Effective Hybrid Stochastic Local Search Algorithms for Biobjective Permutation Flowshop Scheduling

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
jeremie.dl@gmail.com, {manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. This paper presents the steps followed in the design of hybrid stochastic local search algorithms for biobjective permutation flow shop scheduling problems. In particular, this paper tackles the three pairwise combinations of the objectives (i) makespan, (ii) the sum of the completion times of the jobs, and (iii) the weighted total tardiness of all jobs. The proposed algorithms are combinations of two local search methods: two-phase local search and Pareto local search. The design of the algorithms is based on a careful experimental analysis of crucial algorithmic components of the two search methods. The final results show that the newly developed algorithms reach very high performance: The solutions obtained frequently improve upon the best nondominated solutions previously known, while requiring much shorter computation times.

1 Introduction

In this paper, we tackle biobjective flowshop scheduling problems. The flowshop environment models problems where each job consists of a set of operations that are to be carried on machines, and the machine order is the same for each job. Flowshops are a common production environment, for example in the chemical or the ceramic tile industry. We consider flowshops minimizing the following criteria: the completion time of the last job (makespan), which has been the most intensively studied criterion for this problem [1]; the sum of completion times (total flowtime) of all jobs, which recently has attracted a lot of efforts [2,3]; and the weighted tardiness, a criterion which is important in practical applications [4]. For an overview of the biobjective flowshop problems that result from each combination of these objectives, we refer to Minella et al. [5].

At a high level, our approach is based on the hypothesis that effective stochastic local search (SLS) algorithms for multi-objective combinatorial optimization problems (MCOPs) can be obtained by (i) developing (or simply using known) very effective algorithms for the underlying single-objective problems, and (ii) using these single-objective algorithms as components of higher-level algorithm frameworks for tackling multi-objective problems. Further, multi-objective specific local search routines may be examined as an alternative for reaching high-performance algorithms or as a post-processing step.

M.J. Blesa et al. (Eds.): HM 2009, LNCS 5818, pp. 100–114, 2009.

As a first step in our SLS algorithm development, we adopted the state-of-the-art SLS algorithm for the flowshop problem under makespan objective, the iterated greedy (IG) algorithm of Ruiz and Stützle [6]. Subsequently, this algorithm was extended to the sum of flowtime and weighted tardiness objectives and we fine-tuned the resulting algorithms. In a next step, we extended these algorithms to tackle the biobjective versions of the flowshop problem that result from each of the pairwise combinations of the three above mentioned objectives. This was done by integrating the IG algorithms into the two-phase local search (TPLS) framework [7]. At the same time, we also implemented Pareto local search (PLS) [8] algorithms that use different neighborhood structures. A core part of our work is the careful experimental study of the main algorithmic components of the resulting TPLS and PLS algorithms. The insights from this study were then used to propose a hybrid SLS algorithm that combines the TPLS and the PLS algorithms. The final experimental results with this algorithm show its excellent performance: It often finds better Pareto fronts than those of a reference set that was extracted from the best nondominated solutions obtained by a set of 23 other algorithms.

The paper is structured as follows. In Section 2 we introduce basic notions needed in the following. In Section 3 we describe the single-objective algorithms that underlie the two-phase local search approach. Section 4 presents results of the various experimental studies and we conclude in Section 5.

2 Preliminaries

2.1 Multi-objective Optimization

In MCOPs, (candidate) solutions are ranked according to an *objective function vector* $\boldsymbol{f} = (f_1, \ldots, f_d)$ with d objectives. If no *a priori* assumptions upon the decision maker's preferences can be made, the goal typically becomes to determine a set of feasible solutions that "minimize" $\boldsymbol{f}$ in the sense of Pareto optimality. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are vectors in $\mathbb{R}^d$, we say that $\boldsymbol{u}$ *dominates* $\boldsymbol{v}$ ($\boldsymbol{u} \prec \boldsymbol{v}$) iff $\boldsymbol{u} \neq \boldsymbol{v}$ and $u_i \leq v_i,\ i = 1, \ldots, d$; we say that $\boldsymbol{u}$ *weakly dominates* $\boldsymbol{v}$ ($\boldsymbol{u} \preceq \boldsymbol{v}$) iff $u_i \leq v_i,\ i = 1, \ldots, d$. We also say that $\boldsymbol{u}$ and $\boldsymbol{v}$ are *nondominated* iff $\boldsymbol{u} \not\prec \boldsymbol{v}$ and $\boldsymbol{v} \not\prec \boldsymbol{u}$ and are (pairwise) *non weakly dominated* if $\boldsymbol{u} \not\preceq \boldsymbol{v}$ and $\boldsymbol{v} \not\preceq \boldsymbol{u}$. For simplicity, we also say that a solution s dominates another one s' iff $\boldsymbol{f}(s) \prec \boldsymbol{f}(s')$. If no other s' exists such that $\boldsymbol{f}(s') \prec \boldsymbol{f}(s)$, the solution s is called a *Pareto optimum*. The goal in MCOPs typically is to determine the set of all Pareto optimal solutions. Since this task is in many cases computationally intractable, in practice the goal becomes to find an approximation to the set of Pareto optimal solutions in a given amount of time that is as good as possible. In fact, any set of mutually nondominated solutions provides such an approximation. The notion of Pareto optimality can be extended to compare sets of mutually nondominated solutions [9]. In particular, we can say that one set A dominates another set B ($A \prec B$), iff every $\boldsymbol{b} \in B$ is dominated by at least one $\boldsymbol{a} \in A$.

2.2 Bi-objective Permutation Flowshop Scheduling

In the flowshop scheduling problem (FSP) a set of n jobs $(J_1, \ldots, J_n)$ is given 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, candidate solutions correspond to permutations of the jobs and the resulting problem, on which we focus here, is the permutation flowshop scheduling problem (PFSP). All processing times p_{ij} for a job J_i on a machine M_j are fixed, known in advance and nonnegative. In the following, we denote by C_i the completion time of a job J_i on machine M_m. For a given job permutation π, the makespan is the completion time of the last job in the permutation, i.e., $C_{\max} = C_{\pi(n)}$. For $m \geq 3$ this problem is $\mathcal{NP}$-hard in the strong sense [10]. In the following, we refer to this problem as *PFSP-*$C_{\max}$.

The other objectives we study are the minimization of the sum of flowtimes and the minimization of the weighted tardiness. The sum of flowtimes is defined as $\sum_{i=1}^{n} C_i$. The resulting PFSP with this objective is strongly $\mathcal{NP}$-hard even with only two machines [10]. We refer to this problem as *PFSP-SFT*. For the weighted tardiness objective, each job has a due date d_i by which it is to be finished and a weight w_i indicating its priority. The tardiness is defined as $T_i = \max\{C_i - d_i, 0\}$ and the total weighted tardiness is given by $\sum_{i=1}^{n} w_i \cdot T_i$. This problem we denote *PFSP-WT*; it is strongly $\mathcal{NP}$-hard even for a single machine.

In this paper, we tackle the three biobjective problems that result from the three possible pairs of objectives. A number of algorithms have been proposed to tackle each of these biobjective problems, but rarely more than one possible combination of the objectives has been addressed in a paper. The algorithmic approaches range from constructive algorithms to applications of SLS methods such as evolutionary algorithms, tabu search, or simulated annealing. Minella et al. [5] give a comprehensive overview of the literature on the three problems we tackle here and present the results of an extensive experimental analysis of 23 algorithms, either specific or adapted for tackling the three biobjective PFSPs. They identify MOSA [11] as the best performing algorithm.

2.3 Two-Phase Local Search and Pareto Local Search

In this paper, we study SLS algorithms that represent two main classes of multi-objective SLS algorithms [12]: algorithms that follow a component-wise acceptance criterion (CWAC), and those that follow a scalarized acceptance criterion (SAC). As two paradigmatic examples of each of these classes, we use two-phase local search (TPLS) [7] and Pareto local search (PLS) [8].

Two-Phase Local Search. The first phase of TPLS uses an effective single-objective algorithm to find a good solution for one objective. This solution is the initial solution for the second phase, where a sequence of scalarizations are solved by an SLS algorithm. Each scalarization transforms the multi-objective problem into a single-objective one using a weighted sum aggregation. For a given weight

Algorithm 1. Two-Phase Local Search

```
Input: A random or heuristic solution π
π' := SLS1(π);
for all weight vectors λ do
   π' := SLS2(π', λ);
   Add π' to Archive;
end for
Filter Archive;
```

vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$, the value w of a solution s with objective function vector $\boldsymbol{f}(s) = (y_1, y_2)$ is computed as $w = (\lambda_1 \cdot y_1) + (\lambda_2 \cdot y_2)$, s.t. $\lambda_1, \lambda_2 \in [0, 1] \subset \mathbb{R}$ and $\lambda_1 + \lambda_2 = 1$. In TPLS, each run of the SLS algorithm for solving a scalarization uses as an initial solution the best one found for the previous scalarization. The motivation for using such a method is to exploit the effectiveness of the underlying single-objective algorithm. Algorithm 1 gives the pseudocode of TPLS. We denote by SLS_1 the SLS algorithm to minimize the first single objective. SLS_2 is the SLS algorithm to minimize the weighted sums.

Pareto Local Search. PLS is an iterative improvement method for solving MCOPs that is obtained by replacing the usual acceptance criterion of iterative improvement algorithms for single-objective problems by an acceptance criterion that uses the dominance relation. Given an initial archive of unvisited nondominated solutions, PLS iteratively applies the following steps. First, it randomly chooses an unvisited solution s from the candidate set. 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 eliminated. Once the neighborhood of s is fully explored, s is marked as visited. The algorithm stops when all solutions in the archive have been visited.

We also implemented the *component-wise step* (CW-step) procedure as a post-processing step of the solutions produced by TPLS. It adds nondominated solutions in the neighborhood of the solutions returned by TPLS to the archive, but it does not explore the neighborhood of these newly added solutions further. Hence, CW-step may be interpreted as a specific variant of PLS with an early stopping criterion. Because of this early stopping criterion, the CW-step results in worse nondominated sets than PLS. However, compared to running a full PLS, CW-step typically requires a very small additional computation time.

3 Single-Objective SLS Algorithms

The performance of the single-objective algorithms used by TPLS is crucial. They should be state-of-the-art algorithms for the underlying single-objective problems and as good as possible for the scalarized problems resulting from the weighted sum aggregations. Motivated by these considerations, for *PFSP*-$C_{\max}$ we reimplemented in `C++` the iterated greedy (IG) algorithm (*IG*-$C_{\max}$) by Ruiz

Algorithm 2. Iterated Greedy

```
π := NEH;
while termination criterion not satisfied do
  π_R := Destruction(π);
  π' := Construction(π_R);
  π' := LocalSearch(π')      % optional;
  π := AcceptanceCriterion(π, π');
end while
```

and Stützle [6], which is a current state-of-the-art algorithm for this problem. An algorithmic outline is given in Algorithm 2. The essential idea of IG is to iterate over a construction heuristic by first destructing partially a complete solution; next, from the resulting partial solution π_R a full problem solution is reconstructed and possibly further improved by a local search algorithm. This solution is then accepted in dependence of an acceptance criterion.

More concretely, *IG-*$C_{\max}$ uses the NEH heuristic [13] for constructing the initial solution and for reconstructing full solutions in the main IG loop. (NEH is an insertion heuristic that sorts the jobs according to some criterion and inserts jobs in this order into the partial schedule. Note that this sorting is only relevant when NEH constructs the initial solution; in the main loop of IG the jobs are considered in random order.) In the destruction phase a small number of d randomly chosen jobs are removed. The local search is an effective first-improvement algorithm based on the *insert* neighborhood, where two solutions are neighbors if they can be obtained by removing a job from one position and inserting it in a different one. The acceptance criterion uses the Metropolis condition: A worse solution is accepted with a probability given by $\exp(f(\pi') - f(\pi))/T$, where f is the objective function and the temperature parameter T is maintained constant throughout the run of the algorithm. Parameter values are given in Table 1.

Given the known very good performance of *IG-*$C_{\max}$, we use it also for the other two objectives. However, the speed-ups of Taillard for $C_{\max}$ [14] are not anymore applicable, which leads to a factor n increase of the local search time complexity. As a side result, it is unclear whether the same neighborhood as for the makespan criterion should be chosen. We have therefore considered also (i) the exchange neighborhood, where two solutions are neighbors if they can be obtained by exchanging the position of two jobs; and (ii) the swap neighborhood, where only two adjacent jobs are exchanged. We tested only restricted versions of the insert and exchange neighborhoods, where the possible insertion and exchange moves of only one job are examined.

Other changes concern the formula for the definition of the temperature parameter for the acceptance criterion. This is rather straightforward for *PFSP-SFT*, which can be done by adapting slightly the way the temperature is defined. For *PFSP-WT* no input data-driven setting as for the other two objectives could be obtained due to large variation of the objective values. Therefore, the temperature parameter is defined relating it to a given target percentage deviation from the current solution. Finally, for *PFSP-WT* we explored different ways of defining the initial sequence of jobs for the NEH heuristic.

Table 1. Adaptation of IG for each objective and for the scalarized problems from each combination of objectives. A (↓) denotes a decreasing order, and a (↑) denotes an increasing order. Tp is a parameter of the formula for Temperature. Settings for *IG-*$C_{\max}$ follow [6]. For *IG-SFT*, the formula of Temperature is the same as for *IG-*$C_{\max}$ but multiplied by n. For *IG-WT*, the initial order for NEH is given by the well-known SLACK heuristic [4] augmented with priorities w_i. Parameter d is the number of jobs removed in the destruction phase. Insert-T refers to a full insert iterative improvement using the speed-ups of Taillard [14]; Swap to a full iterative improvement execution using the swap neighborhood and Ins. to the insertion search for one job. For details see the text.

Algorithm	Init. order for NEH	Temperature T	Tp	d	LS
IG-$C_{\max}$	$\sum_{j=1}^{m} p_{ij}$ (↓)	$Tp \cdot \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}}{n \cdot m \cdot 10}$	0.4	4	Insert-T
IG-SFT	$\sum_{j=1}^{m} p_{ij}$ (↓)	$Tp \cdot \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}}{m \cdot 10}$	0.5	5	Swap
IG-WT	$w_i \cdot (d_i - C_i(s))$ (↑)	$\frac{100}{Tp \cdot f(s)}$	0.7	4	Swap+Ins.
IG-$(C_{\max}, SFT)$	$\sum_{j=1}^{m} p_{ij}$ (↓)	$\frac{100}{Tp \cdot f(s)}$	0.5	5	Swap
IG-$(\cdot, WT)$	$w_i \cdot (d_i - C_i(s))$ (↑)	$\frac{100}{Tp \cdot f(s)}$	0.5	5	Swap

We tuned the IG algorithms for *PFSP-SFT* and *PFSP-WT* using iterated F-Race [15] on training instances that are different from those used for the final test results. The final configurations retained from this tuning phase are given in Table 1. The lines *IG-*$(C_{\max}, SFT)$ and *IG-*$(\cdot, WT)$ concern the scalarized problems where the weights are different from one and zero for the indicated objectives. A closer examination of the performance of the resulting single-objective algorithms (not reported here) showed that for total flowtime the final IG algorithm is competitive to current state-of-the-art algorithms as of 2009; for the total tardiness objective the performance is also very good and very close to state-of-the-art; in fact we could improve with the IG algorithms a large fraction (in each case more than 50%) of the best known solutions of available benchmark sets.

4 Multi-Objective SLS Algorithms

In what follows, we first study main algorithm components of the PLS and TPLS algorithms and then present a comparison of a final hybrid SLS algorithm to reference sets of the best solutions found so far for a number of benchmark instances. We used the benchmark from Minella et al. [5], which consists of the benchmark set of Taillard [16] augmented with due dates and priorities. In order to avoid over-tuning, we performed the algorithm component analysis on 20 training instances of size `50x20` and `100x20`, which were generated following the procedure used by Minella et al. [5].

The results are analyzed by graphically examining the attainment surfaces of a single algorithm and differences between the empirical attainment functions

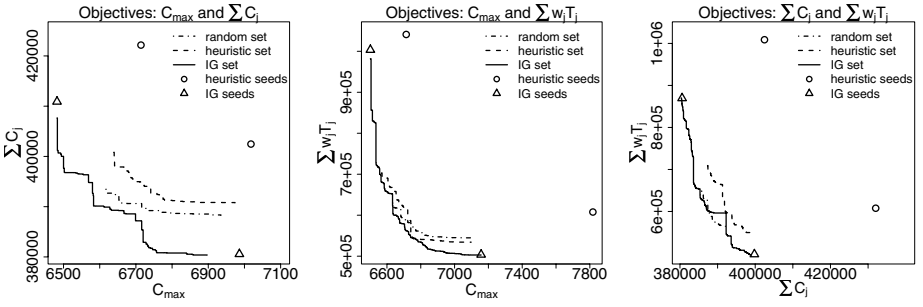

Fig. 1. Nondominated sets obtained by PLS using different quality of seeds for instance `100x20_3`. The randomly generated solution is outside the range shown.

(EAF) of pairs of algorithms. The EAF of an algorithm provides the probability, estimated from several runs, of an arbitrary point in the objective space being attained (weakly dominated) by a solution obtained by a single run of the algorithm [17]. An attainment surface delimits the region of the objective space attained by an algorithm with a certain minimum probability. In particular, the worst attainment surface delimits the region of the objective space always attained by an algorithm, whereas the best attainment surface delimits the region attained with the minimum non-zero probability. Similarly, the median attainment surface delimits the region of the objective space attained by half of the runs of the algorithm. Examining the attainment surfaces allows to assess the likely location of the output of an algorithm. On the other hand, examining the differences between the EAFs of two algorithms allows to identify regions of the objective space where one algorithm performs better than another. Given a pair of algorithms, the differences in favor of each algorithm are plotted side-by-side and the magnitude of the difference is encoded in gray levels. López-Ibáñez et al. [18] provide a detailed explanation of these graphical techniques.

4.1 Analysis of PLS Components

Seeding. As a first experiment, we analyzed the computation time required and the final quality of the nondominated sets obtained by PLS when PLS is seeded with solutions of different quality. We test seeding PLS with: (i) one randomly generated solution, (ii) two good solutions (one for each single objective) obtained by the NEH heuristics (see Table 1), and (iii) two solutions obtained by IG for each objective after 10 000 iterations. Figure 1 gives representative examples of nondominated sets obtained by PLS for each test and indicates the initial seeding solutions of NEH and IG. The best nondominated sets, in terms of a wider range of the Pareto front and higher quality solutions, are obtained when using the IG seeds. Generally, seeding PLS with very good solutions produces better nondominated sets; this result is strongest for the biobjective problem that considers makespan and total flowtime. We further examined the

Table 2. Computation time of PLS for different types of seeds

Objectives	Instance Size	random avg.	random sd.	heuristic avg.	heuristic sd.	IG avg.	IG sd.
$(C_{\max}, \sum C_i)$	50x20	8.85	2.05	6.23	2.48	4.56	0.38
	100x20	177.40	27.60	142.23	29.79	162.14	26.09
$(C_{\max}, \sum w_i T_i)$	50x20	31.61	6.84	33.85	7.46	24.02	3.84
	100x20	641.96	215.55	767.23	299.33	626.48	114.08
$(\sum C_i, \sum w_i T_i)$	50x20	26.72	3.02	28.17	2.62	23.70	3.33
	100x20	742.42	157.10	807.75	121.70	895.23	176.29

Table 3. Computation time of PLS for different neighborhood operators

Objectives	Instance Size	exchange avg.	exchange sd.	insertion avg.	insertion sd.	ex. + ins. avg.	ex. + ins. sd.
$(C_{\max}, \sum C_i)$	50x20	2.21	0.35	1.57	0.44	4.84	1.06
	100x20	77.56	19.44	70.91	12.8	157.64	30.26
$(C_{\max}, \sum w_i T_i)$	50x20	12.94	3.11	10.11	1.75	23.03	4.09
	100x20	314.63	69.08	251.84	49.33	611.6	115.02
$(\sum C_i, \sum w_i T_i)$	50x20	14.24	3.79	9.51	1.8	23.72	3.87
	100x20	492.91	102.59	239.04	101.47	872.32	262.21

computation time required by PLS in dependence of the initial seed in Table 2. Our conclusion is that seeding PLS with very good initial solutions does not strongly reduce computation time. However, given the strong improvement on solution quality, seeding PLS with solutions obtained by TPLS is pertinent.

Neighborhood operator. We experiment with PLS variants using three neighborhoods: (i) insertion, (ii) exchange, and (iii) exchange plus insertion. The latter simply checks for all moves in the exchange and insertion neighborhood of each solution. We measured the computation time of PLS with each underlying operator for each combination of objectives (Table 3). The computation time of the combined exchange and insertion neighborhood is slightly more than the sum of the computation times for the exchange and insertion neighborhoods. For comparing the quality of the results, we examine the EAF differences of 10 independent runs. Figure 2 gives two representative examples. Typically, the exchange and insertion neighborhoods lead to better performance in different regions of the Pareto front (top plot), and both of them are consistently outperformed by the combined exchange and insertion neighborhood (bottom plot).

4.2 Analysis of TPLS Components

Number of scalarizations and number of iterations. In TPLS, each scalarization is computed using a different weight vector. In this paper, we use a regular

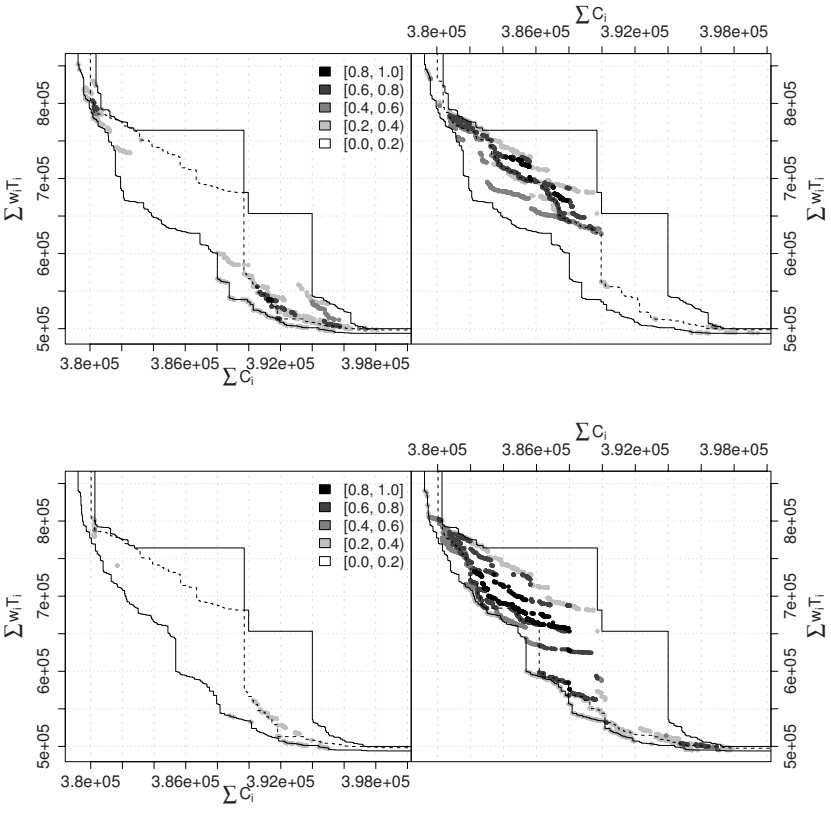

Fig. 2. EAF differences for (*top*) insertion vs. exchange and (*bottom*) exchange vs. exchange and insertion. The combination of objectives is $\sum C_i$ and $\sum w_i T_i$. Dashed lines are the median attainment surfaces of each algorithm. Black lines correspond to the overall best and overall worst attainment surfaces of both algorithms.

sequence of weight vectors from $\boldsymbol{\lambda} = (1, 0)$ to $\boldsymbol{\lambda} = (0, 1)$. If N_{scalar} is the number of scalarizations, the successive scalarizations are defined by weight vectors $\boldsymbol{\lambda}_i = (1 - (i/N_{\text{scalar}}), i/N_{\text{scalar}})$, $i = 0, \ldots, N_{\text{scalar}}$.

For a fixed computation time, in TPLS there is a tradeoff between the number of scalarizations to be used and the number of iterations to be given for each of the invocations of the single-objective SLS algorithm. In fact, the number of scalarizations (N_{scalar}) determines how many scalarized problems will be solved (intuitively, the more the better approximations to the Pareto front may be obtained), while the number of iterations (N_{iter}) of IG determines decisively how good the final IG solution will be. Here, we examine the trade-off between the settings of these two parameters by testing all 9 combinations of the following settings: $N_{\text{scalar}} = \{10, 31, 100\}$ and $N_{\text{iter}} = \{100, 1\,000, 10\,000\}$.

We first studied the impact of increasing either N_{scalar} or N_{iter} for a fixed setting of the other parameter. Although clear improvements are obtained by

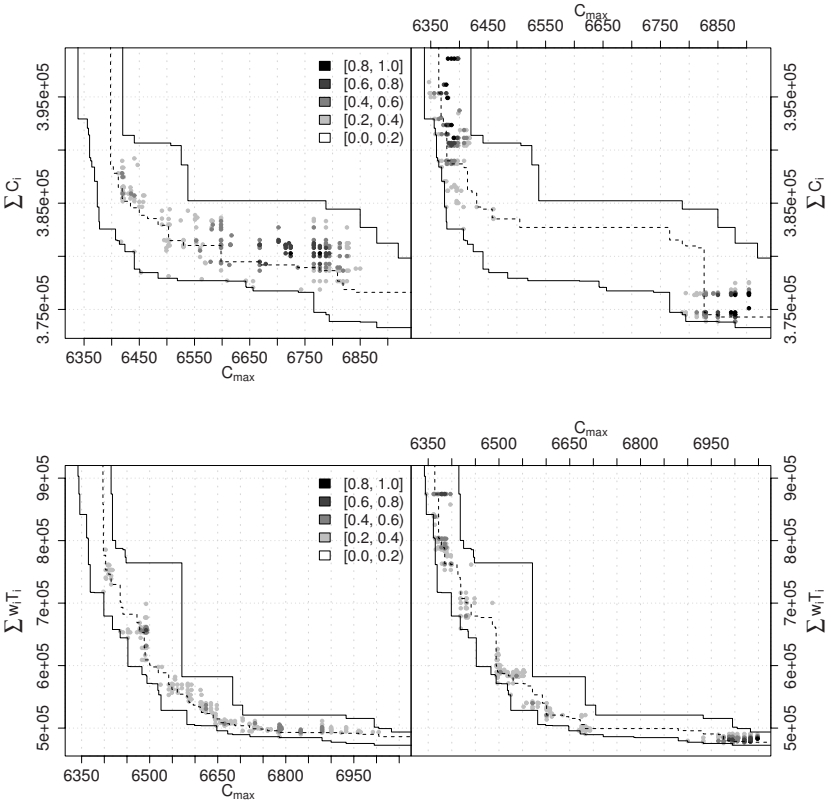

Fig. 3. EAF differences between $N_{\text{scalar}} = 100$ and $N_{\text{iter}} = 1000$, versus $N_{\text{scalar}} = 10$ and $N_{\text{iter}} = 10000$ for two combinations of objectives: $C_{\max}$ and $\sum C_i$ (top) and $C_{\max}$ and $\sum w_i T_i$ (bottom)

increasing each of the two parameters, there are significant differences. While for the number of scalarizations some type of limiting behavior without strong improvements was observed when going from 31 to 100 scalarizations (while improvements from 10 to 31 were considerable), increasing the number of iterations of IG alone seems always to produce significant improvements.

Next, we compare settings that require roughly the same computation time. Figure 3 compares a configuration of TPLS using $N_{\text{scalar}} = 100$ and $N_{\text{iter}} = 1000$ against other configuration using $N_{\text{scalar}} = 10$ and $N_{\text{iter}} = 10000$. Results are shown for two of the three combinations of objective functions. (The results are representative for other instances.) As illustrated by the plots, there is no clear winner in this case. A larger number of iterations typically produces better solutions in the extremes of the Pareto front. On the other hand, a larger number of scalarizations allows to find trade-off solutions that are not found with a smaller number of scalarizations. Given these results, among settings that require

Table 4. Average computation time and standard deviation for CW-step and PLS

Objectives	Instance Size	CW-step avg.	CW-step sd.	PLS avg.	PLS sd.
$(C_{\max}, \sum C_i)$	50x20	0.20	0.02	2.40	0.75
	100x20	1.56	0.40	75.04	32.53
$(C_{\max}, \sum w_i T_i)$	50x20	0.37	0.03	7.43	2.11
	100x20	2.51	0.42	202.71	50.51
$(\sum C_i, \sum w_i T_i)$	50x20	0.34	0.04	8.99	1.71
	100x20	2.75	0.34	373.15	87.44

roughly the same computation time, there is no single combination of settings that produces the best results overall among all objectives and instances.

Double TPLS. We denote as Double TPLS (DTPLS) the following strategy. First, the scalarizations go sequentially from one objective to the other one, as in the usual TPLS. Then, another sequence of scalarizations is performed starting from the second objective back to the first one. To introduce more variability, the weight vectors used in the first TPLS pass are alternated with the weight vectors used for the second TPLS pass. We compared this DTPLS strategy with the simple TPLS using 30 scalarizations of 1000 iterations. Although we found on several instances strong differences, these differences were not consistent in favor of advantages of DTPLS over TPLS or vice versa. This gives some evidence that the two strategies do not behave the same, but we left it for further research to investigate which instances features may explain the observed differences.

TPLS + PLS vs. TPLS + CW-step. As a final step, we compare the performance tradeoffs incurred by either running PLS or the CW-step to the archive of solutions returned by TPLS. For all instances, we generated 10 initial sets of solutions by running TPLS with 30 scalarizations of 1000 iterations each. Then, we independently apply to these initial sets the CW-step and PLS, both using the exchange and the insertion neighborhoods. In other words, each method starts from the same set of initial solutions in order to reduce variance.

Table 4 gives the additional computation time that is incurred by PLS and the CW-step after TPLS has finished. The results clearly show that the CW-step incurs only a minor overhead with respect to TPLS, while PLS requires considerably longer times, especially on larger instances. Moreover, the times required to terminate PLS are much lower than when seeding it with only two very good solutions (compare with Table 2). With respect to solution quality, Figure 4 compares TPLS versus TPLS+CW-step (top), and TPLS+CW step versus TPLS+PLS (bottom). As expected, the CW-step is able to slightly improve the results of TPLS, while PLS produces much better results. In summary, if computation time is very limited, the CW-step provides significantly better results at almost no computational cost; however, if enough time is available, PLS improves much further than the sole application of the CW-step.

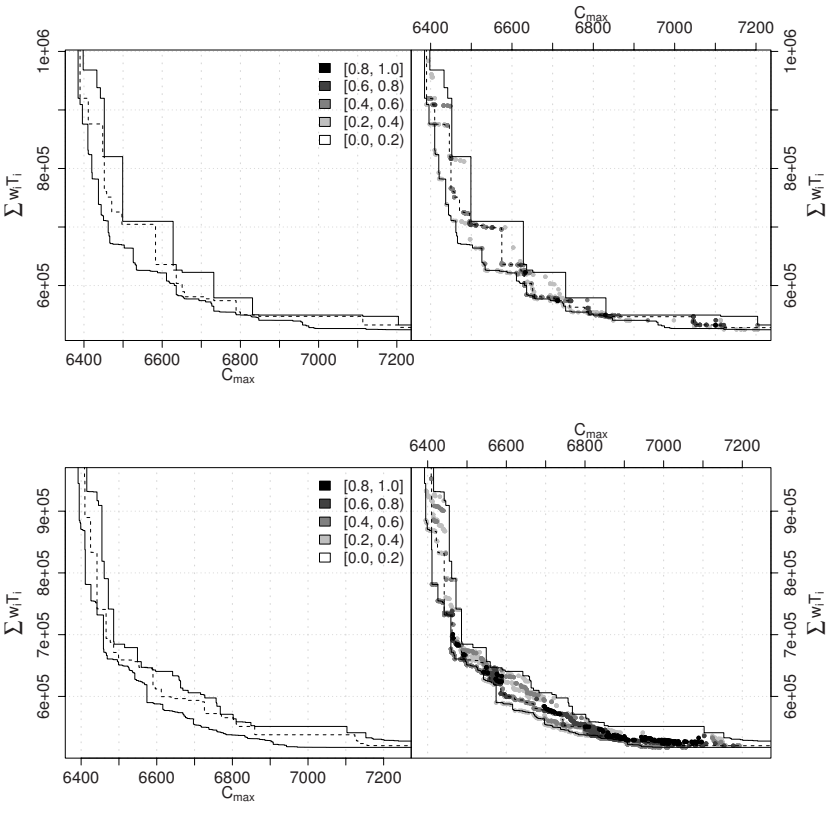

Fig. 4. EAF differences between (top) simple TPLS vs. TPLS + CW-step, and (bottom) TPLS + CW-step vs. TPLS + PLS. Objectives are $C_{\max}$ and $\sum w_i T_i$.

4.3 Comparison to Existing Algorithms

In order to compare our algorithm with existing work, we used the benchmark of Minella et al. [5], which is Taillard's benchmark set augmented with due dates. In their review, the authors compare 23 heuristics and metaheuristics using biobjective combinations of makespan ($C_{\max}$), sum of flowtime ($\sum C_i$), and total tardiness ($\sum T_i$). They also provide the best-known nondominated solutions obtained across 10 runs of each of the 23 algorithms. We use these sets as reference sets to compare with our algorithm. As the reference sets are given for the total (not weighted) tardiness criterion, we slightly modified our algorithm by setting all the priorities to one ($w_i = 1$).

In particular, we compare our results with the reference sets given for a computation time of 200 seconds and corresponding to instances from `ta081` to `ta090` (size `100x20`). These reference sets were obtained on an Intel Dual Core E6600 CPU running at 2.4 Ghz. By comparison, our algorithms were run on a Intel Xeon E5410 CPU running at 2.33 Ghz with 6MB of cache size, under Cluster

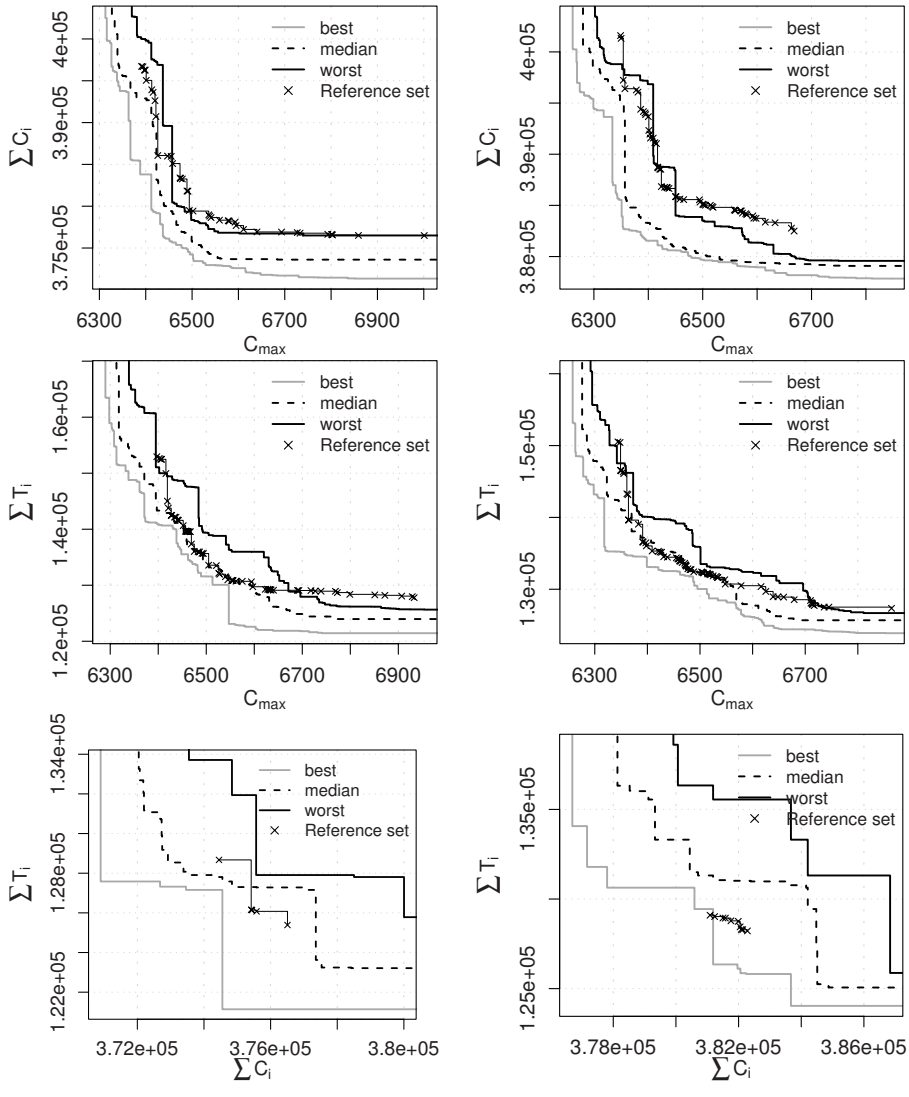

Fig. 5. Comparison of our algorithm against the reference set for objectives $C_{\max}$ and $\sum C_i$ (top), $C_{\max}$ and $\sum T_i$ (middle), and $\sum C_i$ and $\sum T_i$ (bottom) on instances `DD_Ta081` (left) and `DD_Ta082` (right)

Rocks Linux. Both machines result in approximately similar speed, however, to be conservative, we decided to round down the quality of our results by using only 150 CPU seconds. For our algorithms, we used the following parameter settings. The two extreme solutions are generated by running IG for 10 seconds each. Then TPLS starts from the solution obtained for the first objective and runs 14 scalarizations of 5 seconds each. Finally, we apply PLS in the exchange

and insertion neighborhoods and stop it after 60 CPU seconds. We repeat each run 10 times with different random seeds.

For each instance, we compare the best, median and worst attainment surfaces obtained by our algorithm with the corresponding reference set. Figure 5 shows results for the three objective combinations. In most cases, the median attainment surface of our algorithm is very close to (and often dominates) the reference set obtained by 10 runs of 23 algorithms, each run using 200 CPU seconds. Moreover, the current state-of-the-art algorithms for these problems are among these algorithms. Therefore, we conclude that our algorithm is clearly competitive and probably superior to the current state-of-the-art for these problems. All the additional plots are available online at `http://iridia.ulb.ac.be/supp/IridiaSupp2009-004`

5 Conclusions

In this paper, we have studied algorithmic components of the TPLS and PLS algorithms for three biobjective permutation flowshop problems, and we proposed a hybrid, high-performing SLS algorithm for these problems.

The final experimental results have shown that our SLS algorithms are able to significantly improve upon the reference sets of the nondominated solutions that have been obtained during an extensive experimental study of 23 algorithms for the same biobjective problems. These and other recent results in the literature [7,19] suggest that hybrid algorithms combining the TPLS and PLS frameworks have a large potential to improve upon the current state-of-the-art in multiobjective optimization.

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. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate.

References

1. Ruiz, R., Maroto, C.: A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 165, 479–494 (2005)
2. Liao, C.J., Tseng, C.T., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research 34(10), 3099–3111 (2007)
3. Tsenga, L.Y., Lin, Y.T.: A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 198(1), 84–92 (2009)
4. Vallada, E., Ruiz, R., Minella, G.: Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. Computers & Operations Research 35(4), 1350–1373 (2008)
5. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20(3), 451–471 (2008)

6. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3), 2033–2049 (2007)
7. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 479–493. Springer, Heidelberg (2003)
8. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation. LNEMS, vol. 535, pp. 177–200. Springer, Heidelberg (2004)
9. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
10. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)
11. Varadharajan, T.K., Rajendran, C.: A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. European Journal of Operational Research 167(3), 772–795 (2005)
12. Paquete, L., Stützle, T.: Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In: Gonzalez, T.F. (ed.) Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/CRC (2007); 29–1—29–15
13. Nawaz, M., Enscore Jr., E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA 11(1), 91–95 (1983)
14. Taillard, É.D.: Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research 47(1), 65–74 (1990)
15. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HM 2007. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
16. Taillard, É.D.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2), 278–285 (1993)
17. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.: Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 213–225. Springer, Heidelberg (2001)
18. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. Technical Report TR/IRIDIA/2009-015, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (May 2009)
19. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics (to appear, 2009), doi:10.1007/s10732-009-9103-9