobjective production scheduling problems with tabu search. *Control Cybernetics* **29** 819–828.

Loukil, T., J. Teghem, D. Tuyttens. 2005. Solving multiobjective production scheduling problems using metaheuristics. *Eur. J. Oper. Res.* **161** 42–61.

Melab, N., M. Mezmaz, E. G. Talbi. 2006. Parallel cooperative meta-heuristics on the computational grid. A case study: The biobjective flow-shop problem. *Parallel Comput.* **32** 643–659.

Montgomery, D. C. 2004. *Design and Analysis of Experiments*, 6th ed. John Wiley & Sons, New York.

Murata, T., H. Ishibuchi, M. Gen. 2001. Specification of genetic search directions in cellular multiobjective genetic algorithms. E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, D. Corne, eds. *Evolutionary Multicriterion Optim., First Internat. Conf., EMO 2001, Zurich (March 7–9), Proc., Lecture Notes in Computer Science*, Vol. 1993. Springer, Heidelberg, 82–95.

Murata, T., H. Ishibuchi, H. Tanaka. 1996. Multiobjective genetic algorithm and its applications to flowshop scheduling. *Comput. Indust. Engrg.* **30** 957–968.

Nagar, A., S. S. Heragu, J. Haddock. 1995a. A branch-and-bound approach for a two-machine flowshop scheduling problem. *J. Oper. Res. Soc.* **46** 721–734.

Nagar, A., S. S. Heragu, J. Haddock. 1995b. Mutiple and bicriteria scheduling: A lierature survey. *Eur. J. Oper. Res.* **81** 88–104.

Nagar, A., S. S. Heragu, J. Haddock. 1996. A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem. *Ann. Oper. Res.* **63** 397–414.

Nawaz, M., E. E. Enscore, Jr., I. Ham. 1983. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *OMEGA, Internat. J. Management Sci.* **11** 91–95.

Neppalli, V. R., C.-L. Chen, J. N. D. Gupta. 1996. Genetic algorithms for the two-stage bicriteria flowshop problem. *Eur. J. Oper. Res.* **95** 356–373.

Paquete, L. F. 2005. Stochastic local search algorithms for multiobjective combinatorial optimization: Method and analysis. Ph.D. thesis, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany.

Pasupathy, T., C. Rajendran, R. K. Suresh. 2006. A multiobjective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *Internat. J. Adv. Manufacturing Tech.* **27** 804–815.

Ponnambalam, S. G., H. Jagannathan, M. Kataria, A. Gadicherla. 2004. A TSP-GA multiobjective algorithm for flow-shop scheduling. *Internat. J. Adv. Manufacturing Tech.* **23** 909–915.

Rahimi-Vahed, A. R., S. M. Mirghorbani. 2007. A multiobjective particle swarm for a flow shop scheduling problem. *J. Combin. Optim.* **13** 79–102.

Rajendran, C. 1992. Two-stage flowshop scheduling problem with bicriteria. *J. Oper. Res. Soc.* **43** 871–884.

Rajendran, C. 1994. A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multicriteria. *Internat. J. Production Res.* **32** 2541–2558.

Rajendran, C. 1995. Heuristics for scheduling in flowshop with multiple objectives. *Eur. J. Oper. Res.* **82** 540–555.

Rajendran, C., H. Ziegler. 2005. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. *Comput. Indust. Engrg.* **48** 789–797.

Ravindran, D., A. N. Haq, S. J. Selvakuar, R. Sivaraman. 2005. Flow shop scheduling with multiple objective of minimizing makespan and total flow time. *Internat. J. Adv. Manufacturing Tech.* **25** 1007–1012.

Ruiz, R., C. Maroto. 2005. A comprehensive review and evaluation of permutation flowshop heuristics. *Eur. J. Oper. Res.* **165** 479–494.

Sayın, S., S. Karabatı. 1999. A bicriteria approach to the two-machine flow shop scheduling problem. *Eur. J. Oper. Res.* **113** 435–449.

Schaffer, J. D. 1985. Multiple objective optimization with vector evaluated genetic algorithms. *Proc. 1st Internat. Conf. Genetic Algorithms, Pittsburgh*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 93–100.

Selen, W. J., D. D. Hott. 1986. A mixed-integer goal-programming formulation of the standard flowshop scheduling problem. *J. Oper. Res. Soc.* **37** 1121–1128.

Sivrikaya-Şerifoğlu, F., G. Ulusoy. 1998. A bicriteria two-machine permutation flowshop problem. *Eur. J. Oper. Res.* **107** 414–430.

Sridhar, J., C. Rajendran. 1996. Scheduling in flowshop and cellular manufacturing systems with multiple objectives: A genetic algorithmic approach. *Production Planning Control* **7** 374–382.

Srinivas, N., K. Deb. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Comput.* **2** 221–248.

Suresh, R. K., K. M. Mohanasundaram. 2004. Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives. *Proc. IEEE Conf. Cybernetics and Intelligent Systems* (*CIS*), *Singapore*, Vol. 2. Institute of Electrical and Electronics Engineers, Washington, D.C., 712–717.

Taillard, E. 1993. Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **64** 278–285.

T'kindt, V., J.-C. Billaut. 2001. Multicriteria scheduling problems: A survey. *RAIRO Recherche opérationnelle—Oper. Res.* **35** 143–163.

T'kindt, V., J.-C. Billaut. 2002. *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer, Berlin.

T'kindt, V., J. N. D. Gupta, J.-C. Billaut. 2003. Two-machine flowshop scheduling with a secondary criterion. *Comput. Oper. Res.* **30** 505–526.

T'kindt, V., N. Monmarché, F. Tercinet, D. Laügt. 2002. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *Eur. J. Oper. Res.* **142** 250–257.

Toktaş, B., M. Azizoğlu, S. K. Köksalan. 2004. Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan. *Eur. J. Oper. Res.* **157** 286–295.

Vallada, E., R. Ruiz, G. Minella. 2008. Minimising total tardiness in the *m*-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Comput. Oper. Res.* **35** 1350–1373.

Varadharajan, T. K., C. Rajendran. 2005. A multiobjective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *Eur. J. Oper. Res.* **167** 772–795.

Wilson, J. M. 1989. Alternative formulations of a flowshop scheduling problem. *J. Oper. Res. Soc.* **40** 395–399.

Yeh, W.-C. 1999. A new branch-and-bound approach for the $n/2/flowshop/\alpha F + \beta C_{\max}$. *Comput. Oper. Res.* **26** 1293–1310.

Yeh, W.-C. 2001. An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem. *J. Manufacturing Systems* **20** 113–123.

Yeh, W.-C. 2002. A memetic algorithm for the $n/2/flowshop/\alpha F + \beta C_{\max}$ scheduling problem. *Internat. J. Adv. Manufacturing Tech.* **20** 464–473.

Zitzler, E., S. Künzli. 2004. Indicator-based selection in multiobjective search. *8th Internat. Conf. Parallel Problem Solving from Nature* (*PPSN VIII*), *Birmingham, UK*. Springer-Verlag, Berlin, 832–842.

Zitzler, E., L. Thiele. 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Comput.* **3** 257–271.

Zitzler, E., M. Laumanns, L. Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

Zitzler, E., L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evolutionary Comput.* **7** 117–132.