Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study

Michele Ciavotta[*], Gerardo Minella, Rubén Ruiz,

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática,

Ciudad Politécnica de la Innovación, Edificio 8G. Acceso B. Universitat Politècnica de València,

Camino de Vera s/n, 46022 Valencia, Spain

{mciavotta,mgerar}@iti.upv.es, rruiz@eio.upv.es

October 20, 2012

Abstract

The permutation flowshop scheduling problem has been thoroughly studied in recent decades, both from single objective as well as from multi-objective perspectives. To the best of our knowledge, little has been done regarding the multi-objective flowshop with Pareto approach when sequence dependent setup times are considered. As setup times and multi-criteria problems are important in industry, we must focus on this area. We propose a simple, yet powerful algorithm for the sequence dependent setup times flowshop problem with several criteria. The presented method is referred to as Restarted Iterated Pareto Greedy or RIPG and is compared against the best performing approaches from the relevant literature. Comprehensive computational and statistical analyses are carried out in order to demonstrate that the proposed RIPG method clearly outperforms all other algorithms and, as a consequence, it is a state-of-art method for this important and practical scheduling problem.

Keywords: scheduling, permutation flowshop, multi-objective, sequence dependent setup times, iterated greedy

[*]Corresponding author. Tel: +34 96 387 99 52. Fax: +34 963 87 72 39

1 Introduction

The flowshop scheduling problem (FSP) is characterized by a set N of n jobs that must be processed by a set M of m machines. All m machines are disposed in series and, without loss of generality, jobs visit machine 1 first, then machine 2 and so on until machine m. Each job needs a given, known in advance, fixed and non-negative processing time at each machine. This is denoted as p_{ij}, for each $j \in N$ and $i \in M$. A job cannot be in process at more than one machine simultaneously and one machine can only process one job at a time. The aim of this problem is to sequence the n jobs on the m machines so that a given criterion is optimized. Basically, there are $n!$ possible job permutations at each machine. In the most general case, each machine is associated with a different queue of jobs and hence, there are $(n!)^m$ possible solutions to this problem, where each solution is commonly referred to as a sequence.

The FSP has been criticized for being too theoretical as most real industry settings seldom fit into such a model. In part, this is attributable to the absence of setup times, which are very common in industry. Additionally, real-life problems have a multi-objective nature. Furthermore, flowshops rarely have the inter-machine flexibility to manipulate jobs in the processing queues. For all these reasons, this paper studies the multi-objective sequence dependent setup times permutation flowshop variant.

Setup times involve non-productive operations that have to be performed on machines and that are not part of the job's processing times. These may include, but are not limited to, cleaning, fixing and releasing parts to machines. Although on some occasions setup times can be included in the processing times, in the majority of industrial contexts it is not possible to ignore them. We can roughly classify setups into two main categories. The first one defines those setups which are *Sequence Independent* (SIST) i.e., the setup or changeover time for a machine only depends on the job that is to be processed next. The second category is the *Sequence Dependent* setup times (SDST) case, where setup time depends both on the current job being processed and on the next job in the sequence. This second category is much more complex and includes the first one as a particular case, permitting it to describe several operational scenarios. Furthermore, setups can be either anticipatory or non-anticipatory. In the former case, setups can be performed as soon as the machine is free and before the next job in the sequence is loaded. In this setting, we denote by $S_{ijk}, \forall i \in M, \forall j,k \in N, j \neq k$ the job sequence dependent setup time at machine i when processing job k after having processed job j.

A large body of research in the FSP deals with the optimization of a single criterion. The most commonly studied objective is the minimization of the maximum completion time or makespan, denoted to as $C_{\max}$ which is calculated as $\max_{j=1}^{n}\{C_{mj}\}$, where C_{mj} is the completion time of job j, i.e., the time at which job j finishes processing at machine m. Often, C_{mj} is simply denoted as C_j. Given a sequence π of n jobs where $\pi_{(l)}$ denotes the job occupying position l in the permutation with $l = \{1, \ldots, n\}$, $C_{\max}$ with SDST can be calculated in $\mathcal{O}(nm)$ steps with the following recursive formula: $C_{i,\pi_{(l)}} = \max\left\{C_{i-1,\pi_{(l)}}, C_{i,\pi_{(l-1)}} + S_{i,\pi_{(l-1)},\pi_{(l)}}\right\} + p_{i,\pi_{(l)}}$ where

$C_{i,\pi_{(0)}} = 0,\ C_{0,\pi_{(l)}} = 0$ and $S_{i,\pi_{(0)},\pi_{(l)}} = 0,\ \forall i \in M,\ l = \{1, \ldots, n\}$.
Makespan has been widely studied since a minimum value translates into a high resource utilization, throughput and Overall Equipment Efficiency (OEE). A second commonly studied criterion is the total flowtime, defined as $TFT = \sum_{j=1}^{n} C_j$ if we assume that all jobs are available at time 0 (i.e., the release dates or r_j are all zero). TFT, albeit certainly related, is quite different from makespan. A low TFT value reduces the Work-In-Process (WIP) inventory which is of paramount importance in real production shops. TFT also ensures a minimum cycle time in production environments. $C_{\max}$ and TFT are production-oriented criteria and neglect an important aspect of production which is client satisfaction. Jobs often model client orders that have a desired delivery date. This date is accounted for in scheduling by means of a due date d_j for each job. A job is said to be tardy if $C_j > d_j$. With this in mind, we define the tardiness of a job j as $T_j = \max\{C_j - d_j, 0\}$. As can be expected, not all orders from clients are equally important. To model this, a priority, importance or weight w_j is also given for each job. Considering all previous definitions, the third most commonly treated objective is the total weighted tardiness or $TWT = \sum_{j=1}^{n} w_j \cdot T_j$.
The previous objectives are the three most common, but are not the only ones. In practice, as one can expect, a combination of objectives is usually sought. For example, optimizing makespan results in a very high machine utilization. However, most due dates are likely to be violated. As a consequence, a multi-objective approach is needed. As concluded from the multi-objective flowshop review of Minella et al. (2008), most authors deal with several objectives in the most simple way, which is just adding them into a single weighted linear combination measure for example $\alpha \cdot C_{\max} + (1 - \alpha) \cdot TFT$, where $0 \leq \alpha \leq 1$. This is an example of the "a priori" approach. The problem with this is that often, objectives are measured in different scales and it is difficult to map α into a valid user preference. Another procedure, referred to as the "a posteriori" approach consists of finding out a set of solutions. Each solution represents a trade-off in the optimization of a given set of independent objective functions. This solution set is called the Pareto front. It is assumed that a multi-objective procedure returns this set to the decision maker, which later picks one solution from it.
We restrict ourselves to the permutation version of the flowshop problem where job passing is not allowed from machine to machine, i.e., the permutation of jobs cannot change from one machine to the next. This results in a smaller solution space of $n!$ This version of the problem is denoted as the permutation flowshop problem or PFSP in short. This special case is important in practice since in-process storage of products is very limited in most situations. Note that even this simplification of the problem, with no setups and one single objective still remains $\mathcal{NP}$-hard for many common criteria (Garey et al., 1976) and remains intractable for low values of n.

Following the well known classification scheme of Graham et al. (1979) and the extension of the notation for the multi-objective problems by T'kindt and Billaut (2006), the problem studied in this paper is denoted as $F/prmu, S_{ijk}/\#(\gamma_1, \gamma_2)$ where γ_1 and γ_2 are the two objectives that are considered in a Pareto approach. The two combinations of objectives that we consider in this paper are ($C_{\max}$,TFT) and ($C_{\max}$,TWT). To the best of our knowledge, this problem (even with a different set of objectives) has not been studied in the scientific literature and this paper presents the first attempt to solve it. We approach this problem with a recently proposed metaheuristic strategy, specially tailored for multi-objective problems.

The remainder of this paper is organized as follows: Section 2 presents a review of the literature on multi-objective optimization as well as existing results for the PFSP with setup times. Section 3 details the proposed algorithm which is later tested in Section 4 by carrying out a wide campaign of experiments and the results are statistically analyzed in detail. Finally, in Section 5, some conclusions and further research topics are given.

2 Literature review

To the best of our knowledge, no paper has been published that considers all characteristics of the problem studied in this work, i.e., multi-objective permutation flowshop problem with sequence dependent setup times. Hence, in the following subsections we present first a brief review of multi-objective flowshop and second a review about SDST flowshop with one single objective.

2.1 Multi-objective flowshop

The literature on multi-objective optimization is extremely rich. However, the multi-objective PFSP field is relatively scarce, specially in comparison with the large number of papers published dealing with the single criterion flowshop problem. The few proposed multi-objective methods for the PFSP are mainly based on evolutionary optimization and on local search techniques like simulated annealing (SA) or tabu search. In Minella et al. (2008), the authors carefully reviewed the literature related to this problem. Thus, here we restrict ourselves to only the most significant works and to some other more recent published material.

Methods belonging to the “a priori” multi-objective approach (weighted objective functions, lexicographical and goal optimization, etc.), in general, return a single solution, the closest one to decision-maker’s desires. Differently, methods belonging to the “a posteriori” approaches return several equivalent solutions (Pareto set) among which the decision maker can choose.
Focusing only on the “a posteriori” approach, the number of publications in the flowshop literature is reduced to a little set. A genetic algorithm (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 (Multi Objective Genetic Algorithm), applies elitism and the selection phase employs a fitness value assigned to each solution as a function of the weighted sum of the objectives. The weights for each objective are randomly assigned at each iteration of the algorithm. Later, in Ishibuchi and Murata (1998), the authors extended this algorithm by means of a local search procedure applied to every newly generated solution.

A genetic algorithm is shown by Bagchi (2001), which is based on the Srinivas and Deb (1994) NSGA method. Some short experiments are given for a single flowshop instance with flowtime and makespan objectives. Murata et al. (2001) improve the earlier MOGA algorithm by Murata et al. (1996). This new method, called CMOGA, refines the weight assignment.

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 multi-objective genetic algorithms. In Loukil et al. (2000), several scheduling problems are solved with different combinations of objectives. The main technique used is a multi-objective tabu search, referred to as MOTS. Later, in Loukil et al. (2005), a similar study is carried out. In this case the multi-objective simulated annealing (MOSA) approach is employed.

Suresh and Mohanasundaram (2004) propose a Pareto-based simulated annealing algorithm for makespan and total flowtime criteria. Experiments are conducted and the proposed method is compared against that of Ishibuchi et al. (2003) and against an early unpublished version of the SA later presented in Varadharajan and Rajendran (2005). Arroyo and Armentano (2004) studied heuristics for several two and three objective combinations among makespan, flowtime and maximum tardiness. 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, when used as a seed sequence, also improves the results of the GA of Murata et al. (1996). 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. 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 genetic algorithms. The makespan and total flowtime objectives 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). According to the comprehensive computational evaluation of Minella et al. (2008), where 23 methods were tested for the multi-objective flowshop, an enhanced version of MOSA_Varadharajan algorithm (named MOSA-II in the original paper) is shown to consistently outperform all other methods.

Pasupathy et al. (2006) proposed a Pareto-archived genetic algorithm with local search and have tested it with the makespan and flowtime objectives.

Geiger (2007) has published an interesting study where the topology of the multi-objective 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. Yanda and Tamura (2007) presented a variant of the NSGAII of Deb (2002), referred to as hMGA, which uses a working population with dynamic size made of only heterogeneous solutions. According to the authors, this choice prevents the algorithm from getting stalled in local optima. Framinan and Leisten (2008) presented an iterated greedy (IG) procedure based on the NEH heuristic. This algorithm is an evolution of the IG basic principle for the multi-objective PFSP. Recently, Rajendran and Ziegler (2009) proposed an ant-colony algorithm for the flowshop sceduling problem (MOACA) with the objective of minimizing the makespan and total flowtime. The authors presented 20 variants of the algorithm and some of them turned out to be highly competitive for the considered benchmark.

2.2 SDST flowshop

Hundreds of papers dealing with the permutation flowshop problem have been published in the literature but only a relatively minor fraction of them consider sequence dependent setup times. Exact techniques for the SDST permutation flowshop have shown rather limited results. The latest reference and most advanced study is that of Ríos-Mercado and Bard (2003) which studied the polyhedral structure of two mixed-integer programs for the SDST flowshop in order to generate more effective cuts to use in a branch-and-cut framework. Some heuristics and metaheuristic algorithms for the $F/S_{ijk}, prmu/C_{\max}$ have been proposed. For example, Ríos-Mercado and Bard (1998) presented a modification of the well known NEH heuristic for the regular flowshop from Nawaz et al. (1983) that takes into account setup times. In the same paper a GRASP algorithm is also proposed. In a later work, the same authors presented a modification of the heuristic of Simons (1992) resulting in a new method called HYBRID (Ríos-Mercado and Bard, 1999). Ruiz et al. (2005) proposed a genetic and a memetic algorithm for the $F/S_{ijk}, prmu/C_{\max}$. They carried out an comprehensive experimental study comparing their proposals against several methods adapted from the $F//C_{\max}$ problem.
About the $F/S_{ijk}, prmu/\sum_{j=1}^{n} w_j T_j$, little has been published. In Parthasarathy and Rajendran (1997a) and Parthasarathy and Rajendran (1997b), a Simulated Annealing heuristic was proposed for the SDST flowshop problem with the goal of minimizing the maximum weighted tardiness and the total weighted tardiness, respectively. Rajendran and Ziegler (1997) introduced an algorithm formed by a new heuristic and a local search improvement scheme for the weighted flowtime objective. Another similar work is that of Rajendran and Ziegler (2003) were a combined objective of total weighted flow-time and tardiness is considered. Ruiz and Stützle (2008) proposed two iterated greedy algorithms for the PFSP with sequence dependent setup times. The first one follows the guidelines of the IG framework adapted to setup times and the second incorporates a simple descent local search. More details can be found in Ruiz et al. (2005) where the authors carried out an extensive literature survey about this problem and in Allahverdi et al. (2008), an updated and comprehensive review of scheduling research with setup times.

3 Restarted Iterated Pareto Greedy

The iterated greedy methodology (IG) belongs to the stochastic local search techiques (SLS) and the basic scheme was first presented by Jacobs and Brusco (1995) for the set covering problem. Afterwards, Schrimpf et al. (2000) named as "Ruin and Recreate" a very similar algorithm for the vehicle routing problem. Iterated greedy method was applied by Ruiz and Stützle (2007) to the regular permutation flowshop problem with the makespan minimization objective. The results have encouraged several other authors to propose variants and adaptations to other problems, including the already cited paper of Ruiz and Stützle (2008) for the SDST flowshop, Pan et al. (2007) for the no-wait flowshop or Ying (2009) for the hybrid flowshop. In all these problems, IG has produced state-of-the-art results. Framinan and Leisten (2008) proposed a multi-objetive IG for the regular permutation flowshop problem which is basically an evolution of the NEH heuristic of Nawaz et al. (1983) modified to use Pareto dominance. As a result, it seems plausible to attempt an extension for the multi-objective flowshop with setup times. Note that IG works with a single solution and was proposed for a single objective, therefore, a just simple adaptation is not possible if high quality results have to be achieved.

Similarly to Minella et al. (2011), we propose an extension of the original Iterated Greedy algorithm named *Restarted Iterated Pareto Greedy* (RIPG), which is now presented. The rationale of this algorithm is very simple: a greedy multi-objective strategy is iteratively applied over a set of non-dominated solutions.
The proposed RIPG is broken into five phases: In the first phase (*Initialization*), an initial set of good solutions is generated using different heuristics, each one designed to attain good values for a specific criterion. The remaining four phases are iteratively repeated and constitute the main loop of the algorithm. The second phase, called *Selection*, chooses one solution from the current working set for the next phase: the *Greedy* one in which the selected solution is disrupted by means of a *Destruction* operator, by removing some elements and then a greedy procedure, called *Construction*, is applied. Afterwards, a *Local search* phase is applied over a selected element of the current working set. Lastly, a *Restart* procedure is implemented to prevent the algorithm from getting stuck in local optima. The following sections describe each phase in detail.

3.1 Algorithm initialization and selection phase

As is of common knowledge, initial solutions expressly generated to have opportune features often play an underlying role in creating a high performing algorithm. On the other hand, since our first concern is to create an algorithm capable of performing well for different pairs of obejectives, we chose heuristics that return good enough solutions for many different objectives ensuring also a sufficient degree of diversity. In order to generate a good *Initial Solution Set* (ISS) we make use of the initialization procedure proposed by Varadharajan and Rajendran (2005), which demonstrated to return high quality initial solutions in the review of Minella et al. (2008). This procedure uses the NEH heuristic of Nawaz et al. (1983) and the heuristic of Rajendran (1995), both designed for the optimization of a single criterion. Such heuristics are used to generate two distinct solutions for each objective to optimize.

In a first step, all initial solutions are processed by the Greedy phase one by one. The resulting frontiers of this process are added to the ISS and then, the dominated solutions are removed and the initial current working set is conformed. The aim behind this policy is to avoid that a likely large improvement during the initial iterations might generate a set of solutions that dominate the remaining initial solutions, impoverishing the quality and diversity of the working set too early. At each iteration of the algorithm, the selection phase is responsible for pointing the search towards promising directions. Selection achieves this goal by choosing one solution from the current working set on the basis of considerations related to their quality. In this way, the algorithm focuses on only those solutions that are more likely to increase the quality of the current working set, speeding up the whole search process.

A modified version of the *Crowding Distance Assignment* (CDA) procedure, originally presented in (Deb, 2002), has been developed in order to carry out the selection process. This procedure first divides the working set into several dominance levels. Each solution of one level strictly dominates all the solutions in the next level. The CDA then assigns to each solution a value (*Crowding Distance*) dependent on the normalized Euclidean distances between it and the solutions that precede and follow it in the same dominance level. The main difference resides in the fact that the modified procedure considers the number of times each solution has been already selected in previous iterations (*Selection Counter*), and uses this information to calculate the Modified Crowding Distance (MCD). The element with the highest value of MCD is selected as the starting point for the Greedy or local search phases.

The aim of this MCD procedure is to select a candidate solution belonging to a less crowded region of the Pareto front and at the same time has already been selected a small number of times. The use of such an operator demonstrated, in preliminary experiments, to significantly improve the Pareto front in terms of quality and spread of its solutions. The pseudocode of this procedure is presented in figure 1a.

3.2 Greedy phase

This is the main and most innovative part of the algorithm even though the original IG structure with two phases: *Destruction* and *Construction* respectively, is preserved. However, this greedy phase in the RIPG is radically different from the original IG where only one partial solution is maintained and a NEH-like greedy heuristic is applied in one unique step at each iteration of the algorithm. In our case, the Greedy phase becomes an iterative process, that works with a set of partial solutions and returns a set of non-dominated permutations.

The *Destruction* step chooses a random starting position and a block of k consecutive elements (jobs) are removed from the selected solution. Note that in the original IG algorithm, the removal

of jobs is not carried out in blocks.
For the *Construction* step, a variation of the NEH insertion scheme is used. The main difference from that heuristic lies in the use of Pareto dominance to maintain not just one incomplete partial solution at each iteration (as in NEH), but a whole set of non-dominated partial solutions generated during the insertion process. Actually, the *Construction* procedure inserts, one by one, all removed elements from the block back into each partial solution from the non-dominated partial solution set. This inserting schema was already effectively used in Arroyo and Armentano (2004). At each step, a new set of partial solutions is generated. More specifically, let n be the length of the initial solution and k the size of the block of removed elements. At the end of the first step, after the first removed element is inserted into all positions of the partial solution, we have $n - k + 1$ partial solutions of length $n - k + 1$. In the second iteration, the procedure inserts the second removed element in all positions of each one of the $n - k + 1$ partial solutions generated in the previous step. Then, at the end of this second iteration, the number of partial solutions is: $(n-k+1) \times (n-k+2)$. Following the same reasoning, at the end of the construction phase, a set of $\prod_{i=1}^{k} (n - k + i)$ of complete solutions is generated. This defines an upper bound for the number of solutions generated by the greedy phase of the algorithm. Regardless of this, the bound is very far from being tight because, at each iteration, all the dominated incomplete sequences are removed. The greedy phase therefore returns a set of non-dominated solutions, which is added to the current working set, and then, dominated solutions are removed. Finally, the MCD selection procedure is applied to the current working set and a solution is selected to be processed by the *local search* phase, which is explained next.

3.3 Local search phase

A simple and fast local search procedure has been demonstrated to be very helpful in improving the quality of solutions in the single as well as in the multi-objective cases. Hence, we added to our algorithm a simple local search phase aimed at refining the work of the greedy phase. In the original IG, the local search procedure uses as an input the outcome of the greedy phase. For the multi-objective case, the greedy phase returns a set of non-dominated elements, and it is likely that the current working set changes after adding it. To better tackle the multi-objective nature of the problem, thereby, it is not trivial to decide which solutions undergo the local search procedure. Therefore, we decided to entail once more the selection procedure previously described to choose a solution from the current working set for feeding the local search phase.
In order to maintain the algorithm as simple and fast as possible we focused our effort in obtaining a simple and fast local search procedure. The rationale of this phase is quite straightforward: n_{sel} elements belonging to the selected solution are randomly chosen, removed and re-inserted into n_{neigh} consecutive positions, half of which usually precede and half follow the original position of the element. In fact, depending on the distance of the original position from the beginning or from the end of the sequence, the neighborhood, still having the same amount of moves, may or

may not be symmetric. Local search in a multi-objective setting is not as simple as one might think. As a matter of fact, the concept of first improvement or best improvement does not directly apply. In our case, all movements, i.e., $n_{sel} \times n_{neigh}$ insertions are carried out and all solutions are evaluated. Afterwards, Pareto dominance is checked and a final non-dominated set is generated as a result. Note that is is much faster than generating, one by one, all neighbors and checking each time for dominance. In order to further speed up this local search, we employ the well known accelerations of Taillard (1990). Finally, this set is included into the working set and dominance is applied again. During the initial design phase we studied the algorithm performance by varying n_{sel} and n_{neigh}. We obtained the best results for $n_{neigh} = 5$ and for n_{sel} by dynamically changing its value with the n_{count} value (*Selection Counter* introduced in section 3.1) of the selected solution, in accordance with the following formula:

$$n_{sel} = \begin{cases} n_{count} & \text{if } n_{count} \leq n/2 \\ n/2 & \text{otherwise} \end{cases}$$

The pseudocode of the local search procedure is presented in figure 1b.

3.4 Restart phase

IG methods have one main drawback: they are prone to get stuck in local optimum solutions. The reason lies behind their very nature as they are greedy methods. RIPG is no different. To avoid this potential problem, we have included a simple, yet reliable restart phase. This procedure merely consists of storing all the elements of the current working set in a separate archive and then creating a new random working set of 100 elements. The main advantage of this restart procedure is that it is a very fast way to introduce diversification inside our metaheuristic scheme, whereas its main inconvenience consists of the difficulty in choosing of a suitable restarting criterion. The general idea is to execute a restart when the working set has not been changed during a sufficiently large number of iterations. To accomplish with our objective of simplicity and reliability, we use the simple approach of checking whether the size of the current working set changes after each iteration. Of course, this strategy is sometimes inaccurate because it can not detect a change in the working set that does not affect its cardinality, but has the unquestioned advantage of being very fast. Another important issue to take into account while designing an effective restart is to establish the minimum number of iterations to safely assert that the search process is in a stalemate. In fact, if the restart condition is not carefully designed, it might either be applied too often, thus preventing reaching a steady state in the search or be too seldom applied, wasting, in this way, valuable CPU time. We carried out short initial tests considering several fixed as well as dynamic restarting points and the best results were achieved by applying a restart after $n \times 2$ iterations without changes in the cardinality of the current working set. Note that this rule is effective since it takes into account the size of the instance allowing for more iterations to bigger ones for which the algorithm needs a larger amount of time to reach a steady state.

```
**Modified-**
% $PS$ is the
$SetSize :=|$

for all $i :=$

    $PS[i]_{dist.}$

for all objec

    $PS := so$

    for all

        $PS[i]_{d}$

        % whe
values for th

for all $i :=$

    if $PS[i]$

        $PS[i]_{d}$

for all $i :=$

    $PS[i]_{dist.}$

    $PS[i]_{dista}$
```

(a) *Modified*

4 Experimental phase

This section is aimed at introducing the reader to all the elements needed in order to fully understand the experiments carried out, the results and their implications. First we deal with the description of the state-of-the-art algorithms the RIPG is compared to within the experimental phase. Later on, we describe in detail the test bed instance sets used and discuss the performance assessment methodologies considered. Then, the design process of our proposed algorithm is described and ultimately, we describe the test campaign and analyze the results.

4.1 Adaptation of existing metaheuristics

In the previously cited review work of Minella et al. (2008), we carried out a comprehensive analysis of the performances of the most well known multi-objective algorithms. We carefully re-implemented 23 algorithms and made an exhaustive test campaign with three couples of objectives, three different stopping times and a large set of instances expressly created for the multi-objective flowshop problem. Recall that in this paper we extend this problem with the presence of setup times. A preliminary experiment in which all the algorithms have been tested using a reduced set of instances has been carried out and on the basis of the results obtained we have selected the best algorithms. Many of these are specifically designed to tackle flowshop problems without setups while the other three are general purpose multi-criteria optimization procedures. Note that, to the best of our knowledge, no multi-objective methods have been specifically proposed for the setup times flowshop with Pareto approach. The generic methods are now briefly explained. A genetic algorithm is proposed by Corne et al. (2000). This method, called PESA uses a selection and replacement procedure based on a crowding measure. 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.
Kollat and Reed (2005) proposed a variation of NSGAII proposed by Deb (2002) referred to as $\varepsilon-$NSGAII by adding $\varepsilon-$dominance archiving and adaptive population sizing. The reader can find a more detailed description of these algorithms in Minella et al. (2008).
We decided to include another seven algorithms recently presented in the literature that did not make it for Minella et al. (2008) evaluation. Five of them have been modified in order to use up all the available time unlike the original versions which ended after a fixed number of iterations. Those algorithms are highlighted by adding a "_M" at the end of the name. Note that one of them, a modified version of the multi-objective simulating annealing of Varadharajan and Rajendran (2005), referred to as MOSA_Varad_M has been already presented in Minella et al. (2011) where it was shown to clearly outperform the original version. For space reasons we included at `http://soa.upv.es` as on-line material the pseudo-codes of the original as well as the modified versions of these algorithms. The selected algorithms, seventeen in total, either specific for the

PFSP or for the general multi-objective version, are summarized in Table 1. Table 2 shows the values for the parameters used for all the algorithms in the comparison. Those values are those reported in their respective original works.

Acronym	Year	Author/s	Type
MOGA_Murata	1996	Murata et al.	Genetic algorithm. Specific
PESA	2000	Corne et al.	Genetic algorithm. General
PESAII	2001	Corne et al.	Genetic algorithm. General
CMOGA	2001	Murata et al.	Genetic algorithm. 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_Varad	2005	Varadharajan and Rajendran	Simulated annealing. Specific
PGA_ALS	2006	Pasupathy et al.	Genetic algorithm. Specific
PILS	2007	Geiger	Iterated local search. Specific
hMGA	2007	Yanda and Tamura	Genetic algorithm. Specific
MOIGS	2008	Framinan and Leisten	Iterated Greedy. Specific
MOACA17_M	2009	Rajendran and Ziegler	Ant-colony algorithm. Specific.
MOACA18_M	2009	Rajendran and Ziegler	Ant-colony algorithm. Specific.
MOACA19_M	2009	Rajendran and Ziegler	Ant-colony algorithm. Specific.
MOACA20_M	2009	Rajendran and Ziegler	Ant-colony algorithm. Specific.
MOSA_Varad_M	2011	Minella et al.	MOSA_Varad improved version

Table 1: Re-implemented methods for the SDST multi-objective flowshop.

Algorithm	Parameter	Value
MOSA_Varad_M	Non Dominated Solutions Archive Size	500
	Initial Temperature	575
	Final Temperature	when time is over
	Ratio of Temperature Decreasing	0.9
MOSA_Varad	Non Dominated Solutions Archive Size	500
	Initial Temperature	575
	Final Temperature	20
	Ratio of Temperature Decreasing	0.9
MOTS	Population Size	10
PILS	Number of environments	3
MOIGS	Destruction Block Size	7
PESA	Population Size	100
	Crossover Probability	0.7
	Mutation Probability	1/Number of jobs
	Non Dominated Solutions Archive Size	100
	Number of Divisions in the Grid	32
PESAII	Population Size	100
	Crossver Probability	0.7
	Mutation Probability	1/Number of jobs
	Cardinality of the Non Dominated Solutions set	100
	Number of Divisions in the Grid	32
MOGALS_Arroyo	Number of elite solutions	20
	Population size	100
	Crossover probability	0.9
	Mutation probability	0.6
	Max Paths	10
	Max Iterations of Local Search	15
	Iterations between each 2 search executions	100
PGA_ALS	Population size	100
	Crossver probability	1
	Mutation probability	0.1
MOGA_MURATA	Number of elite solutions	5
	Population size	100

Algorithm	Parameter	Value
	Crossover probability	0.9
	Mutation probability	0.1
$\varepsilon-$NSGAII	Population	10
	Crossover probability	1
	Mutation probability	1/Number of jobs
	Epsilon value	0.001
CMOGA	Number of elite solutions	3
	Population size	101
	Crossover probability	0.8
	Mutation probability	0.3
	Maximum cell distance	20
hMGA	Population size	100
	Crossover probability	0.9
	Mutation probability	0.01
MOACA17_M	ρ	0.7
	C	2
	P	1.5
	K	50
	Population size	500
MOACA18_M	ρ	0.7
	C	2
	P	1.5
	K	20
	Population size	500
MOACA19_M	ρ	0.7
	C	2
	P	1.5
	K	50
	Population size	500
MOACA20_M	ρ	0.7
	C	2
	P	1.5
	K	20
	Population size	500

Table 2: Table of parameter values for the re-implemented methods.

4.2 Benchmark description

In the experiments presented in this paper we make use of three different instance sets based on the original work of Taillard (1993) for the regular flowshop and on the papers of Ruiz et al. (2005) and Ruiz and Stützle (2008) for the setup times version.

Each set contains instances with several combinations for the number of jobs (n) and number of machines (m). The $n \times m$ combinations are: $\{20, 50, 100\} \times \{5, 10, 20\}$ and $200 \times \{10, 20\}$ for a total of 11 different groups of instances. The first two sets, referred to as **SSD50** and **SSD125**, respectively, have ten instances for each group, resulting in a total of 110 instances per set. Setup times in SSD50 and SSD125 are generated to be respectively the 50% and the 125% of the processing times (p_{ij}). This means that since the p_{ij} in Taillard's instances are generated using a uniform distribution in the range $[0-99]$ in the first set, setup times are uniformly distributed in the range $[0-49]$, while in the second, their range is $[0-124]$. Each instance is assigned with a set of weights and due dates. The weights are drawn from a uniform $U[1, 10]$ distribution while the due dates d_j are generated by means of 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 jobs $j \in N$ and $random$ is a random number uniformly distributed in $[0, 1]$. Finally, we created from scratch a third set of instances, referred to as **SSDTest**, borrowing the structure from the first two. This set is used to calibrate our algorithm and contains only four instances for each group, half of which have setup times in $[0 - 49]$ and half in $[0 - 124]$. All benchmarks as well as all solutions obtained in this paper are available at `http://soa.upv.es`.

4.3 Performance assessment methodologies

In single objective optimization, the concept of a better solution is straightforward. However, when multiple objectives are present, deciding when a given result A is better than B is far from simple. As mentioned, the outcome of a multi-objective optimizer is not just one solution, but a set of several different solutions. Usually, the outcome is already processed so that only non-dominated solutions are given. This is commonly referred to as a *frontier*. It is obvious, of course, that a frontier is better than another if all the points of the former (strongly) dominate all the points of the latter. In the case that this condition does not hold, an unambiguous way to establish which frontier outperforms the other does not exist to the best of our knowledge.

Many indices have been introduced in the literature to overcome this impasse but this is still an open issue so much so that, recently, first Knowles et al. (2006) and later Zitzler et al. (2008) demonstrated that many frequently used metrics are non Pareto-compliant i.e., in some cases, they can assign a better value to a Pareto frontier B respect to frontier A even if A dominates B. Those metrics therefore, can often give wrong and misleading results. Therefore, special attention must be given to the choice of quality measures to ensure fair and generalizable results.

In this paper, we choose the hypervolume I_H and the unary Epsilon I^1_ε indicators that Knowles et al. (2006) and Zitzler et al. (2008) demonstrated as being Pareto-compliant and, at the moment, can be considered as the state-of-the-art indicators for the evaluation of multi-objective algorithms. Additionally, considering two quality indicators instead of one, increases the soundness of our conclusions. Given two multi-objective optimizers, these two indicators can, sometimes, give conflicting results, indicating that neither one can be considered superior.

The hypervolume indicator I_H, first introduced by Zitzler and Thiele (1999) measures the normalized (hyper)volume of the solution space dominated by the Pareto front approximation generated by one algorithm. A reference point is needed for closing the hypervolume. We obtained this reference point by considering the worst value for each objective over the whole set of Pareto front generated by all the methods for a certain instance and multiplying them by 20%. More formally, the I_H indicator can be defined as follows: Given a set of Pareto frontiers $\mathcal{F}$, being $F \in \mathcal{F}$ a frontier, $pt \in F$ a point belonging to the frontier, $NObj$ the number of objectives and $NSol$ the number of points in F, then the hypervolume for F can be calculated as $I_H(F) = \sum_{1 \leq i \leq NSol} \sum_{1 \leq j \leq NObj} \frac{pt_{i,j} - 1.2 \cdot \min_j \mathcal{F}}{1.2 \cdot (\max_j \mathcal{F} - \min_j \mathcal{F})}$, where $\min_j \mathcal{F}$ and $\max_j \mathcal{F}$ are the best and worst values, respectively, for objective j over all the frontiers in $\mathcal{F}$. Notice that, as the objective values are normalised, the maximum I_H value can by obtained by the product of the reference point values: $1.2 \times 1.2 = 1.44$.

As far as the Unary Epsilon Indicator I^1_ε is concerned, it was proposed initially by Zitzler et al. (2003) and was later extended in Knowles et al. (2006) and in Zitzler et al. (2008). It measures the minimum distance between a given Pareto front and the optimal one or an approximation of it. Since for our problem the optimal front for each instance is not known, a reference set, constituted by gathering all non-dominated solutions obtained by all the tested algorithms, is used. As said, the objectives values are normalized and additionally translated by one unit in order to avoid division by zero errors in the calculation of the indicator. This approach was used with success by Minella et al. (2008) and by Minella et al. (2011). In this way the indicator varies between 1 and 2. A value close to 1 means that the considered frontier is close to the reference set, whereas a value close to 2 means that the set of solutions is distant. It is formally calculated as follows. P is the Pareto front or a reference set and S is an approximation to the Pareto front. Actually, $I^1_\varepsilon = I_\varepsilon(S, P)$ where $I_\varepsilon(S, P) = \max_{x^2} \min_{x^1} \max_j \frac{f_j(x^1)}{f_j x^2}$.

The main drawback of using unary indicators when comparing different frontiers is that such measures cannot supply us with information about the spatial behavior of the considered algorithms, i.e., in which region of the objectives space one algorithms behaves better or worse than the other. This knowledge is especially important during the design phase of a new method. In fact, having at our disposal a technique that permits us to analyze the outcomes and highlights potential problems of convergence would highly simplify and accelerate the development processes. Fonseca et al. (2001) proposed a probabilistic measure, called *Attainment Function*, which is able to provide useful insight into the spatial behavior of an algorithm. To be more precise, let $\mathcal{X} \in \mathbb{R}^d$ be an arbitrary point of a d-objective solution space and $\mathcal{F} = \{v_p \in \mathbb{R}^d, p = 1, \ldots, P\}$ be a frontier generated in a single run of and algorithm referred to as α. P is the total number of points present at the frontier F. The attainment function is defined by $AF_\alpha(\mathcal{X}) = P(\exists\ v_p \in \mathcal{F} \colon v_p \trianglelefteq \mathcal{X})$ which describes the probability the method α has to generate, in a single run, a Pareto front approximation $\mathcal{F}$ in which at least one element weakly dominates ($\trianglelefteq$) the arbitrary point $\mathcal{X}$. In the case of stochastic algorithms, it is not possible to express this function in a closed form but it can be empirically approximated by employing the outcomes of several algorithm runs. This approximation is called *Empirical Attainment Function* and is defined in Fonseca et al. (2001) as follows: $EAF(\mathcal{X}) = \frac{1}{q} \sum_{h=1}^{q} I(\mathcal{F}_h \trianglelefteq \mathcal{X})$ where $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_q$ represent q Pareto set approximations obtained in q independent algorithm's runs and

$$I(\mathcal{F}_h \trianglelefteq \mathcal{X}) = \begin{cases} 1 & \text{if } \mathcal{F}_h \trianglelefteq \mathcal{X} \\ 0 & \text{otherwise} \end{cases}$$

The attainment function has one inconvenience. The problem is that it has to be calculated for each pair of compared methods. When the number of evaluated methods grows, the number of pairwise comparisons grows quadratically. In order to overcome this drawback, López-Ibáñez et al. (2006) presented the Diff-EAF that is a probability function defined as the difference between two EAFs. Let α and β be two algorithms. The Diff-EAF is then defined as follows:

$$Diff - EAF_{(\alpha,\beta)}(\mathcal{X}) = \frac{1}{q}\sum_{h=1}^{q}\left[I(\mathcal{F}_h^{\alpha} \trianglelefteq \mathcal{X}) - I(\mathcal{F}_h^{\beta} \trianglelefteq \mathcal{X})\right]$$

This function represents the probability of $\mathcal{X}$ to be dominated by a frontier of α but not of β. Note that the $Diff - EAF_{(\alpha,\beta)}$ may have positive as well as negative values. A positive value indicates that the algorithm α prevails over β in $\mathcal{X}$, the other way around in case of a negative value.

Finally, another important issue related to the comparison among algorithms is the choice of a suitable termination criterion. We believe that the fairest way to confront different methods is assigning the same amount of CPU time to each one of them and more time to larger instances with respect to smaller ones. Therefore, we assign to each algorithm the same elapsed CPU time limit that depends on the size of the considered instance. The algorithms are then stopped after $n \cdot m/2 \cdot t$ milliseconds of CPU time, where t is an input parameter that will be tested at different values. In this way, we assign more time to larger instances that are obviously more time consuming to solve.

4.4 Design and calibration of the RIPG

The RIPG is the result of an thorough engineering process in which all parts that constitute it have been compared against several possible alternatives. The aim is to create an efficient and effective algorithm. To do this we employ of a sound statistical methodology called Design of Experiments (DoE) (see for more details Montgomery, 2009). In order to prevent a possible over-calibration of our proposed algorithm, we decided to use, in this phase, an instance set (**SSDTest**) different to those used in the comparison of RIPG against other algorithms (**SSD50** and **SSD125**). Initially, we studied the influence of the local search and the restart mechanism on the quality of the produced solutions. Thereby, we assembled 4 algorithms called *IPG_B*, *IPG_LS*, *RIPG_B* and *RIPG_LS*, respectively. *B* and *LS* stand for *Basic* and *Local Search* and the prefix *R* indicates the presence of the restart mechanism. Each algorithm was run ten independent times (replicates) against all 44 instances of the SSDTest set and both hypervolume and epsilon indicators are calculated, leading to a total of $4 \times 44 \times 10 = 1760$ results.

Only the $(C_{\max}, TWT)$ pair of objectives was considered for the calibration and the stopping time was fixed at $t = 100ms$. Figures 2a and 2b depict the means plots with Tukey confidence intervals with a 95% confidence level ($\alpha = 0.05$) from the ANOVA test and report on the interaction between the response variable, respectively I_H and I_ε^1, and the type of algorithm which is tested as a controlled factor with four levels. Due to reasons of space, we only highlight here the most important findings. Overall, the method with restart and local search phases (*RIPG_LS*) achieves better average results, that are, even for the grand average, statistically better than the non-restart versions, with and without local search. The local search phase improves the quality of the results for all the instances while the restart phase only affects the smaller ones. This behavior is mainly attributable to the huge number of elements in the solution space that prevents the algorithm from reaching a steady state in the available computation time. On the contrary, small instances have Pareto fronts with few solutions and the search soon gets trapped. In this scenario, the restart procedure operates by increasing the diversification in the process, thus enhancing the probability of improving the solutions generated.

Figure 2: Means plots and Tukey HSD confidence intervals ($\alpha = 0.05$) in the ANOVA test for the calibration of the RIPG. Makespan and total weighted tardiness criteria and $t = 100ms$ CPU time stopping criterion.

4.5 Computational analysis

In this section we detail the campaign of experiments we have carried out and analyze the outcomes by means of statistical tests. We run the 18 tested algorithms ten times each (replicates) for each one of the 220 instances of the sets SSD50 and SSD125, for both pairs of criteria. Furthermore, two stopping times have been considered ($t = 150ms$ and $t = 200ms$) raising up the total number of data samples to 158,400. All the tested methods are coded in Delphi XE language with all the optimization options activated. The experiments have been executed on a cluster of 12 Core Duo 2.4 Ghz computers running with Windows$^{®}$ XP SP3 O.S. and 2 GBytes of RAM memory. The results are summarized in Table 3, for the $C_{\max} - TWT$ case and Table 4 for $C_{\max} - TFT$. Although each cell in the tables is the average of no less than 2200 data points, it is still necessary to carry out a careful statistical experiment in order to assess whether the observed differences in the average values are statistically meaningful. We did parametric ANOVA analyses as well as non-parametric Friedman rank-based tests for both instance sets, for the two performance indicators, the two pairs of objectives and for the two stopping times. This results in a total of 32 different experiments. Thereby, we carried out 16 multi-factor ANOVAS where the size of the instance and the algorithm are the controlled factors. One half of the experiments have been carried out employing the hypervolume as a response variable, and then we repeated the same tests considering the epsilon indicator. All the tests have been executed with a confidence level of 95% ($\alpha = 0.05$). Note that since we are carrying out four tests over the same results (parametric and non-parametric, epsilon and hypervolume), we employ the Bonferroni adjustment for the α level, i.e., we use an adjusted α_s of 0.01 for a real α of 0.05. In order to safely apply ANOVA, it is necessary to check three main hypothesis: normality, homogeneity of variance (or homoscedasticity) and independence of residuals. The residuals resulting from the experimental data have been analyzed and all three hypothesis can be accepted.

We make use of parametric as well as non-parametric tests to strengthen the soundness of our conclusions. Therefore we compare the results of the first group of tests against those of a second group of non-parametric experiments. Non-parametric Friedman rank-based tests have been carried out. Since there are 18 algorithms and 10 different replicates, the results for each instance are ranked between 1 and 180. All these tests substantially validate and strengthen the results shown in the tables. In table 3, the mean values concerning both instance sets and $C_{\max} - TWT$ criteria are reported. Both indicators I_H and I_ε^1 are considered and the methods are sorted decreasingly according to the value of I_H. RIPG turns out to be the best performing algorithm for each combination of indicators, stopping time and instance set. In comparison to MOSA_Varad_M, the second method in the ranking, the RIPG's percentage improvement is between 2% and 7% of the hypervolume.

Please note that the global ranking of algorithms remains substantially unchanged for the different stopping times and instance sets, whereas there are few differences if we compare the ranking of I_H and I_ε^1. This is mainly due to the fact that the unary epsilon indicator is the average minimum distance between the frontier generated in a single run and the best Pareto front known. Also it is much more conservative than the hypervolume, therefore it is likely that by using this indicator, closer values are assigned to the algorithms and there is the possibility that they might have a different rank. Since at the moment it is not clear which indicator is the most reliable, when this anomaly happens, the affected methods are considered incomparable. It is worth to note that RIPG turns out to be more competitive in the SSD125 set respect to SSD50.

In Table 4 results concerning $C_{\max} - TFT$ criteria are reported and again RIPG comes out as being the best method in the set, but unlike the $C_{\max} - TWT$ case, MOIGS performs much better and achieves the second position of the rank for the SSD125 set. This happens because it is an improved version of an algorithm specially designed to tackle the multi-objective flowshop problem with $C_{\max} - TFT$ criteria.

Note also how Tables 3 and 4, contain additional columns with the total number of evaluated solutions for each algorithm, combination of objectives, instance set and termination criterion. It is interesting to see that there are enormous differences. Take for example algorithms PILS and MOSA_Varad_M for instance set SSD50 and termination criterion 150ms. PILS evaluated about 426 thousand solutions on average whereas MOSA_Varad_M evaluated more than 3.5 million. This is, on average, MOSA_Varad_M evaluates almost 8.5 more solutions than PILS for the same CPU time. This highlights the importance of stopping algorithms after the same elapsed CPU time and not after the same number of iterations. Stopping both algorithms after the same number of iterations would result in wildly different employed CPU times.

Figures 3a and 3b show some means plots for ANOVA and Friedman tests where only the first five best algorithms are depicted. It has to be stressed that ranking tests neglect the real differences in the indicators and therefore, differences may appear smaller of greater respect to the equivalent ANOVA test.

Due to reasons of limited space, we cannot reproduce here the complete 32 plots, each one with all the 14 algorithms. These are available as part of the on-line material.

Lastly, we present six figures (4a-4f) which represent the EAF for MOSA_Varad_M (4a), MOIGS (4b) and RIPG (4c) and differences of EAFs (4d,4e,4f) for instance 71 of SSD50 (100 jobs and 10 machines) and ($C_{\max} - TWT$) criteria. Each image is elaborated employing 50 replicates for each algorithm. Although these pictures give us information only for a single instance, during a design phase one can use such knowledge to understand by and large the behavior of the involved algorithms. Let us focus for example on picture 4a, it is clear that MOSA_Varad_M method is worse than MOIGS only in a central part of the objective space while it achieves better results than MOIGS at the extremes of the frontier (see figure 4d). Finally, also using this tool we confirm that RIPG widely outperforms its competitors. For more details, the reader is referred to the on-line material where similar figures are reported for other instances.

(a) *Hypervolu*

Figure 3: Mean
SSD50

	1
Method	I
RIPG	1.2
MOSA_Varad_M	1.2
MOSA_Varad	1.2
MOIGS	1.1
MOGALS_Arroyo	1.1
MOTS	1.1
PESAII	1.1
PESA	1.1
MOACA17_M	1.0
MOACA18_M	1.0
MOACA19_M	1.0
MOACA20_M	1.0
PGA_ALS	1.0
MOGA_Murata	0.9
$\varepsilon-$NSGAII	0.9
CMOGA	0.8
hMGA	0.8
PILS	0.7

	1
Method	I
RIPG	1.3
MOSA_Varad_M	1.2
MOIGS	1.1
MOSA_Varad	1.1
MOGALS_Arroyo	1.1
MOTS	1.0
PESA	1.0
PESAII	1.0
MOACA18_M	0.9
MOACA17_M	0.9
MOACA20_M	0.9
MOACA19_M	0.9
PGA_ALS	0.9
MOGA_Murata	0.8
$\varepsilon-$NSGAII	0.7
CMOGA	0.7
PILS	0.6
hMGA	0.5

Figure 4: Empi
A single instanc

5 Conclusions and future research

There have been several methods proposed in the literature for the a posteriori multi-objective flowshop problem. However, as important as they are in practice, setup times have not been considered, as far as we know, for this setting. This paper represents a first attempt to tackle this problem.

We have presented two main contributions to the field of the multi-objective flowshop. First, we have adapted the best performing algorithms for the multi-objective flowshop by adding anticipative sequence dependent setup times to the problem. We carried out a study of these algorithms to establish which ones show better performance for the problem with setups. Second, we have extended a new strategy which achieved state-of-the-art results for the single objective flowshop, the Iterated Greedy metaheuristic, in order to deal with several objectives and setup times simultaneously. The extended IG method has been referred to as RIPG. A thorough algorithm engineering process, along with statistical calibration led to a refined proposal which has shown to reach state-of-the-art results for this problem. This has been confirmed by a wide campaign of tests where the results have been analyzed by means of parametric as well as non-parametric statistical tests. We employed two Pareto compliant performance indicators, two stopping criteria based on the elapsed CPU time and two combinations of scheduling objectives. As a consequence, RIPG can be considered the state-of-art procedure for this scheduling problem. Future research lines stem from the possibility of applying this scheme to solve different or more constrained scheduling problems such as, the hybrid flowshop or the parallel machines problems. Another interesting area of future research relates to the EAF, a rather new statistical tool employed here, to study the probability that one algorithm has to cover a certain zone of the objective space. Its main drawback is that only one instance at a time is represented. An extension of the EAF that takes into account a whole instance set would be a useful tool for a deeper understanding of algorithm's behavior and, furthermore, it would also lead to new and more reliable performance indicators.

Acknowledgments

The authors would like to thank the anonymous referees for their careful and detailed comments which have helped improve this manuscript considerably. This work is partially financed by the Spanish Ministry of Science and Innovation, under the projects "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theorerical Advances" with reference DPI2008-03511/DPI and "RESULT - Realistic Extended Scheduling Using Light Techniques" with reference DPI2012-36243-C02-01 and by the Small and Medium Industry of the Generalitat Valenciana (IMPIVA) and by the European Union through the European Regional Development Fund (FEDER) inside the R+D program "Ayudas dirigidas a Institutos Tecnológicos de la Red IMPIVA" during the year 2011, with project numbers IMDEEA/2011/142 and IMDEEA/2012/143.

References

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

Armentano, V. A. and Arroyo, J. E. C. (2004). An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, 10(5):463–481.

Arroyo, J. E. C. and Armentano, V. A. (2004). A partial enumeration heuristic for multi-objective flowshop scheduling problems. *Journal of the Operational Research Society*, 55(9):1000–1007.

Arroyo, J. E. C. and Armentano, V. A. (2005). Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167(3):717–738.

Bagchi, T. P. (2001). Pareto-optimal solutions for multi-objective production scheduling problems. In Zitzler, E., Deb, K., Thiele, L., Coello Coello, C. A., and Corne, D., editors, *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland, March 7-9, 2001, Proceedings*, volume 1993 of *Lecture Notes in Computer Science*, pages 458–471. Springer.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H. M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, San Francisco, California, USA. Morgan Kaufmann.

Corne, D. W., Knowles, J. D., and Oates, M. J. (2000). The pareto envelope-based selection algorithm for multiobjective optimization. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo Guervós, J. J., and Schwefel, H. P., editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 839–848. Springer.

Deb, K. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Fonseca, V. G. D., Fonseca, C. M., and Hall, A. O. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. In *In Evolutionary Multi-Criterion Optimization, First International Conference*, volume 1993 of *Lecture Notes in Computer Science*, pages 213–225. Springer-Verlag.

Framinan, J. and Leisten, R. (2008). A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum*, 30(18):787–804.

Framinan, J. M., Leisten, R., and Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, 141(3):559–569.

Garey, M., Johnson, D., and Sethi, R. (1976). The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research*, 1:117–129.

Geiger, M. J. (2007). On operators and search space topology in multi-objective flow shop scheduling. *European Journal of Operational Research*, 181(1):195–206.

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

Ishibuchi, H. and Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems Man and Cybernetics*, 28(3):392–403.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Jacobs, L. W. and Brusco, M. J. (1995). A local-search heuristic for large set-covering problems. *Naval Research Logistics*, 42(7):1129–1140.

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

Kollat, J. B. and Reed, P. M. (2005). The value of online adaptive search: A performance comparison of nsgaii, $\varepsilon-$nsgaii and $\varepsilon-$moea. In Coello Coello, C. A., Hernández Aguirre, A., and Zitzler, E., editors, *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, volume 3410 of *Lecture Notes in Computer Science*, pages 386–398. Springer.

Loukil, T., Teghem, J., and Fortemps, P. (2000). Solving multi-objective production scheduling problems with tabu search. *Control and Cybernetics*, 29(3):819–828.

Loukil, T., Teghem, J., and Tuyttens, D. (2005). Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, 161(1):42–61.

López-Ibáñez, M., Paquete, L., and Stützle, T. (2006). Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137.

Minella, G., Ruiz, R., and Ciavotta, M. (2008). A review and evaluation of multi-objective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471.

Minella, G., Ruiz, R., and Ciavotta, M. (2011). Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11):1521–1533.

Montgomery, D. C. (2009). *Design and Analysis of Experiments.* Wiley, New York, seventh edition.

Murata, T., Ishibuchi, H., and Gen, M. (2001). Specification of genetic search directions in cellular multi-objective genetic algorithms. In Zitzler, E., Deb, K., Thiele, L., Coello Coello, C. A., and Corne, D., editors, *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland, March 7-9, 2001, Proceedings*, volume 1993 of *Lecture Notes in Computer Science*, pages 82–95. Springer.

Murata, T., Ishibuchi, H., and Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957–968.

Nawaz, M., Jr, E. E. E., and Ham, I. (1983). A heuristic algorithm for the m machine, n job flowshop sequencing problem. *Omega-International Journal of Management Science*, 11(1):91–95.

Pan, Q.-K., Wang, L., and Zhao, B.-H. (2007). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *The International Journal of Advanced Manufacturing Technology*, 38(7-8):778–786.

Parthasarathy, S. and Rajendran, C. (1997a). An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs. *International Journal of Production Economics*, 49(3):255–263.

Parthasarathy, S. and Rajendran, C. (1997b). A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs - a case study. *Production Planning & Control*, 8(5):475–483.

Pasupathy, T., Rajendran, C., and Suresh, R. K. (2006). A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *The International Journal of Advanced Manufacturing Technology*, 27(7-8):804–815.

Rajendran, C. (1995). Heuristics for scheduling in flowshop with multiple objectives. *European Journal of Operational Research*, 82(3):540–555.

Rajendran, C. and Ziegler, H. (1997). A heuristic for scheduling to minimize the sum of weighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. *Computers & Industrial Engineering*, 33(1-2):281–284.

Rajendran, C. and Ziegler, H. (2003). Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. *European Journal of Operational Research*, 149(3):513–522.

Rajendran, C. and Ziegler, H. (2009). A multi-objective ant-colony algorithm for permutation flowshop scheduling to minimize the makespan and total flowtime of jobs. In Chakraborty, U., editor, *Computational Intelligence in Flow Shop and Job Shop Scheduling*, volume 230 of *Studies in Computational Intelligence*, chapter 3, pages 53–99. Springer, Berlin / Heidelberg.

Ríos-Mercado, R. Z. and Bard, J. F. (1998). Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Computers & Operations Research*, 25(5):351–366.

Ríos-Mercado, R. Z. and Bard, J. F. (1999). An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. *Journal of Heuristics*, 5(1):53–70.

Ríos-Mercado, R. Z. and Bard, J. F. (2003). The flow shop scheduling polyhedron with setup times. *Journal of Combinatorial Optimization*, 7(3):291–318.

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

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033 – 2049.

Ruiz, R. and Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143 – 1159.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159:139–171.

Simons, Jr, J. V. (1992). Heuristics in flow shop scheduling with sequence dependent setup times. *Omega-International Journal of Management Science*, 20(2):215–225.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Suresh, R. K. and Mohanasundaram, K. M. (2004). Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives. In *IEEE Conference on Cybernetics and Intelligent Systems (CIS), Singapore, December 1-3, 2004, Proceedings*, volume 2, pages 712–717.

Taillard, E. (1990). Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

T'kindt, V. and Billaut, J.-C. (2006). *Multicriteria scheduling: Theory, models and algorithms.*

Springer, Berlin, second edition.

Varadharajan, T. and Rajendran, C. (2005). A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European Journal of Operational Research*, 167(3):772–795.

Yanda and Tamura, H. (2007). A new multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems. *International journal of computer integrated manufacturing*, 20(5):465–477.

Ying, K.-Y. (2009). An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks. *Journal of the Operational Research Society*, 60(6):810–818.

Zitzler, E., Knowles, J., and Thiele, L. (2008). Quality assessment of pareto set approximations. In Branke, J., Deb, K., Miettinen, K., and Słowinski, R., editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*, pages 373–404, Berlin, Heidelberg. Springer-Verlag.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE, Transactions on Evolutionary Computation*, 7(2):117–132.