European Journal of Operational Research 240 (2015) 666–677

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

New hard benchmark for flowshop scheduling problems minimising makespan

Eva Vallada [a], Rubén Ruiz [a,*], Jose M. Framinan [b]

[a] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edificio 8G, Acc. B. Universitat Politécnica de València, Camino de Vera s/n, 46021 València, Spain

[b] Industrial Management School of Engineering, University of Seville, Av.Descubrimientos s/n, 41092 Seville, Spain

ARTICLE INFO

Article history:
Received 18 July 2013
Accepted 25 July 2014
Available online 12 August 2014

Keywords:
Flowshop
Makespan
Benchmark

ABSTRACT

In this work a new benchmark of hard instances for the permutation flowshop scheduling problem with the objective of minimising the makespan is proposed. The new benchmark consists of 240 large instances and 240 small instances with up to 800 jobs and 60 machines. One of the objectives of the work is to generate a benchmark which satisfies the desired characteristics of any benchmark: comprehensive, amenable for statistical analysis and discriminant when several algorithms are compared. An exhaustive experimental procedure is carried out in order to select the hard instances, generating thousands of instances and selecting the hardest ones from the point of view of a gap computed as the difference between very good upper and lower bounds for each instance. Extensive generation and computational experiments, which have taken almost six years of combined CPU time, demonstrate that the proposed benchmark is harder and with more discriminant power than the most common benchmark from the literature. Moreover, a website is developed for researchers in order to share sets of instances, best known solutions and lower bounds, etc. for any combinatorial optimisation problem.

1. Introduction

Advancements in algorithms in the field of operational research frequently require careful and comprehensive computational comparisons against well known and established benchmarks of instances. Once a standard set of problems is recognised as the de facto standard, different proposed techniques can be easily compared using this set. As per the recommendations of Beasley (1990), such benchmarks are nowadays shared easily through the Internet and the best known solutions (usually in the form of best known upper bounds in minimisation problems) are shared and used in order to compare presented algorithms against such bounds.

The importance of benchmarks cannot be overstated. A result is published only after showing better performance for a given problem in the standard benchmark accepted by peers most of the time. Therefore, the quality of the benchmark is of paramount importance. Poorly designed benchmarks might not be representative of real problems. Furthermore, other problems might arise. The set of instances might be of a limited size, too easy or specific for a given combination of input parameters. In such cases, if a given method outperforms another in the benchmark, it is not guaranteed that the performance can be generalised over the population of real instances.

One of the major fields in operational research is scheduling. This is recognised by Potts and Strusevich (2009) where it is stated that hundreds of papers are published per year in all relevant journals in the field. In scheduling, the pioneering work is the paper of Johnson (1954) where the famous two machine flowshop scheduling problem with makespan minimisation criterion was studied. Therefore, flowshop scheduling has been in the spotlight ever since. This prolific field is summarised in the reviews of Framinan, Gupta, and Leisten (2004); Ruiz and Maroto (2005); Hejazi and Saghafian (2005) or in Gupta and Stafford (2006). Reviews for other objectives apart from makespan are given in Vallada, Ruiz, and Minella (2008) for tardiness related criteria and in Pan and Ruiz (2013) for flowtime objectives. Literally hundreds of papers have been proposed just for the minimisation of the makespan in flowshop problems, even more if one considers all other studied objectives. This paper focuses specifically on the flowshop problem.

The most widely used benchmark for flowshop scheduling is that of Taillard (1993). There are other much less employed benchmarks, like the ones of Demirkol, Mehta, and Uzsoy (1998) or

* Corresponding author. Tel.: +34 96 387 70 07x74946; fax: +34 96 387 74 99.
E-mail addresses: evallada@eio.upv.es (E. Vallada), rruiz@eio.upv.es (R. Ruiz), framinan@us.es (J.M. Framinan).

http://dx.doi.org/10.1016/j.ejor.2014.07.033
0377-2217/© 2014 Elsevier B.V. All rights reserved.

Reeves (1995) or older benchmarks that are not currently being used, like the ones of Carlier (1978) and Heller (1960). Taillard's benchmark comprises 120 instances for the flowshop problem that range from 20 jobs and 5 machines all the way up to 500 jobs and 20 machines. At the time of writing, only 28 instances[1] in the benchmark are "open" meaning that the optimum solution has not yet been found. As we will show, several authors have recently been unable to assess outperformance in Taillard's benchmark due to several factors that we will later point out. Notable examples are Dong and Ping Chen (2008) and Kalczynski and Kamburowski (2008). These authors could not find statistically better performance using Taillard's benchmark and showed that using other randomly generated instances of their own, better performance was observed. In a sense, Taillard's benchmark is reaching exhaustion.

The previous potential problems, along with other shortcomings motivate this research. In this paper we present a new, computationally challenging and comprehensive benchmark for the flowshop scheduling problem with makespan criterion. First, we define the flowshop problem and study the existing literature in an attempt at characterising the hardness of flowshop instances in Section 2. Then, following a large computational campaign, we present the new benchmark in Section 3. Contrary to existing research, where benchmarks are simply presented, we carry out a comprehensive computational and statistical testing of the presented benchmark in Section 4. We compare the statistical capability of the new benchmark against the benchmark of Taillard with successful results. Another contribution of this research is the new website of instances with many potentially useful features for other researchers to use. This web 2.0 portal contains different benchmarks along with historical data of best results, lower bounds and all types of information as the data is held in a database and in a content management system. All this is explained in Section 5. Finally, Section 6 concludes the paper and gives further research directions.

2. Flowshop scheduling problem and the hardness of the instances

The problem consists of determining a processing sequence of n jobs in a set of m machines that are disposed in series. All jobs must be processed sequentially in all machines. This processing sequence is, without loss of generality, $\{1,\ldots,m\}$. Each job $j,\ j=\{1,\ldots,n\}$ needs a processing time of p_{ij} units at each machine $i,\ i=\{1,\ldots,m\}$. This processing time is a non-negative, deterministic and known amount. A flowshop is a common production setting in factories where products start processing at machine or stage 1 and continue processing until they are finished in the last machine m. The determination of a production sequence for all machines needs the exploration of $(n!)^m$ sequences, as there are $n!$ possible job permutations at each machine and this permutation can be changed from machine to machine with what is known as job passing. However, a common simplification in the flowshop literature is to consider only $n!$ schedules and once the production sequence of jobs for the first machine is determined, is kept unaltered for all other machines. This simplified problem is known as the permutation flowshop scheduling problem or PFSP in short. The completion time of a job in the factory is denoted as C_j. The most common objective for the PFSP is the minimisation of the maximum C_j. This is referred to as makespan and denoted as $C_{\max}$.

Johnson (1954) represents the earliest known contribution in the literature, where the author studied the two machine flowshop problem with makespan minimisation. From this work, the well known Johnson's algorithm can be used to optimally solve the problem. In general, for m machines, the problem is denoted as $F/prmu/C_{\max}$ using the well known $\alpha/\beta/\gamma$ notation of Graham, Lawler, Lenstra, and Rinnooy Kan (1979). When $m \geqslant 3$ the flowshop problem is known to be $\mathcal{NP}$-hard for $C_{\max}$ minimisation as per the results of Garey, Johnson, and Sethi (1976).

According to the results of the computational comparison of Ruiz and Maroto (2005), the NEH heuristic of Nawaz, Enscore, and Ham (1983) is a clear performer. More recent methods, such as those of Dong and Ping Chen (2008); Kalczynski and Kamburowski (2008) Rad, Ruiz, and Boroojerdian (2009) have shown NEH outperforming algorithms. As regards metaheuristics, the list is also long. In this case, some of the best performing methods are the Hybrid Genetic Algorithm of Ruiz, Maroto, and Alcaraz (2006) and the Iterated Greedy of Ruiz and Stützle (2007). With Taillard's benchmark, the state-of-the-art as regards metaheuristics for the PFSP has reached a high level of maturity. For example, Vallada and Ruiz (2009) managed to obtain, with a parallel iterated greedy method, an average percentage deviation over the best known solutions of Taillard's benchmark of only 0.25%. However, this small deviation might just be another sign of Taillard's benchmark aging. From the 120 instances of Taillard, the optimum solution is known today for 92 instances. For the remaining 28, the average gap between the best known solution and the highest known lower bound is just 0.94%. Therefore, given the current state-of-the-art performance and how close to the best known solutions are for Taillard's instances, there is a big potential problem in the near future: New and better methods might end up being disregarded due to it not being possible to show better performance than existing algorithms in Taillard's benchmark. However, the fact that method A does not give better solutions than method B in a benchmark that has been practically solved does not mean that in another harder and/or bigger benchmark, method A would not give better solutions.

Let us note that other existing benchmarks for the PFPS and $C_{\max}$ criterion are not more difficult than Taillard's. For example, Demirkol et al. (1998) proposed a total of 600 instances for different flow and job shop problems, including objectives with due dates. As regards flowshop scheduling, the paper presented 120 instances for makespan minimisation. The problem with these instances is that they only reach 50 jobs and 20 machines, which is a much smaller size than the benchmark of Taillard.

In order to come up with a new benchmark one has to make sure that the instances are varied, numerous, representative of real-life situations and, above all, hard. The reason behind the needed hardness is that the benchmark needs to have discriminant power, i.e., given two methods A and B, we need to conclude if A is better than B. If both A and B are very good performers, they might be able to solve easy instances to almost optimality in most cases and thus, the benchmark will be of no use. In summary, the desired characteristics of a good benchmark are the following:

- Exhaustive: large number of instances, small and large sized instances, different combinations of instance size.
- Amenable for statistical analysis: equidistant, that is, the number of jobs and machines go up by a uniform quantity each time.
- Discriminant: statistically significant differences can be easily found when several algorithms are compared.

Benchmark instances have been constructed almost exclusively from uniform random distributions. It has been customary to draw the processing times from a $U[1,99]$ distribution. This is the case of Taillard's benchmark. It is also known that uniformly distributed processing times result in instances that are harder to solve by algorithms. This has created a number of debates. In real-life it is

[1] The list of best known solutions is found at http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt.

expected to have correlations in the processing times, i.e., ''big'' jobs have large processing times in all machines and/or slow/fast machines have larger/shorter processing times for all jobs. This results, according to Watson, Barbulescu, Whitley, Darrell, and Howe (2002), in instances that are very easy to solve. Given this, one could think that then the concept of flowshop scheduling with random instances is a moot issue. However, let us recall that a flowshop is a simplification of reality and as indicated by Dudek, Panwalkar, and Smith (1992), real problems have many more additional constraints. In any case, using hard random instances in benchmarks is an accepted norm in the field of operations research. The same is being routinely done in the travelling salesman problem, location, assignment and in the majority of studied problems. Furthermore, when an algorithm excels in a difficult random problem, it is expected to also excel in more difficult problems and or instances. As a matter of fact, the good performance of the iterated greedy method presented in Ruiz and Stützle (2007) is translated in Ruiz and Stützle (2008) to flowshops with sequence dependent times and tardiness criterion. In Urlings, Ruiz, and Stützle (2010) is also applied to complex hybrid flowlines with many side constraints of application in real industries. In all these works the iterated greedy algorithm that showed good performance in random flowshop instances also resulted in state-of-the-art results for much more difficult and complex problems.

When generating good instances, Taillard (1993) carried out an unspecified number of runs in which he selected hard instances by minimising the gap between a trivial lower bound and the upper bound obtained with taboo search methods. While this might seem unsophisticated, we will later see that elaborating on this procedure is the only known approach.

Characterising hard instances is extremely difficult. While in other areas of computation and optimisation this has been studied in detail, (see for example Lutz, Mhetre, & Srinivasan, 2000 or Borenstein, 2008 for a general discussion), this has been seldom studied for flowshop problems in particular. The work in fitness landscape analysis is a big effort in this direction (see Reeves, 2005 or Reeves, 2007). However, and as pointed out in Reeves (2007), the fitness landscape is not an invariant of the problem instance, i.e., it depends on the algorithm that is being used in the optimisation. For example, Watson, Whitley, Darrell, and Howe (2005) study the search space topology linked to taboo search algorithms for the job shop problem. Marmion, Dhaenens, Jourdan, Liefooghe, and Verel (2011b) study the concept of neutrality of the flowshop fitness landscape. This is the property that the flowshop has when different permutations result in the same makespan value. The paper proposes strategies to take advantage of this property in solution methods. This is further detailed in Marmion, Dhaenens, Jourdan, Liefooghe, and Verel (2011a). However, neither paper proposes alternatives for generating harder instances.

As a result of all of the above, nowadays it is not clear what makes a flowshop instance difficult beyond the fact that uniformly distributed processing times yield harder instances. Therefore, our approach is that of extensive computation. Similar to what Taillard (1993) did, we generate instances randomly and select those that show a high gap between lower and upper bounds. The difference is that we use state-of-the-art lower bounds and extremely effective PFSP algorithms for generating high quality upper bounds. This process, iterated thousands of times, generates extremely hard instances as we will show in later sections.

3. New benchmark

In this section we detail the generation of the new benchmark of instances for the PFSP with $C_{\max}$ minimisation. The objective is to obtain a new hard benchmark with the characteristics explained previously. Some of the new features of the benchmark are: two separated sets of instances (large and small), instances with up to 800 jobs and 60 machines, ease of carrying out statistical analysis and a large number of instances, etc. In the following subsections details about the structure, characteristics and generation of the new benchmark are given.

3.1. Structure and characteristics

The new benchmark consists of 240 large instances and 240 small instances. Small instances are necessary when exact algorithms are proposed for the defined problem since large instances are not suitable for exact methods. In this case, it is important to evaluate both, the exact and heuristic/metaheuristic methods using a set of small instances in order to compare the results and to check the good behaviour of the heuristic/metaheuristic algorithms, when exact solutions are obtained by the exact methods.

Small instances are a set of 240 with the following combinations of number of jobs (n) and number of machines (m): $n = \{10, 20, 30, 40, 50, 60\}$, $m = \{5, 10, 15, 20\}$. For each combination 10 instances are generated, so in total we have $6 \times 4 \times 10 = 240$ small instances. Note that small instances are up to 60 jobs and 20 machines, so we can consider this set to be actually small-medium sized. If we compare with Taillard's, the smallest size is 20 jobs and 5 machines, after 20 jobs, the next size is 50 jobs, so there is an important gap. Regarding the number of machines, there are also gaps, from 10 to 20 machines in some of the instances. Moreover, Taillard's instances are not equidistant, which means, the difference between the number of jobs/machines between two consecutive instances is not the same. For example, from 20 jobs and 5 machines to 20 jobs and 10 machines and 20 jobs and 20 machines. The difference between the two first instances is 5 machines and from the second to the third is 10 machines. All these differences make the statistical analysis of the results when several algorithms are compared difficult. In the same way, our benchmarks allow the orthogonal analysis in design of experiments, i.e., all combinations of n and m are present. That way, two-factor interactions between the number of jobs and machines can also be studied. This is not possible with Taillard's, as some $n \times m$ combinations are missing, like 200×5, 500×5 and 500×10.

Regarding the large instances, they are also a set of 240 where $n = \{100, 200, 300, 400, 500, 600, 700, 800\}$ and $m = \{20, 40, 60\}$. For each combination 10 instances are generated, in total $8 \times 3 \times 10 = 240$ large instances.

In this way, two of the three desired characteristics are satisfied: they are exhaustive and amenable for statistical analysis.

Each instance is saved in a text file following the same structure as Taillards' instances for compatibility: the first row of the file is the number of jobs and number of machines. Then, a matrix with the processing times of each job in each machine is given. The processing times are generated following a uniform distribution between 1 and 99 as usual in the literature and in most existing benchmarks.

3.2. Generation

The generation of the new benchmark is one of the most important parts of this work. A random selection seems not to be a suitable procedure, that is, to generate randomly 240 instances with the processing times to construct the benchmark. In the original paper of Taillard (1993) it is not fully clear how the instances were selected and generated. The author used a taboo searh algorithm to solve several instances and after some experiments he chose the problems that seemed to be the hardest ones, which means, those problems of which the computed makespans are further from a trivial lower bound. Details about how many instances were generated before the selection or how the computational evaluation was carried out are not given.

In this work, an exhaustive and detailed experimental procedure is carried out in order to generate a new benchmark of hard instances. The process is the same for both, small and large instances, and consists of generating thousands of instances and to select the hardest ones.

Specifically, for small instances, 2000 instances are generated for each combination. From these 2000 instances per combination, the hardest 10 are chosen to be part of the new benchmark. For large instances, the procedure is the same, but a 1000 instances are generated for each combination instead of 2000. Therefore, a total of 48,000 small instances and 24,000 large instances are generated. From these ones, 240 small and 240 large will be chosen to be part of the new benchmark, those that result as the hardest to solve.

The difference between two instances of the same size ($n \times m$) is the matrix of processing times, so an instance is more difficult to solve than other instances depending on the processing times of the jobs in the machines. To test how difficult it is to solve an instance we have on the one hand, two effective algorithms for the permutation flowshop scheduling problem with the objective to minimise the makespan (Ruiz et al., 2006; Ruiz & Stützle, 2007). On the other hand, four lower bounds, one from Taillard (1993) and the three best polynomial bounds of Ladhari and Haouari (2005), proposed for the same problem, are computed for each instance. All the generated instances (48,000 small and 24,000 large) are solved by the two effective algorithms, denoted as HGA (Ruiz et al., 2006) and IG (Ruiz & Stützle, 2007), since they are considered the most effective for this problem. Both algorithms are metaheuristics so each one is run three times on each instance. Regarding the stopping criterion for the methods, a maximum elapsed CPU time is set to $n \cdot (m/2) \cdot 120$ milliseconds (small instances) and $n \cdot (m/2) \cdot 90$ milliseconds (large instances), that is, the computational effort inside each group increases as the number of jobs and/or machines increases. For small instances, the time employed ranges from 3 s (10×5) to 72 s (60×20). Regarding large instances, the amount of time varies from 90 s (100×20) to 2160 s (800×60).

Therefore, for each instance we have six makespan values (three for each run of both algorithms) and four lower bounds. We obtain, for each instance, an upper bound (UB) from the minimum makespan among the six makespan values, and a lower bound (LB) from the maximum value among the four computed lower bounds. The objective is to obtain the gap between the upper bound and the lower bound for each instance, following the expression:

$$\text{GAP} = \frac{UB - LB}{LB} \cdot 100. \qquad (1)$$

The higher the GAP value, the harder the instance is, that is, the best known solution is further from the theoretical lower bound. If the upper bound is very close or equal to the lower bound, a gap near zero will be obtained, which means, the instance is easier to solve.

In order to obtain the hardest instances per combination, GAP values for the 2000 instances (small case) or 1000 instances (large case) are sorted from highest to lowest. The 10 first instances per combination are selected to be part of the new benchmark.

The same procedure is applied for both, small and large instances, and as a result of the experimental process, the new benchmark with 240 small instances and 240 large instances is generated.

It is important to note the amount of time needed in total to carry out all the instance generation experiments. For small instances, a total of 87.48 days were needed to obtain the hardest instances. This amount of days is much higher for large instances, 1350 days. That is, in total, 1437 days, which is the same as almost four years, only for the generation experiments in order to select the instances. The time needed for the experimental evaluation of the benchmark (to be discussed in Section 4), was about 670 days. So in total, 2107 days for all the experimental section, which is the same as almost 6 years. Fortunately, we did not spend 6 years on the experiments. All of them were carried out on a cluster of 52 blade servers each one with two Intel XEON E5420 processors running at 2.5 GHz and with 16 GB of RAM memory, reducing drastically the computational time needed. Each processor has four cores and the experiments were carried out in virtualised Windows XP machines, each one with two virtualised processors and 2 GB of RAM memory. The algorithms and lower bounds were coded in Delphi XE.

Table 1
Average relative percentage deviation (*RPD*) for the heuristic methods (Taillard's benchmark).

Instance	NEH	NEHD
20×5	3.35	2.81
20×10	5.02	3.75
20×20	3.73	3.64
50×5	0.84	0.73
50×10	5.12	4.66
50×20	6.35	5.85
100×5	0.46	0.40
100×10	2.13	1.70
100×20	5.23	4.98
200×10	1.43	0.96
200×20	4.53	3.77
500×20	2.24	1.65
Average	3.37	2.91

Table 2
Average relative percentage deviation (*RPD*) for the heuristic methods (VRF_hard_large and VRF_hard_large_short benchmarks).

Instance	NEH	NEHD	Instance	NEH	NEHD
VRF_hard_large benchmark					
100×20	5.63	5.25	500×20	1.98	1.62
100×40	5.44	5.00	500×40	3.24	2.56
100×60	4.80	4.51	500×60	3.47	3.01
200×20	4.24	3.66	600×20	1.78	1.27
200×40	4.54	4.34	600×40	3.17	2.48
200×60	4.61	4.17	600×60	2.99	2.53
300×20	2.91	2.38	700×20	1.40	0.94
300×40	4.06	3.60	700×40	2.85	2.25
300×60	3.92	3.84	700×60	2.89	2.35
400×20	2.42	1.87	800×20	1.32	0.89
400×40	3.55	3.08	800×40	2.61	2.02
400×60	3.70	3.16	800×60	2.74	2.36
Average	NEH	3.35	NEHD	2.88	
VRF_hard_large_short benchmark					
100×20	5.52	5.11	500×20	1.83	1.59
100×40	5.45	5.12	500×40	3.40	2.64
100×60	4.66	4.66	500×60	3.44	3.03
200×20	4.53	3.87	600×20	1.77	1.37
200×40	4.70	4.47	600×40	3.11	2.55
200×60	4.47	4.33	600×60	2.95	2.49
300×20	3.28	2.68	700×20	1.48	0.95
300×40	4.03	3.47	700×40	2.76	2.23
300×60	3.96	3.90	700×60	2.82	2.30
400×20	2.32	1.80	800×20	1.47	1.08
400×40	3.53	3.18	800×40	2.62	2.04
400×60	3.72	3.16	800×60	2.70	2.31
Average	NEH	3.35	NEHD	2.93	

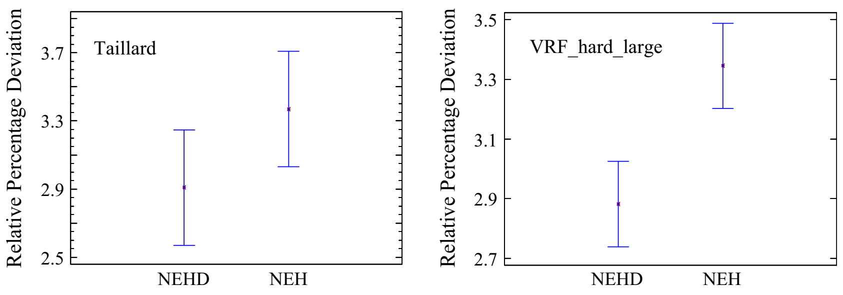


Fig. 1. Means plot and Tukey HSD intervals at 99% confidence level for the heuristic methods (Taillard's instances left side, VRF_hard_large benchmark right side).

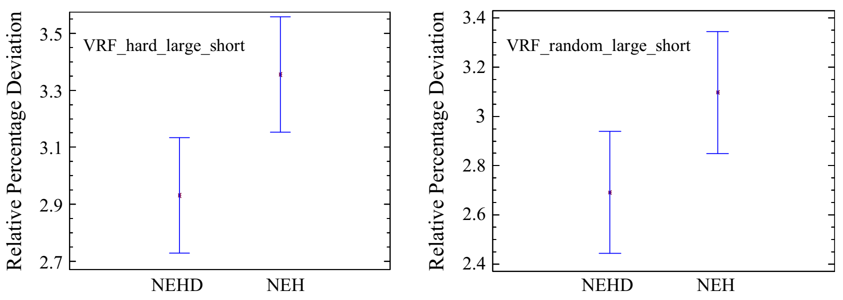


Fig. 2. Means plot and Tukey HSD intervals at the 99% confidence level for the heuristic methods (VRF_hard_large_short benchmark left side, VRF_random_large_short benchmark right side).

4. Computational evaluation of the new benchmark

In this section, the objective is to test, empirically, that the new benchmark is harder to solve than the most used benchmark for this problem, Taillard's benchmark. That is, to satisfy the third desired characteristic: to obtain a discriminant benchmark. In order to check how hard the new proposed benchmark is, several experiments were carried out. In order to make the computational evaluation understandable, firstly a brief summary of all the experiments and instances used is given. Remember that the objective of this work is to propose two new hard benchmarks, small and large. In order to check empirically that the new one proposed benchmarks are really hard, it is necessary to compare the behaviour of some algorithms when solving the new proposed benchmarks and also other benchmarks generated in a different way. That is, to test if the procedure to select the instances for the new benchmarks, explained in Section 3.2, is working, it is necessary to compare with other procedures of instance selection. Furthermore, several benchmarks, apart from the new proposed, are used in the computational evaluation section. In the following, we enumerate the different benchmarks used throughout the paper, as well as the way they are denoted.

- VRF_hard_large benchmark: new proposed benchmark consisting of 240 large instances selected following the procedure explained in Section 3.2.
- VRF_hard_small benchmark: new proposed benchmark consisting of 240 small instances selected following the procedure explained in Section 3.2.
- Taillard benchmark: benchmark proposed by Taillard (1993) with 120 instances.
- VRF_hard_large_short benchmark: in order to be comparable with Taillard's benchmark, a short version of the new proposed benchmark is used, where only 120 instances are selected instead of 240.
- VRF_random_large benchmark: this benchmark consists of 240 instances randomly selected instead of following the process explained in Section 3.2.
- VRF_random_large_short benchmark: similar to the previous one but in order to be comparable with Taillard's benchmark, this random benchmark consists of 120 instances instead of 240.
- VRF_random_small benchmark: as in the previous case, the objective of the use of this benchmark is to test if the exhaustive procedure to select the small instances is working.

In the following subsections details about all the experiments are given.

4.1. Heuristics

First, two heuristics were chosen to solve Taillard's instances and the new proposed hard benchmarks, denoted as VRF_hard_large and VRF_hard_small for large and small instances, respectively. Specifically the NEH developed by Nawaz et al. (1983) and NEHD, which is a modification and improvement of NEH, proposed by Dong and Ping Chen (2008) were selected since they are considered the most effective ones. According to Ruiz and

Table 3
Average relative percentage deviation (*RPD*) for the heuristic methods (VRF_random_large_short benchmark).

Instance	NEH	NEHD	Instance	NEH	NEHD
100 × 20	5.36	4.42	500 × 20	1.19	0.77
100 × 40	5.53	4.97	500 × 40	3.15	2.75
100 × 60	4.90	5.14	500 × 60	3.40	3.06
200 × 20	3.72	2.99	600 × 20	0.75	0.56
200 × 40	4.87	4.63	600 × 40	2.94	2.49
200 × 60	4.50	4.50	600 × 60	3.09	2.82
300 × 20	1.92	1.53	700 × 20	0.73	0.37
300 × 40	4.07	3.53	700 × 40	2.61	2.15
300 × 60	4.01	3.51	700 × 60	2.94	2.35
400 × 20	1.41	1.23	800 × 20	0.56	0.32
400 × 40	3.57	3.02	800 × 40	2.38	2.04
400 × 60	3.85	3.14	800 × 60	2.85	2.29
Average	NEH	3.10	NEHD	2.69	

Table 4
Average relative percentage deviation (*RPD*) for the heuristic methods (VRF_hard_small and VRF_random_small benchmarks).

Instance	NEH	NEHD	Instance	NEH	NEHD
VRF_hard_small benchmark					
10×5	4.09	3.54	40×5	0.93	0.74
10×10	2.18	1.48	40×10	4.93	4.04
10×15	4.55	4.53	40×15	5.92	5.30
10×20	4.43	3.90	40×20	5.50	5.14
20×5	5.51	5.03	50×5	0.56	0.39
20×10	1.87	2.87	50×10	4.66	3.78
20×15	4.58	4.61	50×15	6.59	5.47
20×20	6.04	4.55	50×20	5.94	5.89
30×5	5.55	5.00	60×5	0.70	0.43
30×10	2.09	1.65	60×10	4.59	3.32
30×15	4.43	3.66	60×15	5.79	5.82
30×20	5.90	5.24	60×20	6.19	5.65
Average	NEH	3.87	NEHD	3.49	
VRF_random_small benchmark					
10×5	1.81	1.85	40×5	0.59	0.32
10×10	2.43	2.03	40×10	4.33	3.24
10×15	1.97	1.79	40×15	5.55	5.26
10×20	1.21	1.90	40×20	5.80	5.30
20×5	1.26	1.24	50×5	0.60	0.41
20×10	4.08	3.52	50×10	3.66	3.05
20×15	3.86	3.87	50×15	5.79	5.27
20×20	4.17	3.47	50×20	5.77	5.38
30×5	1.02	1.09	60×5	0.26	0.23
30×10	5.05	3.85	60×10	3.08	2.46
30×15	5.36	4.94	60×15	5.71	4.45
30×20	5.65	4.58	60×20	5.67	5.62
Average	NEH	3.53	NEHD	3.13	

Maroto (2005), the NEH heuristic was the most successful. Dong and Ping Chen (2008) proposed an improvement of the NEH and the authors already needed another benchmark of instances for the comparison since with Taillard's instances they could not show statistically significant differences.

Results of the heuristic methods will be compared against an upper bound for each instance. In order to obtain good upper bounds, the most effective algorithm for this problem, the iterated greedy (IG) proposed by Ruiz and Stützle (2007), is run 20 times with the stopping criterion set to $n\cdot(m/2)\cdot t$ milliseconds ($t=600$) of CPU time. With this amount of time (in the original paper it was $t=60$) we can obtain an extreme solution and furthermore a good upper bound. The minimum makespan value for each instance is selected as an upper bound. The same procedure is applied to all the benchmarks, Taillard, VRF_hard_large and VRF_hard_small, in order to have a comparable scenario.

To measure the effectiveness of the heuristic methods, the Average Relative Percentage Deviation (*RPD*) is computed for each instance according to the following expression:

$$\text{Relative Percentage Deviation}(RPD) = \frac{Method_{sol} - Best_{sol}}{Best_{sol}}\cdot 100, \qquad (2)$$

where $Best_{sol}$ is the best known solution, the upper bound obtained following the previous explanation, and $Method_{sol}$ is the solution obtained with the heuristic method. Remember the algorithms are deterministic heuristics, so only one run is needed.

First, we analyse the large instances, comparing with Taillard's benchmark. After this, we will see the behaviour of the heuristic methods for small instances.

4.1.1. Large instances and Taillard instances

In Tables 1 and 2 (left side) we can see the results for Taillard's and VRF_hard_large benchmarks, respectively, where the 10 instances of each $n\times m$ group have been averaged. We can observe that, on average, results are very similar for both benchmarks. Heuristic NEHD seems to be more effective than NEH. In order to check if these differences in the *RPD* values are statistically significant, we apply an analysis of variance (ANOVA), (Montgomery, 2012), once the three hypotheses for this statistical test are checked: normality, homoscedasticity and independence of the residuals. Fig. 1 show the means plot with HSD Tukey intervals ($\alpha = 0.01$) for both benchmarks. If we focus our attention on the tables, differences, on average, seem to be very similar for both benchmarks. However, the statistical analysis states that these differences are not statistically significant for Taillard's benchmark (intervals are overlapped and p-value is greater than 0.01, specifically 0.07). Results are very different for VRF_hard_large benchmark, we can clearly see that there are statistically significant differences between the average *RPD* values (p-value close to zero). The result with our independent coding and testing of the NEHD matches that of the original

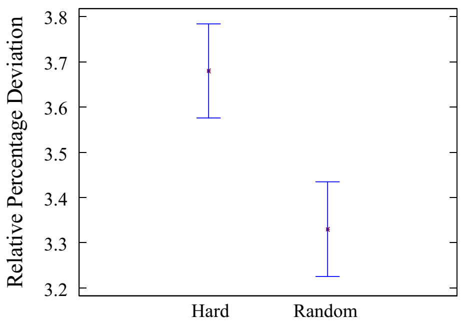


Fig. 3. Means plot and Tukey HSD intervals at the 99% confidence level for the type of instance factor, heuristic methods (small instances).

Table 5
Average relative percentage deviation (*RPD*) for the metaheuristic methods (Taillard's instances and Taillard's instances when only one replicate for each algorithm is run).

Instance	HGA	IG
Taillard benchmark		
20×5	0.04	0.04
20×10	0.10	0.03
20×20	0.08	0.03
50×5	0.01	0.00
50×10	0.47	0.22
50×20	0.93	0.64
100×5	0.01	0.01
100×10	0.22	0.16
100×20	1.01	0.72
200×10	0.13	0.08
200×20	0.85	0.75
500×20	0.46	0.31
Average	0.36	0.25
Taillard benchmark, 1 rep.		
20×5	0.04	0.04
20×10	0.11	0.04
20×20	0.04	0.04
50×5	0.00	0.01
50×10	0.51	0.28
50×20	1.01	0.61
100×5	0.01	0.01
100×10	0.19	0.13
100×20	1.06	0.73
200×10	0.07	0.08
200×20	0.84	0.70
500×20	0.48	0.29
Average	0.36	0.25

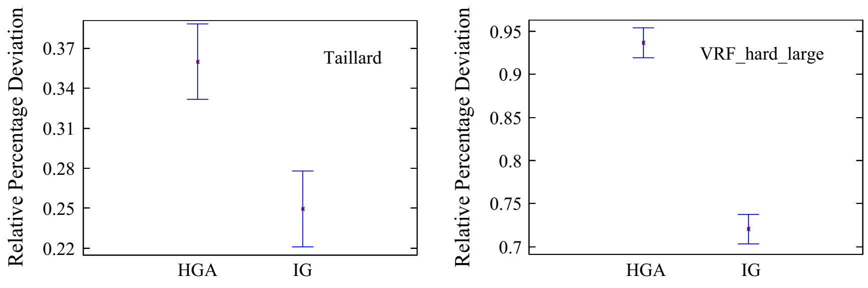


Fig. 4. Means plot and Tukey HSD intervals at the 99% confidence level for the metaheuristic methods (Taillard's instances left side, VRF_hard_large benchmark right side).

authors as they used their own benchmark since there were no differences using Taillard's. So the conclusion is that we have been able to obtain statistically significant differences with our new benchmark which Taillard's benchmark could not.

At this point, one can think that the only reason for obtaining this result is that the VRF_hard_large benchmark is larger (240 instances) than Taillard's (120 instances). Therefore, a short version of VRF_hard_large benchmark was created as a subset of the original one. In the original VRF_hard_large benchmark, 10 instances were chosen per combination. In the short version, 5 instances were selected, so in total, the short version (denoted as VRF_hard_large_short) has 120 instances. Therefore, this new reduced benchmark is comparable in the number of instances with Taillard's benchmark. In Table 2 (right side) we can see the results for the short version. On average, results are very similar to the previous ones. In order to check the differences, an ANOVA analysis is applied, as previously. In Fig. 2 (left side) we can see the means plot with HSD Tukey intervals ($\alpha = 0.01$). From the results, we can conclude that there are statistically significant differences between the average *RPD* values of the heuristic methods. So in conclusion, we can state that the new benchmark is harder to solve despite reducing the size to 120 instances in order to be comparable to Taillard's size.

Another question that arises is what would happen if we selected the instances randomly instead of carrying out the process explained in Section 3.2, i.e., selecting only the instances with the largest GAP. In this case, for large instances, once 1000 instances are generated for each combination, 10 of them are randomly selected for the benchmark. In this way, we can check if the exhaustive process to generate the "hard" instances is working, that is, if the instances from VRF_hard_large_short benchmark are really "harder" to solve.

We carry out a new experiment using a new benchmark with instances randomly selected. In order to be comparable to Taillard's size, the random benchmark (denoted as VRF_random_large_short) consists of 120 instances, that is, 5 instances are randomly selected per each $n \times m$ combination. In Table 3 and Fig. 2 (right side) we can see the results and the means plot with HSD Tukey intervals ($\alpha = 0.01$), respectively. We can observe that there are no statistically significant differences between the average *RPD* values of the heuristic methods. Notice that in Fig. 2 (left side), these differences were significant, so we can conclude that the process for the generation of the new benchmark is working. Obviously in order to compute the Relative Percentage Deviation for the VRF_random_large_short benchmark, the same procedure as VRF_large_hard is applied (as explained at the beginning of Section 4.1), that is, the IG algorithm (Ruiz & Stützle, 2007) is run 20 times for a very long time to obtain good upper bounds.

Table 6
Average relative percentage deviation (*RPD*) for the metaheuristic methods (VRF_hard_large and VRF_random_large benchmarks).

Instance	HGA	IG	Instance	HGA	IG
VRF_hard_large benchmark					
100 × 20	1.22	0.85	500 × 20	0.53	0.37
100 × 40	1.17	0.96	500 × 40	1.05	0.80
100 × 60	0.96	0.80	500 × 60	1.11	0.90
200 × 20	1.03	0.84	600 × 20	0.41	0.28
200 × 40	1.09	0.93	600 × 40	1.12	0.82
200 × 60	1.04	0.87	600 × 60	1.09	0.82
300 × 20	0.69	0.52	700 × 20	0.39	0.30
300 × 40	1.07	0.86	700 × 40	1.03	0.77
300 × 60	1.12	0.84	700 × 60	1.11	0.83
400 × 20	0.55	0.38	800 × 20	0.32	0.22
400 × 40	1.08	0.83	800 × 40	1.05	0.74
400 × 60	1.04	0.88	800 × 60	1.24	0.87
Average	HGA	0.94	IG	0.72	
VRF_random_large benchmark					
100 × 20	1.10	0.73	500 × 20	0.23	0.15
100 × 40	1.13	0.91	500 × 40	1.07	0.80
100 × 60	1.04	0.93	500 × 60	1.22	0.84
200 × 20	0.64	0.50	600 × 20	0.13	0.10
200 × 40	1.17	0.92	600 × 40	1.06	0.75
200 × 60	1.07	0.80	600 × 60	1.13	0.82
300 × 20	0.40	0.30	700 × 20	0.17	0.11
300 × 40	1.09	0.91	700 × 40	1.03	0.68
300 × 60	1.12	0.91	700 × 60	1.20	0.85
400 × 20	0.30	0.22	800 × 20	0.10	0.09
400 × 40	1.08	0.80	800 × 40	1.02	0.63
400 × 60	1.10	0.95	800 × 60	1.17	0.81
Average	HGA	0.87	IG	0.65	

4.1.2. Small instances

It is more difficult to find statistically significant differences between the two tested heuristic methods when small instances are used. Remember, small instances are a set of 240 with the following combinations of number of jobs (n) and number of machines (m): $n = \{10, 20, 30, 40, 50, 60\}$, $m = \{5, 10, 15, 20\}$. It is clear that an instance with 10 jobs and 5 machines will be very easy to solve using any of the heuristic methods. In this case, benchmarks denoted as VRF_hard_small and VRF_random_small are used, where the instances are selected following the procedure explained in 3.2 or randomly, respectively.

NEH and NEHD are run to solve both benchmarks and after the analysis, the conclusions are that there are no statistically significant differences between the heuristics (NEH, NEHD) for either of the benchmarks. This result is quite expected, NEH and NEHD are very similar methods, NEHD is a modification of NEH. For this reason it is very difficult to find statistically significant differences between the methods, especially when small instances are used.

Therefore, a more powerful analysis is carried out in order to compare the Relative Percentage Deviation obtained by the methods using both benchmarks. The higher the *RPD* is, the harder the instance it is trying to solve, that is, the solution obtained by the heuristic is further from the best known solution. A factorial analysis of variance (ANOVA) is carried out where we consider the fol-

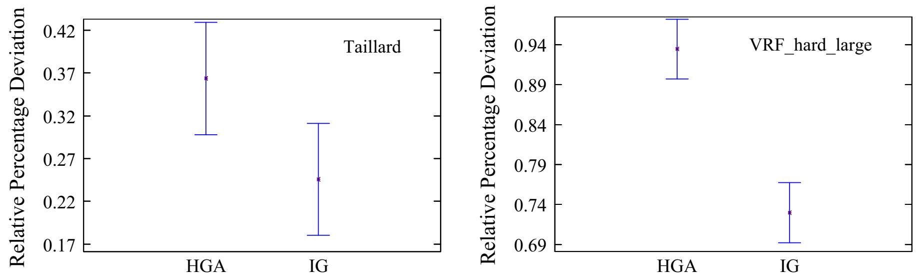


Fig. 5. Means plot and Tukey HSD intervals at the 99% confidence level for the metaheuristic methods when only one replicate for each algorithm is run (Taillard's instances left side, VRF_hard_large benchmark right side).

Table 7
Average relative percentage deviation (*RPD*) for the metaheuristic methods when only one replicate for each algorithm is run (VRF_hard_large and VRF_hard_large_short benchmarks).

Instance	HGA	IG	Instance	HGA	IG
VRF_hard_large benchmark, 1 rep.					
100 × 20	1.09	0.94	500 × 20	0.51	0.36
100 × 40	1.14	0.85	500 × 40	1.05	0.77
100 × 60	0.98	0.89	500 × 60	1.12	0.95
200 × 20	1.07	0.78	600 × 20	0.40	0.28
200 × 40	1.03	1.06	600 × 40	1.11	0.85
200 × 60	1.07	0.91	600 × 60	1.09	0.77
300 × 20	0.67	0.49	700 × 20	0.40	0.30
300 × 40	1.05	0.86	700 × 40	1.02	0.77
300 × 60	1.13	0.82	700 × 60	1.08	0.87
400 × 20	0.54	0.39	800 × 20	0.33	0.25
400 × 40	1.10	0.87	800 × 40	1.05	0.78
400 × 60	1.13	0.81	800 × 60	1.24	0.88
Average	HGA	0.93	IG	0.73	
VRF_hard_large_short benchmark, 1 rep.					
100 × 20	1.01	1.00	500 × 20	0.56	0.36
100 × 40	1.14	0.84	500 × 40	1.13	0.81
100 × 60	0.92	0.84	500 × 60	1.13	0.94
200 × 20	1.16	0.83	600 × 20	0.46	0.30
200 × 40	1.16	1.05	600 × 40	1.11	0.91
200 × 60	1.04	0.97	600 × 60	1.02	0.75
300 × 20	0.79	0.55	700 × 20	0.43	0.38
300 × 40	0.96	0.90	700 × 40	0.98	0.80
300 × 60	1.01	0.81	700 × 60	1.08	0.86
400 × 20	0.49	0.35	800 × 20	0.37	0.25
400 × 40	1.06	0.90	800 × 40	1.04	0.83
400 × 60	1.13	0.80	800 × 60	1.24	0.86
Average	HGA	0.93	IG	0.75	

lowing factors: Algorithm (NEH, NEHD), number of jobs (n), number of machines (m), type of instance. (Hard if the instance belongs to VRF_hard_small benchmark, Random if the instance belongs to VRF_random_small benchmark.)

In this way, we can see the effect of the type of instance in the Relative Percentage Deviation, that is, if hard instances (VRF_hard_small benchmark) result in higher *RPD* values than random instances (VRF_random_small benchmark), we can conclude that hard instances from VRF_hard_small benchmark are more difficult to solve. In Table 4 results for both benchmarks are shown. In Fig. 3 we can see the means plot and Tukey HSD intervals at the 99% confidence level for the Relative Percentage Deviation (*RPD*) according to the type of instance: hard or random. We can observe that the *RPD* computed when hard instances are solved is much higher than for the random case, and the differences between the *RPD* according to the two types of instances are statistically significant (intervals are not overlapped). Then, the conclusion is hard instances are more difficult to solve since the solution obtained by the methods are further from the best known solution. This means that VRF_hard_small benchmark is harder to solve than VRF_random_small benchmark and therefore, the exhaustive procedure to generate VRF_hard_small benchmark explained in Section 3.2 is working also for small instances.

4.2. Metaheuristics

The objective of this section is to test the new benchmarks (VRF_hard_large and VRF_hard_small) with metaheuristic methods. Specifically, two of the most effective algorithms available in the literature for this problem: the Genetic Algorithm proposed by Ruiz et al. (2006) and denoted as HGA and the Iterated Greedy method by Ruiz and Stützle (2007) denoted as IG are selected in order to test the proposed benchmarks and to obtain a comparison with Taillard's. The procedure is the same as that for heuristic methods, several experiments are carried out using all the benchmarks and results are compared.

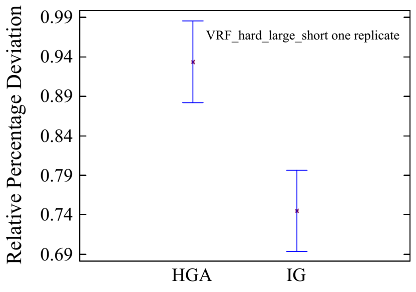
In this case, metaheuristic methods are run until a stopping criterion is met: maximum CPU time is set to $n \cdot (m/2) \cdot 60$ milliseconds. The effectiveness of the methods is measured following the same expression as that for heuristic methods, but in this case five replicates of each algorithm are run as both methods are stochastic. The same computers used for the generation of the instances are used for the rest of the experiments of the paper.

4.2.1. Large instances and Taillard instances

First, both algorithms are run five times to solve Taillard's instances. Results can be seen in Table 5 (left side), where we observe both algorithms are very effective, less than 0.5% from the computed upper bound used in this paper. As before, a statistical analysis (ANOVA) is applied to check whether the differences are statistically significant or not. The means plot at the 99% confidence level is shown in Fig. 4 (left side), where we can see that the differences are statistically significant.

The same procedure is applied for the proposed benchmark (VRF_hard_large). Five replicates of each algorithm are run with the same stopping conditions as those previously. Results can be seen in Table 6 (left side). The statistical analysis is shown in Fig. 4 (right side), where we can observe that the differences are much larger than those obtained for Taillard's and they are also statistically significant. We see that with both benchmarks we obtain statistically significant differences but with our proposed benchmark, the differences are much larger.

At this point, the objective is to check if these statistically significant differences can be obtained earlier (with less computational effort) with the new proposed benchmark. We want to check if instead of five replicates run for each algorithm, results where only one replicate is run give statistically significant differences. In Tables 5 (right side) and 7 (left side) we can see the results where only one replicate is run for each algorithm, for Taillard's and VRF_hard_large benchmarks, respectively. In Fig. 5 (left side) the means plot at the 99% confidence level for Taillard benchmark with



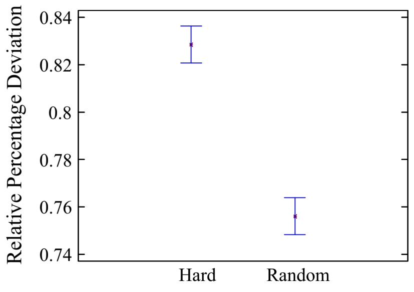


Fig. 6. Means plot and Tukey HSD intervals at the 99% confidence level for the metaheuristic methods when only one replicate for each algorithm is run, left side (VRF_hard_large_short benchmark). Means plot for the type of instance, right side (large instances)

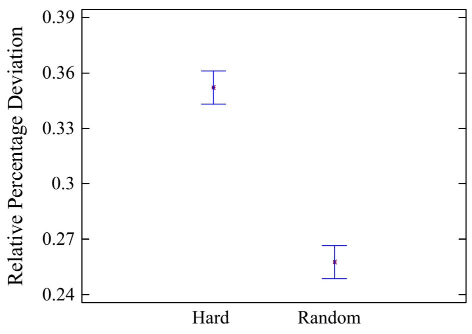


Fig. 7. Means plot and Tukey HSD intervals at the 99% confidence level for the type of instance (small instances).

Table 8
Average relative percentage deviation (*RPD*) for the metaheuristic methods (VRF_hard_small and VRF_random_small benchmarks).

Instance	HGA	IG	Instance	HGA	IG
VRF_hard_small benchmark					
10 × 5	0.00	0.00	40 × 5	0.04	0.03
10 × 10	0.00	0.00	40 × 10	0.67	0.47
10 × 15	0.00	0.00	40 × 15	0.80	0.66
10 × 20	0.00	0.00	40 × 20	0.74	0.64
20 × 5	0.04	0.02	50 × 5	0.00	0.00
20 × 10	0.15	0.05	50 × 10	0.57	0.54
20 × 15	0.06	0.05	50 × 15	1.16	0.86
20 × 20	0.11	0.01	50 × 20	1.05	0.82
30 × 5	0.05	0.03	60 × 5	0.00	0.00
30 × 10	0.58	0.34	60 × 10	0.49	0.34
30 × 15	0.46	0.22	60 × 15	1.04	0.81
30 × 20	0.52	0.31	60 × 20	1.19	0.98
Average	HGA	0.40	IG	0.30	
VRF_random_small benchmark					
10 × 5	0.00	0.00	40 × 5	0.00	0.00
10 × 10	0.00	0.00	40 × 10	0.25	0.11
10 × 15	0.00	0.00	40 × 15	0.87	0.64
10 × 20	0.00	0.00	40 × 20	0.75	0.50
20 × 5	0.02	0.00	50 × 5	0.00	0.00
20 × 10	0.01	0.02	50 × 10	0.22	0.18
20 × 15	0.06	0.03	50 × 15	0.82	0.78
20 × 20	0.06	0.01	50 × 20	0.95	0.65
30 × 5	0.03	0.01	60 × 5	0.00	0.00
30 × 10	0.29	0.22	60 × 10	0.11	0.08
30 × 15	0.31	0.20	60 × 15	0.97	0.64
30 × 20	0.41	0.20	60 × 20	1.06	0.90
Average	HGA	0.30	IG	0.22	

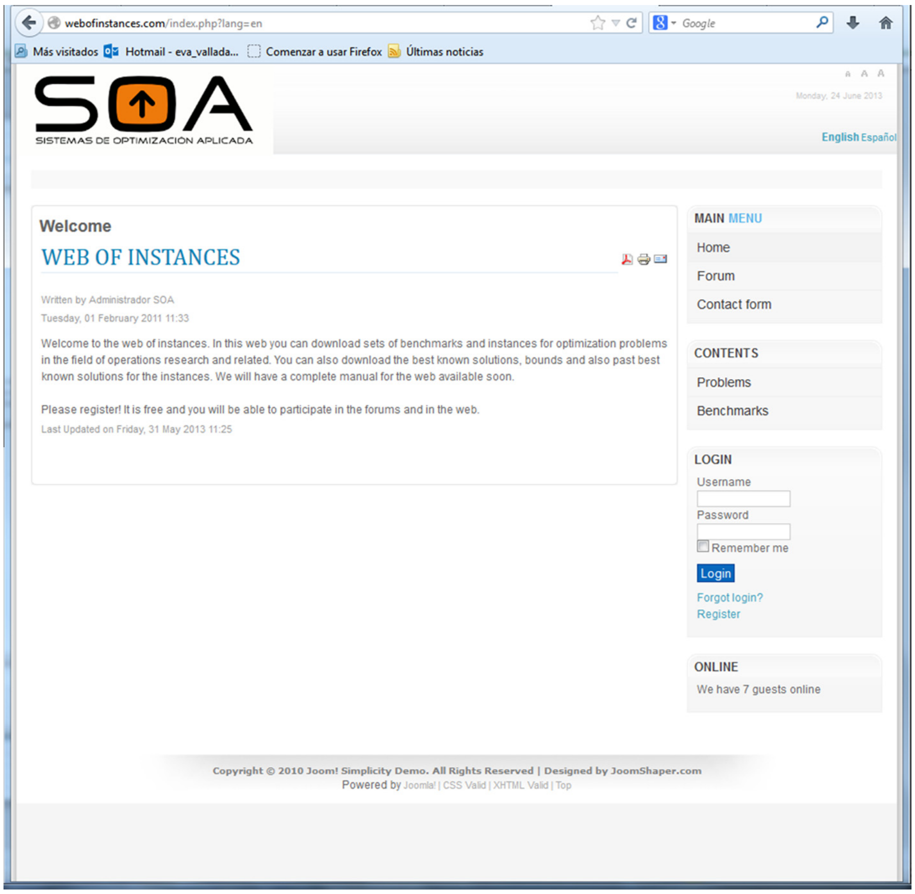
only one replicate of each algorithm, is shown. We can observe that the differences are not statistically significant (intervals are overlapped). The same analysis is carried out for the new VRF_hard_large benchmark, the means plot is shown in Fig. 5 (right side). In this case, we can see that the differences are statistically significant, so the conclusion is that we can obtain the differences with less replicates of the algorithms, that is, with less data and less computational effort. In order to be comparable to Taillard's size, we also analyse the results for the short version of the proposed benchmark (VRF_hard_large_short) also with only one replicate. In Table 7 (right side) results are shown. In Fig. 6 (left side) we can observe the means plot, where the differences are statiscally significant as well. Then, the conclusion after all the experiments is that the new benchmark VRF_hard_large can be considered "harder" and differences are obtained earlier than with Taillard benchmark, even when the size of both benchmarks is the same (VRF_hard_large_short benchmark).

Finally, a last experiment is carried out in order to confirm that the exhaustive procedure to select the "hard" instances is working. In this case, as in Section 4.1, we carry out experiments using both benchmarks, VRF_hard_large and VRF_random_large. At this point, remember we have two new benchmarks: one where the instances are randomly selected among a 1000 instances generated per each combination (VRF_random_large) and the other one where the instances are selected following the process explained in Section 3.2 (VRF_hard_large). The objective of the analysis is to check if there are statistically significant differences in the *RPD* between the two types of instances (Hard and Random). An analysis of variance (ANOVA) is carried out where the following factors are considered: Algorithm (HGA, IG), number of jobs (n), number of machines (m) and type of instance (Hard if the instance belongs to VRF_hard_large benchmark, Random if the instance belongs to VRF_random_large benchmark).

In Table 6 results for HGA and IG algorithms are shown according to the hard and random benchmark, respectively. In Fig. 6 (right side) we can see the means plot and Tukey HSD intervals at the 99% confidence level for the type of instance. We can observe that the type of instance (Hard, Random) affects the *RPD*, and moreover these differences are statistically significant (no overlapped intervals). The *RPD* when hard instances are considered is much higher than for random instances, which means that the two methods (HGA, IG) are further from the best known solution when they solve instances from the "hard" benchmark. In other words, instances from the "hard" benchmark are more difficult to solve than instances from random benchmark, which means that the procedure to select the instances (Section 3.2) is working.

4.2.2. Small instances

The objective of this section is to compare the performance of the metaheuristic methods (HGA and IG) when solving small instances. Remember small instances are a set of 240 with the following combinations of number of jobs (n) and number of machines (m): $n = \{10, 20, 30, 40, 50, 60\}$, $m = \{5, 10, 15, 20\}$. It is clear, and it has been shown in previous experiments that IG and HGA show very differing results. Specifically, the IG method shows



webofinstances.com/index.php?lang=en

Más visitados | Hotmail - eva_vallada... | Comenzar a usar Firefox | Últimas noticias

SISTEMAS DE OPTIMIZACIÓN APLICADA

Monday, 24 June 2013

English Español

Welcome

WEB OF INSTANCES

Written by Administrador SOA

Tuesday, 01 February 2011 11:33

Welcome to the web of instances. In this web you can download sets of benchmarks and instances for optimization problems in the field of operations research and related. You can also download the best known solutions, bounds and also past best known solutions for the instances. We will have a complete manual for the web available soon.

Please register! It is free and you will be able to participate in the forums and in the web.

Last Updated on Friday, 31 May 2013 11:25

MAIN MENU
- Home
- Forum
- Contact form

CONTENTS
- Problems
- Benchmarks

LOGIN

Username

Password

Remember me

Login

Forgot login?

Register

ONLINE

We have 7 guests online

Copyright © 2010 Joom! Simplicity Demo. All Rights Reserved | Designed by JoomShaper.com

Powered by Joomla! | CSS Valid | XHTML Valid | Top

Fig. 8. Home page of the web of instances.

a much better performance than the HGA method, regardless of the benchmark used. Small instances are not an exception, if IG and HGA are run to solve small instances, results are quite expected, IG outperforms HGA. At this point we do not focus our attention on looking for differences between the methods, but rather, looking for differences between the type of instance. As before, we test the two metaheuristic methods (HGA and IG) using the "hard" benchmark (VRF_hard_small) and also the random benchmark (VRF_random_small), as in Section 4.1.2, in order to compare the *RPD*. A full factorial design of experiments and analysis of variance (ANOVA) is carried out where the following factors are considered: Algorithm (HGA, IG), number of jobs (n), number of machines (m), type of instance (Hard if the instance belongs to VRF_hard_small benchmark, Random if the instance belongs to VRF_random_small benchmark). In this way, we can see the effect of the type of instance on the *RPD*, that is, if hard instances result in higher *RPD* values than random instances, we can conclude that hard instances are more difficult to solve. In Fig. 7 we can see the means plot and Tukey HSD intervals at the 99% confidence level for the type of instance. We can observe that when hard instances are solved, the Relative Percentage Deviation is much higher than with random instances. Moreover, in Table 8 results for both benchmarks are shown.

Together with this paper there is available an on-line materials with raw results for all proposed benchmarks (small and large). The author is kindly requested to consult them for further reference.

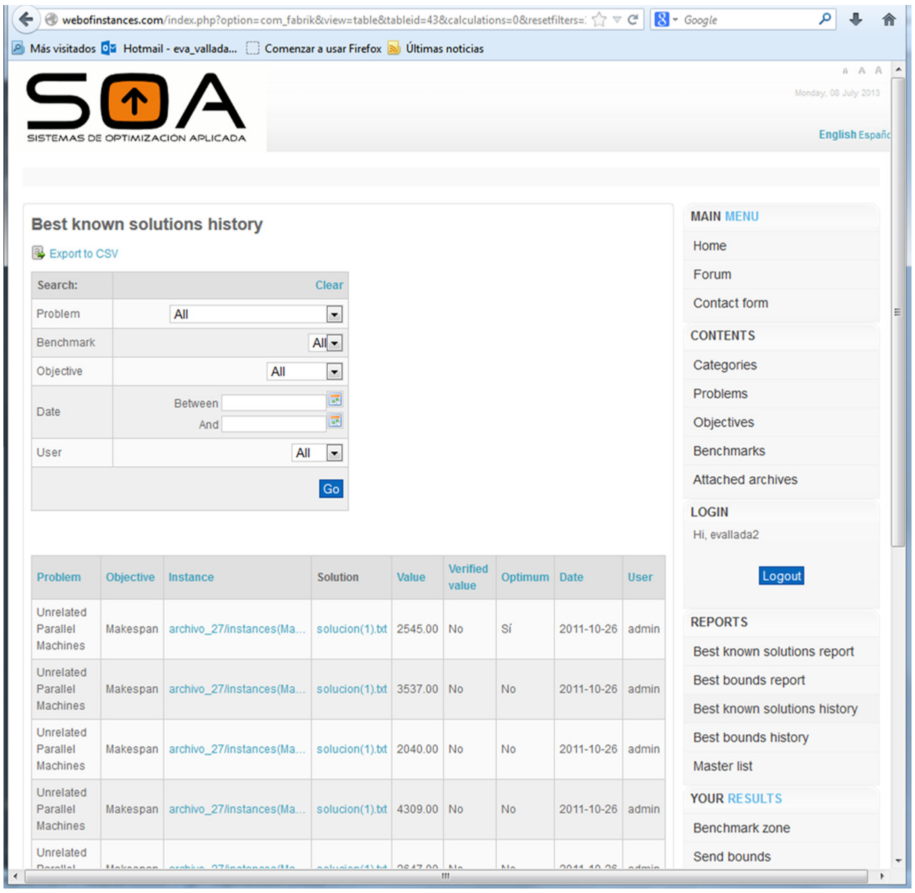
5. Web of instances

There are many examples on the Internet about webs of instances. However, as regards the new development with the so called web 2.0 and content management systems (CMS) little has been done in relation to this. The very famous OR-Library,[2] maintained by professor J.E. Beasley and collaborators is a clear example. It is basically a static file repository, little more than a front-end for an FTP server where all instances are held or links to other websites are found. There is little or no information about the best known objective function values, what the complete solutions are, who obtained the best solutions or when.

It is well known that when an algorithm is proposed in the literature, an exhaustive work related to computational experiments is carried out. For these experiments, it is necessary to have sets of instances available in order to make a comparison with other methods from the literature. It is an essential requirement when a comparison between algorithms is carried out, that the set of instances is the same, in order to make sure of the generalisation and reproducibility of the results. However, the accessibility to the instances and the results from the literature is not always easy, in some cases almost impossible.

One of the objectives of this work is to develop a website for the scientific community (www.webofinstances.com), where sets of instances, including best known solutions and lower bounds, etc., can be uploaded and downloaded. One important question when a scientist wants to make a comparison is not only how to access the set of instances, but also how to access the results obtained from different authors at the date when they carried out their experiments. Let us give an example: we propose and develop an algorithm (denoted as A) to solve a combinatorial optimisation problem. First, we study all the literature available for the same problem. We find a method (denoted as B) published 6 years ago

[2] http://people.brunel.ac.uk/~mastjjb/jeb/info.html.



Best known solutions history

Problem	Objective	Instance	Solution	Value	Verified value	Optimum	Date	User
Unrelated Parallel Machines	Makespan	archivo_27/instances(Ma...	solucion(1).txt	2545.00	No	Sí	2011-10-26	admin
Unrelated Parallel Machines	Makespan	archivo_27/instances(Ma...	solucion(1).txt	3537.00	No	No	2011-10-26	admin
Unrelated Parallel Machines	Makespan	archivo_27/instances(Ma...	solucion(1).txt	2040.00	No	No	2011-10-26	admin
Unrelated Parallel Machines	Makespan	archivo_27/instances(Ma...	solucion(1).txt	4309.00	No	No	2011-10-26	admin
Unrelated								

Fig. 9. Best known solutions history report.

and we want to compare the results of A with B. In this case, we should compare the results of A with those obtained with B 6 years ago. The question is, where are these 6 years old results? Probably nobody knows. Tables with *RPD* values are not helpful since the best solutions with which these *RPD* values were calculated might have changed. Therefore, one of the most important features of the proposed website is the possibility to save results according to a date. In this way, scientists will be able to access the results according to their interests. Moreover, the new website allows us to store full solutions (not just the objective function value). We even allow php code to compute the objective value of any provided full solution so as to check the validity of claimed results. It is also important to remark that best lower bounds are also accessible as well as bibliography referred to lower bounds. Among other features, the website permits the creation of forums for the different benchmarks where researchers can share information about the instances. Finally it is important to remark that the website is not only focused on the permutation flowshop problem with makespan objective, it is available for any combinatorial optimisation problem. In Figs. 8 and 9 we can see pictures for the home page and the best known solution history, respectively.

6. Conclusions

Comprehensive benchmarks are needed in order to cement advancements in algorithmics in general and in scheduling in particular. New solution methodologies must rely on the results obtained with benchmarks in order to assess their performance. Ideally, hard benchmarks with great discriminant power are preferred so that with little experimentation, researchers can ascertain if a given method outperforms the existing state-of-the-art. Sadly, for flowshop problems we still do not know what makes an instance hard. Additionally, existing flowshop benchmarks have already shown problems regarding discriminant power and also present challenges as regards the statistical analysis of results. In this paper, and after more than 6 years worth of CPU time in experiments, we have obtained a new benchmark that we have shown empirically to be hard. The new proposed benchmark complies with the desired characteristics of a good benchmark, that is, it is exhaustive, amenable for statistical analysis and discriminant when several algorithms are compared. The new benchmark consists of two separated sets of instances, small and large, with 240 instances each one. After an exhaustive generation of the instances and a comprehensive computational evaluation we can conclude that the new benchmark is harder to solve than the most commonly used for the same problem, Taillard's benchmark (Taillard, 1993). Moreover, a website for the scientific community has been developed with the objective of sharing benchmarks of instances, best known solutions and historical solutions, etc. for any combinatorial optimisation problem. With this contribution, we hope that the new benchmark will foster new developments in flowshop scheduling in particular and in other optimisation problems in general.

Acknowledgments

Eva Vallada and Rubén Ruiz are partially supported by the Spanish Ministry of Science and Innovation, under the project "RESULT – Realistic Extended Scheduling Using Light Techniques" with reference DPI2012-36243-C02-01 co-financed by the European Union and FEDER funds and by the Universitat Politècnica de València, for the project MRPIV with reference PAID/2012/202. The authors would like to acknowledge the help received from Juan Cabotà in the early development of the web of instances.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejor.2014.07.033.

References

Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*(11), 1069–1072.

Borenstein, Y. (2008). Problem hardness for randomized search heuristics with comparison-based selection: A focus on evolutionary algorithms. PhD thesis, Department of Computer Science, University of Essex.

Carlier, J. (1978). A genetic algorithm for flowshop sequencing. *RAIRO, Operations Research, Recherche Opérationnelle, 12*(4), 333–350.

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

Dong, X., & Ping Chen, H. H. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers and Operations Research, 35*(12), 3962–3968.

Dudek, R. A., Panwalkar, S. S., & Smith, M. L. (1992). The lessons of flowshop scheduling research. *Operations Research, 40*(1), 7–13.

Framinan, J. M., Gupta, J. N. D., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society, 55*(12), 1243–1255.

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

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*(2), 287–326.

Gupta, J. N. D., & Stafford, E. F. Jr., (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research, 169*(3), 699–711.

Hejazi, S. R., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research, 43*(14), 2895–2929.

Heller, J. (1960). Some numerical experiments for an $m \times j$ flow shop and its decision-theoretical aspects. *Operations Research, 8*(2), 178–184.

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

Kalczynski, P. J., & Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers and Operations Research, 35*(9), 3001–3008.

Ladhari, T., & Haouari, M. (2005). A computational study of the permutation flow shop problem based on a tight lower bound. *Computers and Operations Research, 32*(7), 1831–1847.

Lutz, J. H., Mhetre, V., & Srinivasan, S. (2000). Hard instances of hard problems. In H. Reichel & S. Tison (Eds.), *STACS 2000. 17th Annual symposium on theoretical aspects of computer science*. *Lecture notes in computer science* (Vol. 1770, pp. 324–333). Springer.

Marmion, M.-E., Dhaenens, C., Jourdan, L., Liefooghe, A., & Verel, S. (2011a). NILS: A neutrality-based iterated local search and its application to flowshop scheduling. In P. Merz & J.-K. Hao (Eds.), *Evolutionary computation in combinatorial optimization, proceedings of the 11th European conference, EvoCOP 2011. Lecture notes in computer science* (Vol. 6622, pp. 191–202). Springer.

Marmion, M.-E., Dhaenens, C., Jourdan, L., Liefooghe, A., & Verel, S. (2011b). On the neutrality of flowshop scheduling fitness landscapes. In C. A. Coello Coello (Ed.), *Learning and intelligent optimization, selected papers of the 5th international conference, LION 5. Lecture notes in computer science* (Vol. 6683, pp. 238–252). Springer.

Montgomery, D. C. (2012). *Design and analysis of experiments* (eighth ed.). New York: John Wiley & Sons.

Nawaz, M., Enscore, E. E., Jr., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA, The International Journal of Management Science, 11*(1), 91–95.

Pan, Q.-K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers and Operations Research, 40*(1), 117–128.

Potts, C. N., & Strusevich, V. (2009). Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society, 60*(1), S41–S68.

Rad, S. F., Ruiz, R., & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *OMEGA, The International Journal of Management Science, 37*(2), 331–345.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research, 22*(1), 5–13.

Reeves, C. R. (2005). Fitness landscapes. In E. K. Burke & G. Kendall (Eds.), *Search methodologies*. *Introductory tutorials in optimization and decision support techniques* (Vol. 49, pp. 587–610). Springer.

Reeves, C. R. (2007). Landscapes, embedded paths and evolutionary scheduling. In K. P. Dahal, K. C. Tan, & P. I. Cowling (Eds.), *Evolutionary scheduling. Studies in computational intelligence* (Vol. 49, pp. 31–48). Springer.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *OMEGA, The International Journal of Management Science, 34*(5), 461–476.

Ruiz, R., & 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., & 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.

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

Urlings, T., Ruiz, R., & Stützle, T. (2010). Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational Research, 207*(2), 1086–1095.

Vallada, E., & Ruiz, R. (2009). Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research, 193*(2), 365–376.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research, 35*(4), 1350–1373.

Watson, J.-P., Barbulescu, L., Whitley Darrell, L., & Howe, A. E. (2002). Contrasting structured and random permutation flow-shop scheduling problems: Search-space topology and algorithm performance. *INFORMS Journal on Computing, 14*(2), 98–123.

Watson, J.-P., Whitley Darrell, L., & Howe, A. E. (2005). Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search. *Journal of Artificial Intelligence Research, 24*, 221–261.