Archive-aware Scalarisation-based Multi-Objective Local Search

For a Bi-objective Permutation Flowshop Problem

Aymeric Blot[1], Manuel López-Ibáñez[2],
Marie-Éléonore Kessaci[1], and Laetitia Jourdan[1]

[1] Université de Lille, CNRS, UMR 9189 – CRIStAL
{aymeric.blot,mkessaci,laetitia.jourdan}@univ-lille.fr
[2] Alliance Manchester Business School, University of Manchester
manuel.lopez-ibanez@manchester.ac.uk

Abstract. Given the availability of high-performing local search (LS) for single-objective (SO) optimisation problems, one successful approach to tackle their multi-objective (MO) counterparts is scalarisation-based local search (SBLS). SBLS strategies solve multiple scalarisations, i.e., aggregations of the multiple objectives into a single scalar value, with varying weights. They have been shown to work specially well as the initialisation phase of other types of multi-objective local search, such as Pareto local search (PLS). A major drawback of existing SBLS strategies is that the underlying SO optimiser is unaware of the MO nature of the problem and only returns a single solution, discarding any intermediate solutions that may be of interest.
We propose here two new SBLS strategies (ChangeRestart and ChangeDirection) that overcome this drawback by augmenting the underlying SO-LS method with an archive of nondominated solutions that is used to dynamically update the scalarisations. The new strategies produce better results on the bi-objective permutation flowshop problem than other five SBLS strategies from the literature, not only on their own but also when used as the initialisation phase of PLS.

Keywords: Flowshop scheduling · Local search · Heuristics · Multi-objective optimisation · Combinatorial optimisation.

1 Introduction

Multi-objective (MO) local search methods [11, 7, 5] are usually classified into two types. Scalarisation-based local search (SBLS) strategies aggregate the multiple objectives into a single (scalar) one by means of weights, and use single-objective (SO) local search to tackle each scalarised problem. Dominance-based local search (DBLS) strategies search the neighbourhood of candidate solutions for (Pareto) dominating or nondominated solutions. Successful algorithms for MO combinatorial optimisation problems often hybridise both strategies by generating a set of high-quality solutions by means of SBLS, and further improving this set by applying a DBLS method [3, 7, 8, 5].

Various SBLS strategies have been proposed in the literature that mainly differ in the sequence of weights explored during the search and the starting solution for solving each scalarisation. The simplest method, henceforth called Restart [10], uses a uniform set of weights and starts each scalarisation from a randomly (or heuristically) generated solution. More advanced strategies, such as AdaptiveAnytime [4], dynamically compute the next weight and choose a starting solution among the best ones found so far with the goal of closing the largest "gap" in the current Pareto front approximation.

First, we propose to augment the single-objective local search that solves the scalarisations with an archive of nondominated solutions, such that they are able to return more than one solution and make these solutions visible to the overall SBLS strategy and to the other local search runs solving other scalarisations. Then, we propose two new SBLS strategies. With ChangeRestart, we subdivide the time granted to solve each scalarisation in multiple steps, and use intermediary solutions to restart each local search run when it falls behind. With ChangeDirection, we further improve ChangeRestart by changing not only the starting solution of a local search run, but also the weight that defines the scalarisation being solved. As a case study, we focus on a bi-objective variant of the permutation flowshop scheduling problem (PFSP), which has been used previously as a benchmark for MO local search [3].

This paper is organised as follows. Section 2 describes classical SBLS strategies, together with the PFSP variant on which they have been previously applied. Section 3 proposes to augment single-objective local search with a nondominated archive to be used within SBLS strategies; and Section 4 proposes two new SBLS strategies. The experimental setup is then detailed in Section 5, whose results are discussed in Section 6. Section 7 summarises the main conclusions.

2 Background

2.1 Scalarisation-based Local Search (SBLS)

In multi-objective combinatorial optimisation problems, we have a set of feasible solutions S, where each solution may be evaluated according to a vector of M objectives $\boldsymbol{f}(s) = (f_1(s), \ldots, f_M(s))$. Without a priori information, candidate solutions are usually compared in terms of Pareto dominance: s_1 dominates s_2 iff $\forall i = 1, \ldots, M,\ f_i(s_1) \leq f_i(s_2)$ and $\exists j,\ f_j(s_1) < f_j(s_2)$. The goal becomes to find, or approximate as well as possible, the Pareto-optimal set, i.e., the set of solutions $S^* \subset S$ that are not dominated by any other solution in S. The image of the Pareto-optimal set in the objective space is called the Pareto front.

A MO problem can be transformed into a SO one by *scalarising* it, for example, by means of weighted sum. For simplicity, we will focus on the bi-objective ($M = 2$) case in the following. Given a problem with two objectives $\boldsymbol{f}(s) = (f_1(s), f_2(s))$ and a normalised weight vector $\boldsymbol{\lambda} = (\lambda, 1 - \lambda)$, where $\lambda \in [0, 1] \subset \mathbb{R}$, the corresponding scalarised problem (*scalarisation*) is computed as $f_\lambda(s) = \lambda \cdot f_1(s) + (1 - \lambda) \cdot f_2(s)$. An optimal solution of this SO scalarisation

is a Pareto-optimal solution of the MO problem, thus multiple Pareto-optimal solutions (although maybe not all) may be obtained by solving multiple scalarisations with different weights. The main advantage of solving scalarisations instead of the original MO problem is that, very often, highly effective and efficient local search algorithms exist for the single-objective case. SBLS approaches are conceptually related to decomposition-based algorithms (e.g., MOEA/D [14]).

Literature SBLS strategies differ in how weights are generated and which solution is used as the starting point of each local search run LS_λ solving f_λ.

Perhaps the simplest restarting strategy consists in generating a set of uniformly distributed weights and start each LS_λ run from a randomly or heuristically generated solution.

TPLS. In the simplest version of TPLS [10], one high-quality solution is generated by optimising just the first objective. On a second phase, a sequence of scalarisations of the problem, with weights that increasingly favours the second objective, are tackled by running LS_λ, thus generating solutions *along* the Pareto frontier from the first to the second objective. Moreover, each run of LS_λ starts from the best solution found for the previous scalarisation. This strategy is called **1to2** or **2to1** depending on which objective is optimised in the first phase. The 1to2 and 2to1 strategies tend to produce better solutions towards the starting objective. In order to avoid this bias, an alternative strategy (**Double**) uses half of the weights for 1to2 and the other half for 2to1 [10, 4]

AdaptiveAnytime. The above TPLS strategies can lead to uneven exploration of the objective space and poorly distributed approximation of the Pareto front if the problem is not fairly regular in terms of difficulty and the Pareto front is roughly symmetric for all scalarising directions. Similar poor results will also be obtained if the algorithm is terminated before tackling the predefined number of scalarisations. The adaptive anytime TPLS (AdaptiveAnytime) strategy was proposed to address these issues [4]. Similar to TPLS, a first phase generates one high-quality solution for each individual objective (like in Double) and a second phase solves a sequence of scalarisations. AdaptiveAnytime maintains a set G of "*gaps*" in the current approximation to the Pareto front, where each gap is a pair of solutions that are neighbours in the objective space, i.e., no other solution exists within the hyper-cube defined by them, and the size of the gap is the volume of this hyper-cube. The most successful variant of AdaptiveAnytime solves two scalarisations at each step, by first finding the largest gap in G, e.g., (s_1, s_2), with $f_1(s_1) < f_1(s_2)$, then computing:

$$\begin{cases} \lambda_1 = \lambda - \theta \cdot \lambda \\ \lambda_2 = \lambda + \theta \cdot (1 - \lambda) \end{cases} \text{ where } \lambda = \frac{f_2(s_1) - f_2(s_2)}{f_2(s_1) - f_2(s_2) + f_1(s_2) - f_1(s_1)} \quad (1)$$

and $\theta \in [0, 1]$ is a parameter that biases λ_1 towards the first objective and λ_2 towards the second objective; and, finally, solving f_{λ_1} starting from s_1 and f_{λ_2} starting from s_2. The solution returned by solving each scalarisation is used to update G, by removing any dominated solutions and updating the corresponding gaps. Thus, each step of the AdaptiveAnytime strategy tries to reduce the size of the largest gap and adapt the weights to the shape of the current front.

2.2 Bi-objective Permutation Flowshop Scheduling

The above SBLS strategies have been tested on various bi-objective permutation flowshop scheduling problems (PFSP) [4] and AdaptiveAnytime was later used as the initialisation phase of the state-of-the-art MO local search [3].

The PFSP is among the best-known problems in the scheduling literature, since it models several typical problems in manufacturing. Given a set of n jobs to be processed sequentially on m machines, where each job requires a different processing time on each machine, the goal is to find a permutation of the jobs that optimises particular objectives, such that all the jobs are processed in the same order on all machines, and the order of the machines is the same for all jobs. In this paper, we focus on the bi-objective variant (bPFSP) that minimises the makespan, that is, the completion time of the last job, and the total flowtime, that is, the sum of completion times of all jobs.

3 Archive-aware SBLS Strategies

Traditional SBLS strategies, such as the ones described above (Restart, 1to2, 2to1, Double and AdaptiveAnytime), use a SO local search to find a new solution optimised for a given scalarisation. Each local search run (LS_λ) starts from a given solution and returns the single best solution found for that particular scalarisation. Any other solution found during the run is usually discarded, even those solutions that are not dominated by the solution returned.

We propose to augment the SO local search with an *archive* that keeps track of nondominated solutions found while solving a scalarisation, in order to preserve solutions that may be optimal for the MO problem, even if they are not for the particular scalarisation. Since such intermediary solutions are fully evaluated to compute their scalarised value, keeping an archive of these solutions only adds the computational overhead of updating the archive. In practice, adding every solution evaluated to the archive would require too much time. Instead, we only update the archive when a new solution replaces the current one.

As an example of SO local search, let us consider iterated greedy (IG) [12]. At each iteration of IG, the current solution π is randomly destructed (e.g, by removing some jobs from it), heuristically reconstructed (e.g., by re-inserting the jobs in new positions), and the resulting solution may be further improved by another local search. An acceptance criterion replaces the current solution (π) with the new one if the latter is better or some other condition is met. In any case, if the new solution improves the best-so-far one ($\pi^\star$), the latter is replaced. The algorithm returns $\pi^\star$ once it terminates.

Our proposed archive-aware IG adds an archive of nondominated solutions (A) that is updated every time a better current solution is found, and returns the archive in addition to the best solution found. Any other SO local search used within SBLS strategies can be made archive-aware in a similar manner.

We propose straight-forward variants of the classical SBLS strategies that make use of such archive-aware SO local search and we denote such variants

with the suffix "$_{\mathsf{arch}}$". In $\mathsf{Restart}_{\mathsf{arch}}$, $\mathsf{1to2}_{\mathsf{arch}}$, $\mathsf{2to1}_{\mathsf{arch}}$ and $\mathsf{Double}_{\mathsf{arch}}$, each local search run produces an archive instead of a single solution. The resulting N^{scalar} archives are independent of each other until merged into a final archive. Thus, the search trajectory of these archive-aware SBLS variants is the same as their original counterparts, except for the overhead incurred by updating the archives. In the case of $\mathsf{AdaptiveAnytime}_{\mathsf{arch}}$, the archive returned by each local search run is immediately merged with the overall archive so that all solutions returned by the local search are used for computing the next largest gap.

4 New SBLS Strategies: ChangeRestart, ChangeDirection

In this section, we propose two new SBLS strategies that make use of intermediate solutions found while solving the scalarised problems, and that may also use an archive-aware single-objective local search.

4.1 ChangeRestart

We observed that the sub-spaces searched by running the SO local search for different values of λ often overlap, thus the best-so-far solution found for one scalarisation may be worse than the best-so-far solution found for another, when the latter solution is evaluated on the former scalarisation. Thus, the main idea behind ChangeRestart is to divide each local search run (LS_λ) into smaller steps and, at each step, decide to either continue the run until the next step or restart it from a new solution. When interrupted, LS_λ returns its best-so-far solution ($\pi^\star_\lambda$). Then, for each weight, we calculate the scalarised value of all solutions in the current nondominated archive A. If the best-so-far solution by a given scalarised run is actually worse than the best-so-far solution found by a different scalarised run, when the latter solution is evaluated on the scalarised problem being solved by the former run, then this former run restarts its search from the latter solution. By reducing the number of steps (N^{steps}), we can limit the computational overhead of recalculating the scalarised values of each solution in A for all weights. In particular, the time limit assigned to each LS_λ run is divided by N^{steps}. When $N^{\text{steps}} = 1$, ChangeRestart is identical to Restart.

In the archive-aware variant $\mathsf{ChangeRestart}_{\mathsf{arch}}$, each run of LS_λ returns a nondominated archive that is merged with the overall archive A.

Figure 1 shows possible executions of ChangeRestart and $\mathsf{ChangeRestart}_{\mathsf{arch}}$ for two scalarisations and three steps ($N^{\text{steps}} = 3$). Blue points (●) and red triangles (▲) show the initial solutions and the best solutions found after each step. These solutions are connected with arrows to show the trajectory followed by each run of LS_λ. Unfilled points (○) and triangles (△) show intermediary solutions in the archive after each step. For ChangeRestart (left), after the second step, the solution (a) found for $\lambda = 1$ has a worse value in the first objective than the solution (b) found for $\lambda = 0.5$. Thus, the local search for $\lambda = 1$ re-starts from solution b instead of a. For $\mathsf{ChangeRestart}_{\mathsf{arch}}$ (right), the local search re-starts instead from solution (c), as it has an even better value regarding objective f_1.

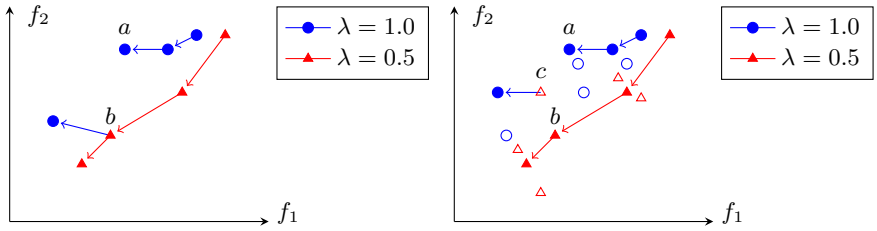

Fig. 1. Example runs of ChangeRestart (left) and $\text{ChangeRestart}_{\text{arch}}$ (right) ($N^{\text{steps}} = 3$).

4.2 ChangeDirection

The second SBLS strategy proposed here is ChangeDirection. While ChangeRestart is an extension of Restart, ChangeDirection is inspired by the more advanced AdaptiveAnytime. In AdaptiveAnytime, scalarisation weights are dynamically chosen according to the gaps in the current overall archive (A), in order to focus the search in the direction which will most improve the current approximation to the Pareto front. In ChangeDirection, as in ChangeRestart, the runs of LS_λ are also divided in a number of steps and, after each step, solutions from different scalarisations are merged into an overall archive A. However, instead of only updating the starting solution of each LS_λ run, the scalarisation weight λ is also updated. In other words, in addition to speeding up an LS_λ run by re-starting from a better initial solution, the scalarisation direction of LS_λ may be changed to focus on the largest gap in the current approximation front. In particular, a scalarisation weight λ is replaced by another weight whenever the best-so-far of LS_λ is worse, according to f_λ, than a solution returned by another local search run. In that case, the computational resources allocated to searching in the direction given by λ could be better used in searching on a different direction.

ChangeDirection only differs from ChangeRestart in the deletion and replacement of scalarisation directions. Thus, we will only explain those novel parts. First, we delete those scalarisation weights for which the best solution found in the last run of LS_λ is worse, according to the same scalarisation f_λ, than a solution in A. Then, following the strategy of AdaptiveAnytime explained earlier, the gaps in the current approximation front are computed and new weights are generated from the largest gap to replace the deleted ones. In particular, two weights are generated from each gap (Eq. 1) until all deleted weights are replaced. When only one additional weight is needed, it is chosen randomly between the two weights produced by the gap. The new scalarisations then start from the solutions constituting the sides of the gap. If all gaps are used and additional weights are needed, they are drawn uniformly at random within $[0, 1]$ and initial solutions are taken uniformly at random from A. Finally, as in ChangeRestart, each LS_λ either re-starts from a new initial solution if its scalarisation was introduced in this step, or continues from its current solution, otherwise. As previously, in the archive-aware variant $\text{ChangeDirection}_{\text{arch}}$, each run of LS_λ returns a nondominated archive that is merged with the overall archive A.

5 Experimental Setup

We wish to investigate not only whether the new proposed SBLS strategies work well on their own, but also if they provide a good initial set for a dominance-based local search (DBLS) algorithm. Thus, we use the various SBLS strategies as the initialisation phase of a hybrid of SBLS+DBLS algorithm, where a SBLS strategy generates an initial approximation front that is further improved by a DBLS strategy, in our case, an iterated variant of Pareto local search (IPLS). We use IG as the single-objective local search (LS_λ) and the algorithms are evaluated on the bi-objective PFSP (bPFSP). In this section, we explain the details of the our experimental setup.

bPFSP Instances. As a benchmark, we consider the well-known Taillard instances [13]. In the following, we focus on 80 instances with 20, 50, 100 and 200 jobs, and 10 and 20 machines, 10 instances being used for each valid combination.

Iterated Greedy (IG). The single-objective local search used by the SBLS strategies is Iterative greedy (IG) [12], which is a state-of-the-art algorithm for the single-objective PFSP. The particular IG variant and parameter settings are directly taken from the bPFSP literature [3]. For the archive-aware SBLS strategies, we augment this IG variant with an archive as explained in Section 3.

Iterated Pareto Local Search (IPLS). As the DBLS component of our hybrid SBLS+DBLS algorithm, we consider an iterated variant of Pareto local search (PLS) [9], as it was shown that even simple perturbations could benefit PLS algorithms [2]. Our iterated PLS (IPLS) extends the PLS used in [3] by perturbing the archive when the latter converges to a Pareto local optimal set, using the generalised framework of [1]. The perturbation used creates a new archive by taking every current solution and replacing it with one of its neighbours, taken uniformly at random, three times in a row; dominated solutions from this new set are then filtered. As the neighbourhood of PLS, we use the union of the exchange and insertion neighbourhoods [6], in which two positions of two jobs are swapped and one job is reinserted at another position, respectively.

Termination criteria. The termination criterion of algorithms applied to the bPFSP is usually set as maximum running time proportional to the instance size. While previous works have used a linear proportion of both the number of jobs n and the number of machines m (e.g., $0.1 \cdot n \cdot m$ CPU seconds [3]), we use a maximum running time for the hybrid SBLS+IPLS of $0.002 \cdot n^2 \cdot m$ CPU seconds. Indeed, the total number of solutions grows exponentially and the typical size of permutation neighbourhoods grows quadratically, making a linear running time less relevant. The coefficient 0.002 was chosen to match the linear formula for $n = 50$ and $m = 20$. As for the termination criterion PLS itself, we use the number of successive iterations without improvement, set here to n, before forcing a perturbation of the archive.

The SBLS strategies are limited to 25% of the total running time of the hybrid SBLS+IPLS, and the remaining 75% is allocated to IPLS. The main parameter of the SBLS strategies is the number of scalarisations (N^{scalar}), that is, the number of runs of IG executed in addition to two individual runs for each of the two single objectives. Following [3], we perform longer runs of IG for the

two single objectives ($IG_{\{1,2\}}$) than for the other scalarisations (IG_λ), with the time assigned to $IG_{\{1,2\}}$ being 1.5 times the time assigned to IG_λ. As more time is allocated to $IG_{\{1,2\}}$ than to IG_Λ, their respective running time budgets are $1.5/(N^{\text{scalar}}+3)$ and $1/(N^{\text{scalar}}+3)$ of the total time assigned to the SBLS strategy. In the case of ChangeRestart and ChangeDirection, the maximum runtime of each IG is further divided by N^{steps}.

The following experiments are separated in three successive phases. First, we analyse the effect of using an archive-aware IG on the five SBLS strategies from the literature (Restart, 1to2, 2to1, Double, and AdaptiveAnytime). Second, we compare all these SBLS variants with the new SBLS strategies proposed here (ChangeRestart and ChangeDirection), including their archive-aware counterparts. Finally, we analyse other possible setting for the parameters N^{scalar} and N^{steps}. Unless stated otherwise, ChangeRestart and ChangeDirection use $N^{\text{steps}} = 20$; all SBLS strategies use a fixed value of $N^{\text{scalar}} = 12$; and both AdaptiveAnytime and ChangeDirection use $\theta = 0.25$ for Eq. 1 [3].

In all cases, we run the hybrid SBLS+IPLS and we save the archive returned by the SBLS strategies (before IPLS) and the final archive (after IPLS). Each experiment is repeated 5 times, using independent random seeds, on each of the 80 Taillard instances. All replications use the same seeds on the same instances. All the experiments have been conducted on Intel Xeon E5-2687W V4 CPUs (3.0GHz, 30MB cache, 64GB RAM).

Results are evaluated according to both the hypervolume and the additive-ε indicators [15]. Indicator values have been computed independently on every instance aggregating all results generated for the instance and scaling both objectives to a 0–1 scale in which 0 (1) corresponds to the minimum (maximum) objective value reached by any solution. The hypervolume variant $1-HV$ is used, with 0 corresponding to the maximum hypervolume, so that both indicators are to be minimised. The reference point used for computing the hypervolume indicator is $(1.0001, 1.0001)$. The reference set for computing additive-ε indicator is the set of nondominated solutions from all aggregated results for each instance.

6 Experimental Results

6.1 Known SBLS strategies vs. their archive-aware variants

First, we compare the five SBLS strategies from the literature with their archive-aware variants. Figure 2 shows the mean hypervolume and additive-ε values obtained by each strategy before and after running IPLS. Averages are computed on the indicator values over all 80 bPFSP instances.

For both indicators, all the proposed archive-aware variants (in red) lead to improved quality before IPLS. After results are improved by IPLS, all archive-aware variants produce again better results than their original counter-parts, with the exception of AdaptiveAnytime. This is somewhat surprising and further analysis is needed to understand this behaviour. Interestingly, some of the archive-aware variants are able to outperform AdaptiveAnytime when their original variants are not.

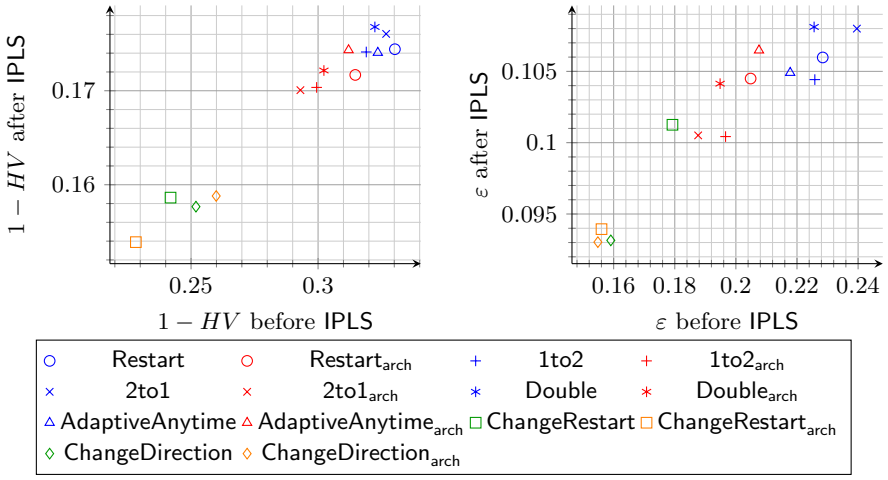

Fig. 2. Comparison of all SBLS strategies according to (left) mean hypervolume and (right) mean additive-ε.

6.2 Performance of Two New SBLS strategies

We now compare the newly proposed SBLS strategies (ChangeRestart and ChangeDirection) to the ones from the literature as well as their archive-aware variants.

In terms of hypervolume, all four new strategies achieve in average much better results than the strategies from the literature, with ChangeRestart$_{\text{arch}}$ achieving the best results both on its own and when further improved by IPLS. However, in terms of additive-ε, the non-archive-aware ChangeRestart strategy performs much worse than the three other new strategies, but still better than most strategies from the literature. Overall, the best strategy according to both indicators appears to be the archive-aware ChangeRestart strategy.

To validate these observations, Table 1 shows the results of a statistical analysis comparing all approaches, without averaging over benchmarks, for both hypervolume (top) and additive ε (bottom). For each benchmark set, and for all possible pairs of strategies, we conducted a statistical Wilcoxon test comparing their final quality (after IPLS) paired on 50 runs. A (✓) in the table indicates strategies for which there was no other strategy performing statistically better (with 95% confidence). In other words, within each row, all strategies with ✓are not statistically better to each other, while for those strategies without a ✓, there was at least one other strategy statistically better. As shown in the Table, the SBLS strategies from the literature are often outperformed by some other strategy, whereas their archive-aware variants are less often so, in particular on the smaller instances with 20 and 50 jobs. Finally, ChangeRestart$_{\text{arch}}$ and both variants of ChangeDirection are almost never outperformed, even on the largest instances, validating our previous observations.

Table 1. SBLS strategies not statistically outperformed by another strategy after IPLS step, using paired Wilcoxon tests (left: hypervolume; right: additive-ε)

	Restart	1to2	2to1	Double	AdaptiveAnytime	$\text{Restart}_{\text{arch}}$	$\text{1to2}_{\text{arch}}$	$\text{2to1}_{\text{arch}}$	$\text{Double}_{\text{arch}}$	$\text{AdaptiveAnytime}_{\text{arch}}$	ChangeRestart	$\text{ChangeRestart}_{\text{arch}}$	ChangeDirection	$\text{ChangeDirection}_{\text{arch}}$	Restart	1to2	2to1	Double	AdaptiveAnytime	$\text{Restart}_{\text{arch}}$	$\text{1to2}_{\text{arch}}$	$\text{2to1}_{\text{arch}}$	$\text{Double}_{\text{arch}}$	$\text{AdaptiveAnytime}_{\text{arch}}$	ChangeRestart	$\text{ChangeRestart}_{\text{arch}}$	ChangeDirection	$\text{ChangeDirection}_{\text{arch}}$
20×10						✓	✓		✓	✓		✓	✓	✓						✓	✓		✓	✓		✓	✓	✓
20×20						✓	✓	✓	✓	✓		✓	✓	✓						✓	✓	✓	✓	✓		✓	✓	✓
50×10		✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
50×20	✓	✓	✓			✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓	✓
100×10											✓	✓	✓	✓									✓	✓		✓	✓	✓
100×20		✓			✓		✓				✓	✓	✓	✓		✓			✓		✓	✓		✓		✓	✓	✓
200×10											✓	✓	✓	✓											✓	✓	✓	✓
200×20											✓	✓	✓									✓			✓	✓	✓	✓

6.3 Analysis of Parameters N^{scalar} and N^{steps}

SBLS strategies strongly depends on the number of scalarisations they use. Our choice of $N^{\text{scalar}} = 12$ was motivated by previous studies claiming that few (10, 20) scalarisations should be preferred [3]. Figure 3 (left) shows for the 14 previous strategies the final performance regarding both hypervolume and additive ε indicators, for both parameter values of $N^{\text{scalar}} \in \{6, 12\}$ scalarisations, in order to see the impact of archives-aware mechanisms when using very few scalarisations.

We can see that for all strategies, both with and without archiving, using 12 scalarisations improve in average significantly the performance regarding hypervolume and slightly the one regarding the ε indicator, hinting that even with archiving a sufficient number of scalarisations are still required.

The number of steps, i.e. how many times we can restart the scalarisations, is at the core of the two new SBLS strategies we propose. Figure 3 (right) shows for all variants of the ChangeRestart and ChangeDirection strategies the impact of the parameter N^{steps}, for values of $N^{\text{steps}} = 1$ (equivalent to Restart) and $N^{\text{steps}} \in \{2, 5, 10, 15, 20, 25\}$, using the final performance regarding both the hypervolume and ε indicators. Marks indicate the value of N^{steps}, while colours indicate the strategy.

As the figure shows, at first increasing the number of steps from one largely improves the quality of the results. Increasing the number of steps further specially benefits the archive-aware variants and in particular, $\text{ChangeDirection}_{\text{arch}}$. However, for large values of N^{steps}, the quality improvements stop or, in several cases, worsen. Thus, it appears that even larger values would not improve the results reported here.

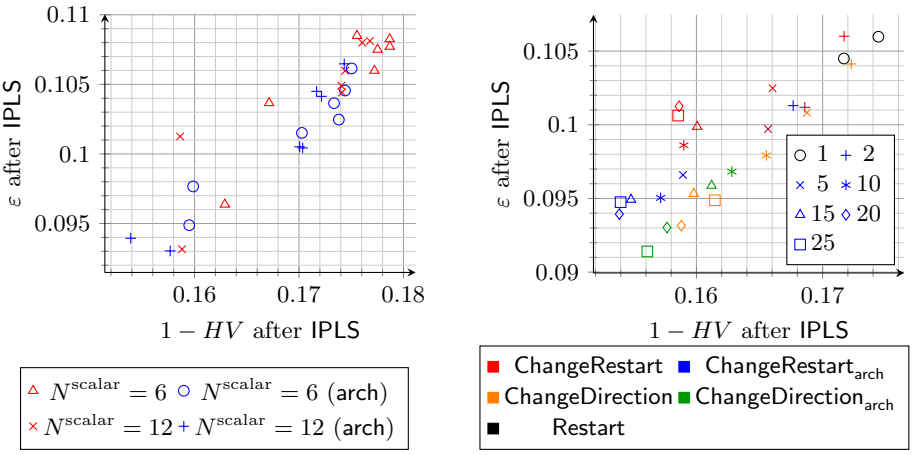

Fig. 3. Impact of the number of scalarisations (left) and the number of steps (right)

7 Conclusion

This paper proposes and evaluates two complementary ways of augmenting scalarisation-based local search (SBLS) strategies by making the underlying single-objective local search aware of the multi-objective nature of the problem. Our first proposal adds an archive of nondominated solutions to the single-objective local search. Our results showed that these archive-aware SBLS variants always improve over their original counterparts when ran on their own. Moreover, this improvement also shows for nearly all SBLS strategies when their results are further improved by means of iterated Pareto local search (IPLS).

Our second proposal was to divide each run of the single-objective local search into a number of smaller steps and, at each step, restart scalarisations that produce poor results. We proposed two SBLS strategies that differ on what is changed by the restart. In ChangeRestart, the local search for solving a scalarisation is restarted from the best-known solution for that scalarisation problem. This solution was possibly generated when solving a different scalarisation. In ChangeDirection, not only the starting solution, but also the weight that defines the scalarisation problem itself being solved are both updated in order to re-focus this particular run on the largest gap of the current approximation front.

Our experimental results show that these two new SBLS strategies outperform five classical SBLS strategies from the literature, even when the latter are using and archive-aware local search. In particular, ChangeDirection produces consistently the best results, either on its own or when used as the initialisation phase of a hybrid SBLS+IPLS algorithm, which suggests that the new strategies may lead to new state-of-the-art results for the bi-objective permutation flowshop [3], and other problems. An additional benefit of ChangeDirection is that it maintains the adaptive behaviour of AdaptiveAnytime, while it also may perform N^{scalar} local search runs in parallel between steps.

Future work will analyse in more detail the interaction between the new SBLS strategies and the archive-aware SO local search. A more comprehensive analysis of the effect of the N^{scalar} and N^{steps} parameters would be needed to understand their interactions with problem features. We would also hope to evaluate the new proposals in terms of their anytime behaviour [4]. Finally, a further comparison, including various common speedups not included here as we focused on archive-aware mechanisms, would be required for a fair comparison with other state-of-the-art algorithms.

References

1. Blot, A., Jourdan, L., Kessaci-Marmion, M.E.: Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem. GECCO 2017, pp. 227–234. ACM Press (2017).
2. Drugan, M.M., Thierens, D.: Path-guided mutation for stochastic Pareto local search algorithms. In: PPSN XI, LNCS, vol. 6238, pp. 485–495. Springer (2010).
3. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. COR **38**(8), 1219–1236 (2011).
4. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Improving the anytime behavior of two-phase local search. AMAI **61**(2), 125–154 (2011).
5. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Combining two search paradigms for multi-objective optimization: Two-Phase and Pareto local search. In: Hybrid Metaheuristics, vol. 434, pp. 97–117. Springer Verlag (2013).
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Anytime Pareto local search. EJOR **243**(2), 369–385 (2015).
7. Liefooghe, A., Humeau, J., Mesmoudi, S., Jourdan, L., Talbi, E.G.: On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. JOH **18**(2), 317–352 (2011).
8. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. ITOR **19**(4), 495–520 (2012).
9. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Metaheuristics for Multiobjective Optimisation, LNMES, vol. 535, pp. 177–200. Springer (2004)
10. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: EMO 2003, LNCS, vol. 2632, pp. 479–493. Springer (2003)
11. Paquete, L., Stützle, T.: Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In: Handbook of Approximation Algorithms and Metaheuristics, pp. 29–1—29–15. Chapman & Hall/CRC (2007)
12. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. EJOR **177**(3), 2033–2049 (2007)
13. Taillard, É.D.: Benchmarks for basic scheduling problems. EJOR **64**(2), 278–285 (1993)
14. Zhang, Q. and Li, H. (2007): MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE TEC*, 11(6):712–731.
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE TEC **7**(2), 117–132 (2003)