Automatic Configuration of State-of-the-art Multi-objective Optimizers Using the TP+PLS Framework

A Case Study on Multi-objective Flow-shop Scheduling

Jérémie Dubois–Lacoste
IRIDIA, Université Libre de Bruxelles
50 Av. F. Roosevelt
1050 Brussels, Belgium
jeremie.dubois-lacoste@ulb.ac.be

Manuel López-Ibáñez
IRIDIA, Université Libre de Bruxelles
50 Av. F. Roosevelt
1050 Brussels, Belgium
manuel.lopez-ibanez@ulb.ac.be

Thomas Stützle
IRIDIA, Université Libre de Bruxelles
50 Av. F. Roosevelt
1050 Brussels, Belgium
stuetzle@ulb.ac.be

ABSTRACT

The automatic configuration of algorithms is a dynamic field of research. Its potential for producing highly performing algorithms may change the way we design algorithms. So far, automatic algorithm configuration tools have almost exclusively been applied to configure single-objective algorithms. In this paper, we investigate the usage of automatic algorithm configuration tools to improve multi-objective algorithms. In fact, this is the first article we are aware of where state-of-the-art multi-objective optimizers are configured in an automatic way. This automatic configuration is done for five variants of multi-objective flow-shop problems. Our experimental results show that we can reach at least the same and often a better final quality than a recently proposed state-of-the-art algorithm for these problems.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Design

Keywords

multi-objective, automatic configuration and tuning, flow-shop, scheduling

1. INTRODUCTION

The automatic configuration of algorithms has received a strong attention in recent research efforts. In particular, in recent years a number of new algorithmic tools for the automatic offline configuration of parameterized algorithms have been developed. These include methods such as ParamILS [12], Iterated F-Race [4, 14], gender-based genetic algorithms [3], or CALIBRA [1]. These have extended earlier research efforts [5, 6]. These methods have been instrumental in designing new state-of-the-art algorithms for well known problems and finding improved parameter settings for high performance exact algorithms. The recent work on SATenstein [13], which configures a new state-of-the-art local search algorithm for the SAT problem, or the automatic tuning of the CPLEX software package [11] are noteworthy examples.

So far, however, automatic algorithm configuration tools have almost exclusively been applied to configure single-objective algorithms. In this paper, we investigate the usage of automatic algorithm configuration tools to improve multi-objective algorithms. Only very recently automatic algorithm configuration techniques have been applied to the configuration of multi-objective optimizers. López-Ibáñez and Stützle [16] have applied Iterated F-race to configure a multi-objective ant colony optimization (ACO) framework, leading to new multi-objective ACO algorithms that outperform previously proposed multi-objective ACO algorithms for the bi-objective traveling salesman problem. Wessing et al. [23] have applied tuning methods to tune the variation operator of a multi-objective evolutionary algorithm applied to a single problem instance.

This article is the first to configure a state-of-the-art multi-objective optimizer for $NP$-hard problems. In particular, we configure state-of-the-art algorithms for five variants of multi-objective flow-shop problems. In fact, we are improving over a very recent state-of-the-art algorithm proposed by Dubois-Lacoste et al. [9]. This is a very noteworthy result since the algorithms that were proposed in that paper (i) outperformed by a substantial margin all previously available algorithms for the same multi-objective flow-shop problems; and (ii) they also substantially improved over the best reference solutions available for these problems, which were obtained as the best results from 23 algorithms for the same problems [18].

Our multi-objective optimizer has the very same implementation as the one used in [9], based on a combination of the two-phase local search (TPLS) and the Pareto local search (PLS) frameworks [21]. TPLS tackles multi-objective problems by a sequence of scalarizations that are solved by efficient algorithms for the resulting single-objective problems. PLS is a local search method that tackles multi-
introduce some important notions for the remainder of the framework is general and can in principle be configured for level view of the TPLS and PLS algorithm classes and the SATenstein is a problem-specific framework (for SAT), our paper and the problems we tackle. Section 3 gives a high-
any underlying multi-objective problem. While rithm configuration tool. This approach distinguishes ours also from other approaches such as SATenstein [13] (besides the fact that we tackle multi-objective problems): While SATenstein is a problem-specific framework (for SAT), our framework is general and can in principle be configured for any underlying multi-objective problem.

The paper is structured as follows. In Section 2.1, we introduce some important notions for the remainder of the paper and the problems we tackle. Section 3 gives a high-level view of the TPLS and PLS algorithm classes and the TP+PLS design. In Section 4, we present the automatic algorithm configuration process we applied and discuss the parameters of the current version of the TP+PLS software framework. The experimental results are discussed in Sec-
tion 5 and we conclude in Section 6 with an outline of future research directions.

2. PRELIMINARIES

2.1 Case Study: Permutation Flow-shop Scheduling Problem

The flow-shop scheduling problem (FSP) is one of the most widely studied scheduling problems. In the FSP, a set of $n$ jobs $(J_1, \ldots, J_n)$ is to be processed on $m$ machines $(M_1, \ldots, M_m)$. All jobs go through the machines in the same order, i.e., all jobs have to be processed first on machine $M_1$, then on machine $M_2$, and so on until machine $M_m$. A common restriction in the FSP is to forbid job passing, i.e., the processing sequence of the jobs is the same on all machines. In this case, a candidate solution can be represented as a permutation of the jobs and, hence, there are $n!$ possible sequences. The resulting problem is called permutation flow-shop scheduling problem (PFSP).

In this paper, we focus on the minimisation of following objectives: the makespan $(C_{\text{max}}$, that is, the completion time of the last job), sum of flowtimes $(SFT$, that is the sum of the completion times of all jobs), weighted tardiness $(WT$, that is, the sum of the amount of time jobs are late weighted by each job's priority) and the total tardiness $(TT$, that is the same as WT but all priorities are equal). We tackle the bi-objective PFSPs that result from five possible pairings of objectives (we do not consider the combination of the total and weighted tardiness): $\text{PFSP}-(C_{\text{max}}, SFT)$, $\text{PFSP}-(C_{\text{max}}, WT)$, $\text{PFSP}-(C_{\text{max}}, TT)$, $\text{PFSP}-(SFT, WT)$ and $\text{PFSP}-(SFT, TT)$. Minella et al. [18] provide more details on these bi-objective problems.

2.2 Bi-Objective Optimization and Quality Indicators

In bi-objective combinatorial optimization problems, candidate solutions are evaluated according to an objective function vector $\bar{f} = (f_1, f_2)$. Given two vectors $\bar{u}, \bar{v} \in \mathbb{R}^2$, we say that $\bar{u}$ dominates $\bar{v}$ ($\bar{u} \prec \bar{v}$) iff $\bar{u} \neq \bar{v}$ and $u_i \leq v_i$, $i = 1, 2$. Without preference information about the objectives, the goal is to minimize the objective functions in terms of Pareto-optimality, that is, to find the set of solutions that are not dominated by any other feasible solution. This set is called the Pareto set, and its image in the objective space is called the Pareto front. Since this goal is in many cases intractable, it instead becomes to find a set of nondominated solutions that approximates well the Pareto front.

The assessment of the relative quality of different Pareto front approximations is a difficult problem, since they are often incomparable in the Pareto sense. For this purpose several unary quality indicators have been proposed that try to summarise the quality of a Pareto front approximation into a single scalar value. In this paper, we use one of the most widely used indicators, the hypervolume [25, 10]. In two dimensions, the hypervolume of a Pareto front approximation is the area dominated by at least one of its solutions, and bounded by a point that is larger in all objectives than all points in the Pareto front.

3. ALGORITHM DESIGN

The algorithmic framework used in this paper consists of the sequential execution of a TPLS and a PLS algorithm. TPLS uses an effective single-objective algorithm to solve a sequence of scalarized problems, that is, weighted sum aggregations of the multiple objective functions. The sequence of weights used to scalarize the multiple objectives is a crucial aspect of TPLS. We use here a recently proposed improved version called Adaptive Anytime Two-Phase Local Search (AA-TPLS) [8]. Contrary to TPLS, PLS does not rely on weights and it is a local search method based on dominance. In this section, we briefly describe these two techniques, and how we combine them into a final hybrid algorithm.

3.1 Adaptive Anytime Two-Phase Local Search

TPLS in its original version [20] consists of two main phases. In the first phase, a high-quality solution is generated for one objective using an effective single-objective algorithm. This high-quality solution is the seed that initializes the second phase. During this second phase, a sequence of scalarizations are tackled. The single-objective algorithm that tackles these scalarizations uses as a seed the best solution found for the previous scalarization. TPLS works best when solutions close in the objective space are also close in the search space and, hence, it is easier to find a Pareto-optimal solution by starting from another one and performing a scalarization with a different weight. In addition, TPLS takes advantage of any known effective algorithm for each single objective to apply them to the multi-objective case.

As said above, TPLS solves a sequence of scalarizations, more precisely weighted sum scalarizations. The advantage of considering weighted sum scalarized problems when tackling a multi-objective one is that an optimal solution for the former is also a Pareto-optimal solution for the latter. Different scalarizing functions may be used, such as Tchebycheff functions, but the property mentioned above does not necessarily hold. In a bi-objective problem, a normalized weight vector is of the form $\lambda = (\lambda, 1 - \lambda)$, $\lambda \in [0, 1] \subset \mathbb{R}$, and the scalar value of a solution $s$ with objective function vector $\bar{f}(s) = (f_1(s), f_2(s))$ is computed as

$$f_3(s) = \lambda \cdot f_1(s) + (1 - \lambda) \cdot f_2(s).$$ (1)
Recent work [8] has shown that TPLS has a few drawbacks. First, its computation time must be known in advance in order to distribute the computational effort equally in all directions; otherwise, if stopped earlier, the approximation to the Pareto front will be very poor in some regions. Second, it cannot adapt the computational effort to different Pareto front shapes. AA-TPLS [8] has been proposed as an improved version of TPLS that has the anytime property, that is, it aims at producing an as high as possible performance at any moment of its execution [24]. Moreover, this improved version adapts to the shape of the Pareto front, focusing the search on those regions that would improve the overall quality of the Pareto front approximation. In this paper we use this new AA-TPLS as a component of a hybrid TP+PLS algorithm.

We use an iterated greedy (IG) as the underlying algorithm of AA-TPLS. IG has been proposed to minimize the makespan, for which it is state-of-the-art [22]. Recent work [7] has adapted IG to minimize other objectives, that is, total tardiness (weighted or not), sum of flowtimes, and scalarized problems arising from all possible pairwise combinations of these three objectives. In that work, single-objective automatic configuration tools were used to find efficient parameter settings of IG for each problem. We use these settings here for IG, since our focus is the automatic configuration of the multi-objective components of our hybrid TP+PLS.

3.2 Pareto Local Search

Pareto Local Search (PLS) can be seen as the extension of iterative improvement algorithms from the single to the multi-objective case [19]. In PLS, a criterion based on Pareto dominance replaces the usual single-objective acceptance criterion.

Given an initial archive of non-dominated solutions, which are initially marked as unvisited, PLS iteratively applies the following steps. First, a solution $s$ is randomly chosen among the ones in the archive that remain unvisited. Then, the neighborhood of $s$ is fully explored and all neighbors that are not weakly dominated by $s$ or by any solution in the archive are added to the archive. Solutions in the archive dominated by the newly added solutions are removed to keep only non-dominated solutions. Once the neighborhood of $s$ has been fully explored, $s$ is marked as visited. When all solutions in the archive have been visited, the algorithm stops. Despite its relative simplicity, PLS is an important component of state-of-the-art algorithms for the bi-objective traveling salesman problem (bTSP) [17], and bi-objective permutation flow-shop scheduling problems (bPFSP) [7, 9]. As the neighborhood operator of PLS, authors in [9] report experiments using three different operators. Two are based on either insertion or exchange moves, while the third one uses a combination of both, thus exploring more solutions but requiring more time. The computation time, for all three operators is unpredictable, and may depend on the instance and even on the order unvisited solutions in the archive are chosen. To ensure that the allocated computation time is used and no more, the version of PLS used in the final hybrid algorithm is time bounded, that is, it stops once the time limit is reached.

3.3 Hybrid TP+PLS Algorithm

Our framework for the hybrid algorithm in this paper is based on the two algorithmic schemes introduced above and is in fact the same as the one proposed in [9]. The hybrid algorithm uses first a single-objective algorithm (in the case of the multi-objective flow-shop problems the IG algorithm) for each single-objective to obtain two high-quality initial solutions for each objective. Then we use AA-TPLS to perform a series of scalarizations that produces a set of high-quality, non-dominated solutions. This set is then further improved by a time-bounded PLS that uses appropriate neighborhood operators; in the specific case of the problems tackled in this paper, these are an insertion operator, an exchange operator, or a combination of both. The result is a hybrid TP+PLS algorithm. Through the particular choices of the underlying single-objective algorithms and the neighborhoods of PLS, we can instantiate the framework of the hybrid algorithm for virtually any bi-objective optimization problem. In the present paper, this instantiation is done for the five bi-objective PFSPs.

The free parameters of the TP+PLS framework are those that define the specific settings used by TPLS and PLS, that is, the multi-objective part of the final algorithm, and the relative duration of these phases. In the next section, we describe the parameters we consider in the current version of the TP+PLS framework and the automatic algorithm configuration method we use.

4. AUTOMATED TP+PLS CONFIGURATION

4.1 Iterated F-Race (I/F-Race)

The automatic configuration tool that we use is I/F-Race [4]; more in detail, we use a new, improved implementation provided by the irace software [14]. This tool handles several parameter types: continuous, integer, categorical, and ordered. Continuous and integer parameters take values within a range specified by the user. Categorical parameters can take any value among a set of possible ones explicitly given by the user. An ordered parameter is a categorical parameter with a pre-defined strict order of its possible values.

As proposed by López-Ibáñez and Stützle [16], I/F-Race may be used to automatically configure multi-objective algorithms by integrating the hypervolume indicator as the evaluation criterion. Moreover, it can be used in a parallel mode to speed-up the automatic configuration process. When evaluating a set of candidate on a given instance we map the largest ever found values (for this particular run on this instance) for the objectives to 2. Then we compute the hypervolume of the set of points we obtained, using as reference point (2.1, 2.1). Since I/F-Race uses a rank-based statistical test, the scale of the hypervolume (based on the quality of the bounds found during the run) does not have any impact.

For the automatic configuration process, we generated 500 training instances of each size, 50 jobs and 20 machines (50x20) and 100 jobs and 20 machines (100x20). These instances were produced following the same procedure described in previous works [18]. The automatic configuration process is stopped after 5000 runs of TP+PLS, and each run is given a time limit proportional to the instance size, 100 seconds for instances of size 50x20 and 200 seconds for instances of size 100x20. Thus, the overall time used by the tuning process for the largest instances is $10^6$ seconds, in practice; much less wall-clock time since the automatic configuration tool was run using a parallel mode.

We compare the configuration of TP+PLS found by I/F-
Race with the configuration reported in the original publication [9]. This configuration was found based on a large set of careful experiments to understand the effect of each algorithm component, and the best design choice for each of them. We call this latter configuration \textit{conf\_hand}. In addition, we also run I/F-Race adding \textit{conf\_hand} to the initial set of candidate configurations. The idea is to “seed” the automatic configuration process with the expertise gained from the manually-tuned configuration. This will hopefully improve the final configuration found by I/F-Race. We call \textit{conf\_tun\_rand} the best configuration obtained from running I/F-Race without knowledge of the \textit{conf\_hand} configuration, and we call \textit{conf\_tun\_ics} the best configuration obtained from running I/F-Race using \textit{conf\_hand} as an initial configuration.

4.2 Parameters

To select the parameters to configure, we considered all parameters having an influence on the multi-objective part of the hybrid algorithm. All values for parameters of the single-objective algorithms are the same one has in [9]. We automatically configure the following parameters of TP+PLS.

\textit{tpls\_tratio}. Since we want to limit the overall time of the TP+PLS algorithm, we need to allocate appropriate computation times to AA-TPLS and to PLS. The ratio of the full computation time allocated to each is determined by a parameter \textit{tpls\_tratio}, which gives the ratio of the overall computation time that is used by AA-TPLS. PLS uses the remaining time. We set it as a real in the range \([0, 1.1]\).

\textit{nb\_scal}. This parameter defines the number of scalarizations performed by AA-TPLS, without including the generation of the two initial solutions for each objective. The higher this value, the larger the number of non-dominated solutions that can potentially be found (only one solution at most is found when tackling a scalarization), but also less time will be available to tackle each scalarization (thus resulting in possibly poorer quality solutions). We call this parameter \textit{nb\_scal}, and we set it as an integer in the range \([0, 30]\). Since this is the number of scalarizations in addition to the two initial solutions, when using \textit{nb\_scal} = 0, AA-TPLS would provide the two initial solutions to PLS.

\textit{iscal\_tratio}. AA-TPLS first generates two initial solutions, one for each objective, by running specialized single-objective algorithms. Intuitively, for subsequent scalarizations to be effective, these two initial solutions must be very close to the Pareto front and, hence, the algorithm should spend more effort in generating them than other solutions. We calculate the time assigned to each scalarization as

\[ T_{scal} = \frac{T_{TPLS}}{nb\_scal + 2 \cdot iscal\_tratio} \]

Therefore, the time assigned to finding each of the two initial solutions becomes \textit{iscal\_tratio} \( T_{scal} \), where \textit{iscal\_tratio} is a parameter of the algorithm. We set \textit{iscal\_tratio} as an ordered parameter with values \([1, 1.5, 2, 3, 4, 6, 8, 10]\).

\textit{two\_seeds}. AA-TPLS can perform one or two scalarizations with the same weight, using two different seeds in the latter case [8]. A recent study [9] has shown that the best choice depends on the behavior of the underlying single-objective algorithm for the problem tackled. The choice of which method is used is controlled by the boolean parameter \textit{two\_seeds}, and we set it as a categorical with two possible values \{yes, no\}.

\textit{restart}. At each scalarization of TPLS, the underlying single-objective algorithm starts from a solution previously found. In a simpler “restart” approach, each scalarization starts from a random solution. The best choice for this is, again, dependent of the single-objective algorithm and of the problem. We call this parameter \textit{restart}, and we set it as a categorical with two possible values \{yes, no\}.

\textit{theta}. In AA-TPLS, the calculation of the weights used at each scalarization may be modified by a parameter \theta to avoid scalarizations with the same seed a very similar weights. The larger is \theta, the larger is the difference between subsequent weights [8]. Based on experiments carried out in previous studies [9], it was shown to possibly lead to some improvements. We call this parameter \textit{theta} and we set it as a real in the range \([0, 0.5]\).

\textit{pls\_operator}. This parameter controls the neighborhood operator used by PLS. For the bPFSPs, we tested two different operators, based on exchange (\textit{ex}) and insertion (\textit{ins}) moves, and we consider the combination of them (\textit{exins}) as a third possibility. We call this parameter \textit{pls\_operator} and we set it as a categorical one with values \{ex, ins, exins\}.

Table 1 gives a summary of the search domain for the multi-objective automatic configuration of our TP+PLS algorithm.

### Table 1: List of parameters and their domains, given either as a range or as an explicit set.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpls_tratio</td>
<td>real</td>
<td>[0, 1, 1]</td>
</tr>
<tr>
<td>iscal_tratio</td>
<td>ordered</td>
<td>{1, 1.5, 2, 3, 4, 6, 8, 10}</td>
</tr>
<tr>
<td>nb_scal</td>
<td>integer</td>
<td>[0, 30]</td>
</tr>
<tr>
<td>two_seeds</td>
<td>categorical</td>
<td>{yes, no}</td>
</tr>
<tr>
<td>restart</td>
<td>categorical</td>
<td>{yes, no}</td>
</tr>
<tr>
<td>theta</td>
<td>real</td>
<td>[0, 0.5]</td>
</tr>
<tr>
<td>pls_operator</td>
<td>categorical</td>
<td>{ex, ins, exins}</td>
</tr>
</tbody>
</table>

5. Experimental Analysis

In this section we compare the configurations proposed in the paper [9], where it is shown that the hybrid algorithm using this parameterization improves greatly upon previous state-of-the-art algorithms. We call this reference configurations \textit{conf\_hand}. It differs for each size of instance but it is the same for all problems; hereafter, wherever we mention \textit{conf\_hand} it refers to the size of instances that is precised by the context. These two configurations are presented in Table 2.

For each problem and each instance size we present in Table 3 the best configurations we obtained. We call the configurations obtained from the run without initial candidate \textit{conf\_tun\_rand}, and the configurations obtained from the runs with \textit{conf\_hand} as an initial candidate \textit{conf\_tun\_ics}. Again, from the context it should be clear when we mention \textit{conf\_tun\_rand} and \textit{conf\_tun\_ics} to which size of instances and which problem we are referring to. Surprisingly, one can see that for PFSP-SFT, TT and instances of sizes 100x20, the best configurations obtained from the two runs of \textit{irace} do not use PLS at all, TPLS being run for all the available computation time. This configuration is clearly different
from the one used in [9]. However, given the results in [9] or [18], this choice is actually understandable. In fact, for the biobjective problems with PFS-(SFT, TT) only very few non-dominated solutions exist and the connectedness of non-dominated solutions is very low; this also implies that PLS is not a good choice for these instances.

5.1 Experimental Setup

For the experimental analysis of the configurations, we use 100 test instances produced following the procedure described in [18]. These instances are different from the ones used in the configuration process, to avoid an overtuning effect that could bias the comparison. This test set consists of 100 instances of size 50x20 and 100 instances of size 100x20. Each experiment is run until a time limit of 0.1·n·m seconds (n being the number of jobs and m the number of machines) to allow a computation time proportional to the instance size, as previously suggested by Minella et al. [18]. Each experiment is repeated 10 times with different random seeds.

To normalize the hypervolume value across all instances, we first normalize all non-dominated points to the range [0,1]. This is done by considering for each instance the smallest values found for an objective across all runs and mapping these values to zero; analogously, we map the largest values ever found for the objectives to 2. Then we compute the hypervolume of the set of points we obtained, using as reference point (2.1, 2.1).

5.2 Statistical Analysis

In this section we analyze the results by means of statistical tools. We first present boxplots of the hypervolume for three different problems and 4 instances of each size in Fig. 2. Note that these 4 instances are the first ones of the 100 test instances, the rest being available as supplementary material [2]. The results are rather similar for the two instance sizes and the different problems. There is a high variability, but conf_hand often obtains the worst results among the three configurations and it is never clearly better than the others.

To assess whether the performance differences among the configurations are significant, we perform a statistical test on the overall results. Table 4 presents the mean and standard deviation of the hypervolume for each problem and each configuration, for instances of size 50x20. We perform a paired Wilcoxon signed-rank test with the null hypothesis of equal performance and a confidence level of 0.99, between the conf_hand configuration and each of the other two. A bold face indicates that the difference is statistically significant in favor of one of the automatically derived configurations, while an italic face indicates that the difference is statistically significant in favor of conf_hand. Table 4 presents the same results for instances of size 100x20.

In all cases except one (PFPS-(SFT, WT) on Table 5), conf_hand obtains the worst results, the difference being often statistically significant. In particular, conf_run-rnd improves in nine out of the ten cases significantly over conf_hand (see Tables 4 and 5). For the only case where conf_hand appears better than conf_run-rnd (PFPS-(SFT, WT) on Table 5), one can observe that using conf_hand as an initial candidate ensure that we reach at least the same final quality, as shown by the result of conf_run-ic. Even if the absolute differences in hypervolume are not very large, this is a remarkable result given the excellent performance that the hybrid TP+PLS using the conf_hand configuration achieved when compared to previous state-of-the-art algorithms [9].

5.3 Graphical Analysis

To explore graphically the performance of each configuration, we examine their empirical attainment functions (EAF). The EAF of an algorithm provides an estimation of the probability for an arbitrary point in the objective space to be attained by (dominated by or equal to) a solution obtained by a single run of the algorithm. Thus, examining the EAF allows to know with which frequency a region of the objective space is attained by a multi-objective algorithm. By examining the differences between the EAFs of two algorithms, one can not only identify regions of the objective space where one algorithm performs better than another but also know by which magnitude. The differences in favor of each algorithm can be plotted side-by-side and the magnitude of the differences be encoded in gray levels. The darker the color of the point, the higher the difference. For more details, we refer to López-Ibáñez et al. [15].

Figure 1 presents the differences of the EAFs for conf_hand and conf_run-rnd for 4 instances of size 50x20, the rest being available as supplementary material [2]. Other instances and objectives, for all automatically derived configurations show the same trend: each algorithm performs better in different regions, but one can hardly assess that one outperforms the other across the whole non-dominated front.

6. CONCLUSION

In this paper, we have automatically configured a state-
Table 2: Candidate configurations proposed in [9], leading to state-of-the-art performance. We call these configurations conf_hand.

<table>
<thead>
<tr>
<th>tpls_ratio</th>
<th>iscal_ratio</th>
<th>nb_scal</th>
<th>two_seeds</th>
<th>restart</th>
<th>theta</th>
<th>pls_operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x20, all problems</td>
<td>0.9</td>
<td>1.5</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>100x20, all problems</td>
<td>0.5</td>
<td>1.5</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 3: Candidate configurations obtained from the automatic configuration process, for each instance size and each problem.

<table>
<thead>
<tr>
<th>tpls_ratio</th>
<th>iscal_ratio</th>
<th>nb_scal</th>
<th>two_seeds</th>
<th>restart</th>
<th>theta</th>
<th>pls_operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C_max, SFT)</td>
<td>conf_tun-rnd</td>
<td>0.94</td>
<td>8</td>
<td>18</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.91</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(C_max, TT)</td>
<td>conf_tun-rnd</td>
<td>0.85</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.81</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(C_max, WT)</td>
<td>conf_tun-rnd</td>
<td>0.81</td>
<td>8</td>
<td>19</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.81</td>
<td>3</td>
<td>16</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(SFT, TT)</td>
<td>conf_tun-rnd</td>
<td>0.91</td>
<td>3</td>
<td>18</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.9</td>
<td>3</td>
<td>25</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(SFT, WT)</td>
<td>conf_tun-rnd</td>
<td>0.90</td>
<td>4</td>
<td>17</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.87</td>
<td>3</td>
<td>21</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100x20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C_max, SFT)</td>
<td>conf_tun-rnd</td>
<td>0.64</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.58</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(C_max, TT)</td>
<td>conf_tun-rnd</td>
<td>0.59</td>
<td>4</td>
<td>14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.49</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(C_max, WT)</td>
<td>conf_tun-rnd</td>
<td>0.39</td>
<td>4</td>
<td>22</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.43</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(SFT, TT)</td>
<td>conf_tun-rnd</td>
<td>0.95</td>
<td>1.5</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.96</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(SFT, WT)</td>
<td>conf_tun-rnd</td>
<td>0.98</td>
<td>1.5</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>conf_tun-ic</td>
<td>0.94</td>
<td>3</td>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

of-the-art algorithm for five bi-objective flow-shop problems.
The general algorithmic outline of the final TP+PLS algorithm follows that of [9] (and uses actually the same implementation). However, due to the usage of an automatic algorithm configuration tool, many more algorithm configurations could be examined than in the more tedious hand-tuning approach, which is probably the finally responsible factor for the improved algorithm performance.

There are a number of directions into which we will extend this work. One is to consider additional parameters for configuration. Examples are the parameters of the underlying single-objective algorithms but also the consideration of algorithmic variants of PLS such as considering different acceptance criteria during the search or the addition ofarchive bounding strategies. Another direction is the application of the TP+PLS framework to other bi-objective problems with the goal of possibly generating new state-of-the-art algorithms. Yet another direction is the exploration of other ways of evaluating the found configurations, in particular, by measures that take more into account the particularities of multi-objective problems. In any case, we hope that the presented results encourage other researchers to consider automatic algorithm configuration techniques as a viable alternative to the design of effective multi-objective optimizers.

7. ACKNOWLEDGMENTS

This work was supported by the META-X project, an Action de Recherche Concertée funded by the Scientific Research Directorate of the French Community of Belgium, and by the MIBISOC network, an Initial Training Network funded by the European Commission, grant PITN–GA–2008–238819. Thomas Stützle acknowledges support from the Belgian F.R.S-FNRS, of which he is a Research Associate. The authors also acknowledge support from the FRFC project “Méthodes de recherche hybrides pour la résolution de problèmes complexes”.

8. REFERENCES

Figure 1: Differences of the empirical attainment functions estimated over 10 runs for $conf_{hand}$ and $conf_{run-rnd}$ for 4 instances of size 50x20. The problem is $PFSP-(C_{\text{max}}, WT)$. All EAF plots (all instances, all problems) are available as supplementary material [2].


Instances of size 50x20

Instances of size 100x20

Figure 2: Boxplots of the hypervolume across 10 runs, for 4 instances of each size (the same instance being on the same column). The problems are $PFSP-(C_{\text{max}}, SFT)$ (first row) and $PFSP-(C_{\text{max}}, TT)$ (second row). On each boxplot $\text{conf}_{\text{hand}}$ is on the left, $\text{conf}_{\text{tun-\text{rnd}}}$ in the middle and $\text{conf}_{\text{tun-ic}}$ on the right. All boxplots (all instances, all problems, and both sizes) are available as supplementary material [2].


