Ann Math Artif Intell (2011) 61:125–154
DOI 10.1007/s10472-011-9235-0

Improving the anytime behavior of two-phase local search

Jérémie Dubois-Lacoste · Manuel López-Ibáñez · Thomas Stützle

Published online: 1 June 2011

Abstract Algorithms based on the two-phase local search (TPLS) framework are a powerful method to efficiently tackle multi-objective combinatorial optimization problems. TPLS algorithms solve a sequence of scalarizations, that is, weighted sum aggregations, of the multi-objective problem. Each successive scalarization uses a different weight from a predefined sequence of weights. TPLS requires defining the stopping criterion (the number of weights) a priori, and it does not produce satisfactory results if stopped before completion. Therefore, TPLS has poor "anytime" behavior. This article examines variants of TPLS that improve its "anytime" behavior by adaptively generating the sequence of weights while solving the problem. The aim is to fill the "largest gap" in the current approximation to the Pareto front. The results presented here show that the best adaptive TPLS variants are superior to the "classical" TPLS strategies in terms of anytime behavior, matching, and often surpassing, them in terms of final quality, even if the latter run until completion.

Keywords Multi-objective · Anytime algorithms · Two-phase local search

Mathematics Subject Classification (2010) 68T20

J. Dubois-Lacoste (✉) · M. López-Ibáñez · T. Stützle
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
e-mail: jeremie.dubois-lacoste@ulb.ac.be

M. López-Ibáñez
e-mail: manuel.lopez-ibanez@ulb.ac.be

T. Stützle
e-mail: stuetzle@ulb.ac.be

1 Introduction

Two-phase local search (TPLS) [19] is a stochastic local search (SLS) method for tackling multi-objective combinatorial optimization problems in terms of Pareto-optimality [20]. TPLS is an essential component of state-of-the-art algorithms for a number of problems such as the bi-objective traveling salesman problem (bTSP) [17], its variant with three objectives [21], and the bi-objective permutation flow-shop scheduling problem (bPFSP) [5, 8]. In these algorithms, the solutions obtained by TPLS are further improved by applying a Pareto Local Search [22].

The central idea of TPLS is to start with a high quality solution for a single objective (first phase) and then to solve a sequence of scalarizations of the multi-objective problem (second phase). In TPLS, each successive scalarization uses a slightly different weight and starts from the best solution found by the previous scalarization. Originally, the set of weights and, hence, the number of scalarizations, is defined before the execution of TPLS in order to equally distribute the computational effort along the Pareto front [19]. This strategy implies that stopping TPLS at an arbitrary time before it has performed the predefined number of scalarizations would produce a poor approximation to the Pareto front in some regions. In this sense, TPLS does not have a good *anytime* behavior.

Anytime algorithms [24] aim at producing an as high as possible performance at any moment of their execution, without assuming a predefined termination criterion. Recently, we proposed two variants of TPLS that improve its *anytime* performance [6]. The first variant, Regular Anytime TPLS (RA-TPLS), uses a regular distribution of the weight vectors, equally distributing the effort in all directions of the objective space. We tested this variant on two bPFSPs, where the objectives result in a different algorithm behavior for the underlying single-objective algorithms. For these problems, we found that an *adaptive* TPLS, which adapts the weights in dependence of the shape of the Pareto front, performed better than a regular strategy.

In this paper, we re-examine RA-TPLS and the adaptive TPLS algorithms for the bTSP and the bPFSP. In addition, we propose new design alternatives of adaptive TPLS. First, we study the effect of performing one or two scalarizations per weight (with different seeds), and we give evidence when each approach may be beneficial. Second, we propose a new method to choose the region of the objective space where the search should be intensified, that is, how to define the largest gap in the current approximation of the Pareto front. Instead of using the Euclidean distance between solutions [6], this new method is based on an optimistic estimate of the improvement in the hypervolume indicator. Our experimental results show that the new method further improves the results of the adaptive TPLS strategy in the bTSP, no matter the shape of the front.

The main conclusion of our study is that the adaptive TPLS variants show a much better anytime behavior than the original TPLS algorithms, and the best performing adaptive variants typically return better quality approximations to the Pareto front, as indicated by the hypervolume indicator.

The paper is structured as follows. In Section 2, we introduce some formal definitions on bi-objective optimization and we present the original TPLS algorithms. In Section 3, we present our first proposal to improve the anytime property of TPLS. In Section 4, we propose an adaptive TPLS framework that not only satisfies

the anytime property, but that also adapts to the shape of the Pareto front to maximize the performance. We perform in Section 5 a detailed statistical analysis to compare all strategies. In Section 6, we propose the use of an optimistic estimation of the potential hypervolume contribution to direct the search of adaptive TPLS, and we show that it leads to further improvements in the bTSP. In Section 7, we graphically examine the differences in quality of the algorithms. We conclude in Section 8.

2 Preliminaries

In this paper, we focus on bi-objective combinatorial optimization problems and here we describe relevant background for the remainder of the paper.

2.1 Bi-objective combinatorial optimization

In bi-objective combinatorial optimization problems, candidate solutions are evaluated according to an *objective function vector* $\mathbf{f} = (f_1, f_2)$. Assuming, without loss of generality, that both objective functions must be minimized, the dominance criterion defines a partial order among objective vectors as follows. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$, we say that $\mathbf{u}$ *dominates* $\mathbf{v}$ ($\mathbf{u} \prec \mathbf{v}$) iff $\mathbf{u} \neq \mathbf{v}$ and $u_i \leq v_i,\ i = 1, 2$. When $\mathbf{u} \not\prec \mathbf{v}$ and $\mathbf{v} \not\prec \mathbf{u}$, we say that $\mathbf{u}$ and $\mathbf{v}$ are mutually *non-dominated*. For simplicity, we extend the dominance criteria to solutions, that is, a solution s dominates another one s' iff $\mathbf{f}(s) \prec \mathbf{f}(s')$. If no s' exists such that $\mathbf{f}(s') \prec \mathbf{f}(s)$, the solution s is called *Pareto-optimal*. In a bi-objective optimization problem where no *a priori* assumptions upon the decision maker's preferences are made, the problem becomes to find a set of feasible solutions that "minimize" $\mathbf{f}$ in the sense of Pareto optimality. Hence, the goal is to determine the set of all Pareto-optimal solutions, whose image in the objective space is called the *Pareto front*. Since this goal is in many cases computationally intractable [9], in practice the goal becomes to find the best possible approximation to the Pareto front within a specific time limit.

2.2 Two-phase local search

Two-phase local search (TPLS) [19] is a general algorithmic framework that, as the name suggests, is composed of two phases. In the first phase, a single-objective algorithm generates a high-quality solution for one of the objectives. This high-quality solution serves as the starting point of the second phase, where a sequence of *scalarizations*, that is, weighted sum aggregations of the multiple objective functions into single scalar functions, are tackled. Each scalarization uses the best solution found by the previous scalarization as the initial solution. TPLS will be successful if effective single-objective algorithms are available, and solutions that are close to each other in the solution space are also close in the objective space.

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 $\boldsymbol{\lambda} = (\lambda, 1 - \lambda)$,

Algorithm 1 Two-phase local search

```
1: π1 := SLS1()
2: π2 := SLS2()
3: Add π1, π2 to Archive
4: if 1to2 then
5:     π' := π1
6: else
7:     π' := π2
8: end if
9: for each weight λ do
10:    π' := SLSΣ(π', λ)
11:    Add π' to Archive
12: end for
13: RemoveDominated(Archive)
14: Output: Archive
```

$\lambda \in [0, 1] \subset \mathbb{R}$, and the scalar value of a solution s with objective function vector $\mathbf{f}(s) = (f_1(s), f_2(s))$ is computed as

$$f_\lambda(s) = \lambda \cdot f_1(s) + (1 - \lambda) \cdot f_2(s). \quad (1)$$

Depending on the sequence of weight vectors considered, there are two main TPLS strategies in the literature:

*Single direction (*1to2 *or* 2to1*)* The simplest way to define a sequence of scalarizations is to use a regular sequence of weight vectors from the first objective to the second or from the second objective to the first one. We call these alternatives *1to2* or *2to1*, depending on the direction followed. For example, the successive scalarizations in *1to2* are defined by the weights $\lambda_i = 1 - (i - 1)/(N_{\text{scalar}} - 1)$, $i = 1, \ldots, N_{\text{scalar}}$, where N_{scalar} is the number of scalarizations. (For simplicity, we henceforth refer to weight vectors by their first component only, since the second component can be easily derived from (1).) In *2to1* the sequence is reversed. Two drawbacks of this simple strategy are that (i) the direction chosen can give an advantage to the starting objective, that is, the Pareto front approximation will be better on the starting side; and that (ii) one needs to know in advance the computation time that is available in order to define appropriately the number of scalarizations and the time spent on each scalarization. Different from the original TPLS proposal [19], in our implementation we first generate a very good solution for each single objective problem because we have high performing algorithms for them and we want to be consistent with our other TPLS variants in Sections 3, 4 and 6. However, we use only one of the solutions as a starting solution for further scalarizations. Algorithm 1 gives the pseudo-code of the single direction TPLS. We denote by SLS_1 and SLS_2 the SLS algorithms to minimize the first and the second single objectives, respectively. SLS_Σ is the SLS algorithm to minimize the scalarized problem.

Double strategy We denote as Double TPLS (D-TPLS) [19] the strategy that first goes sequentially from one objective to the other one, as in the usual TPLS. Then, another sequence of scalarizations is generated starting from the second objective

back to the first one. This is, in fact, a combination of *1to2* and *2to1*, where half of the scalarizations are defined sequentially from one objective to the other, and the other half in the opposite direction. This approach tries to avoid the bias of a single starting objective. To introduce more variability, in our D-TPLS implementation, the weights used in the second TPLS pass are located in-between the weights used for the first TPLS pass. D-TPLS still requires to define the number of weights, and, hence, the computation time, in advance.

3 Regular anytime TPLS

The original strategy of TPLS, which is based on defining successive weight vectors with minimal weight changes, generates very good approximations to the areas of the Pareto front "covered" by the weight vectors [19, 21]. However, if TPLS is stopped prematurely, it leaves areas of the Pareto front unexplored. In this section, we present our first proposal to improve the anytime behavior of TPLS.

3.1 Regular anytime strategy

We have proposed a TPLS-like algorithm, called *regular anytime* TPLS (RA-TPLS), in which the weight for each new scalarization is defined in the middle of the interval of two previous consecutive weights [6]. This strategy provides a finer approximation to the Pareto front as the number of scalarizations increases, ensuring a fair distribution of the computational effort along the Pareto front and gradually intensifying the search. The set of weights is defined as a sequence of progressively finer "levels" of 2^{k-1} scalarizations (at level k) with maximally dispersed weights Λ_k in the following manner: $\Lambda_1 = \{0.5\}$, $\Lambda_2 = \{0.25, 0.75\}$, $\Lambda_3 = \{0.125, 0.375, 0.625, 0.875\}$, and so on. Successive levels intensify the exploration of the objective space, filling the gaps in the Pareto front. The two initial solutions minimizing each objective could be seen as level 0: $\Lambda_0 = \{0, 1\}$. Once RA-TPLS completes one level, the computational effort has been equally distributed in all directions. However, if the search stops before exploring all scalarizations at a certain level, the search would explore some areas of the Pareto front more thoroughly than others. In order to minimize this effect, RA-TPLS considers the weights within one level in a random order.

In order to be an alternative to TPLS, RA-TPLS starts each new scalarization from a solution obtained from a previous scalarization. In particular, the initial solution of the new scalarization (using a new weight) is one of the two solutions that were obtained using the two weight vectors closest to the new weight. The algorithm computes the weighted sum scalar values of these two solutions according to the new weight, and selects the one with the better value as the initial solution of the new scalarization.

The implementation of RA-TPLS requires three main data structures: L_i is the set of pairs of weights used in previous scalarizations, where i determines the level of the search; *Sd* is a set of potential initial solutions, each solution being associated with the corresponding weight that was used to generate it; *Archive* is the archive of non-dominated solutions.

Algorithm 2 describes RA-TPLS in detail. In the initialization phase, an initial solution is obtained for each objective using appropriate single-objective algorithms,

Algorithm 2 RA-TPLS

```
1: s_1 := SLS_1()
2: s_2 := SLS_2()
3: Add s_1, s_2 to Archive
4: L_0 := {(1, 0)}; L_i := ∅   ∀i > 0
5: Sd := {(s_1, 1), (s_2, 0)}
6: i := 0
7: while not stopping criterion met do
8:    (λ_sup, λ_inf) := extract randomly from L_i
9:    L_i := L_i \ (λ_sup, λ_inf)
10:   λ := (λ_sup + λ_inf)/2
11:   s := ChooseSeed(Sd, λ)
12:   s' := SLS_Σ(s, λ)
13:   Add s' to Archive
14:   Sd := Sd ∪ (s', λ)
15:   RemoveDominated(Sd)
16:   L_{i+1} := L_{i+1} ∪ (λ_sup, λ) ∪ (λ, λ_inf)
17:   if L_i = ∅ then i := i + 1
18: end while
19: RemoveDominated(Archive)
20: Output: Archive
```

$\mathsf{SLS}_1()$ and $\mathsf{SLS}_2()$. These new solutions and their corresponding weights, $\lambda = 1$ and $\lambda = 0$, respectively, are used to initialize L_0 and Sd. In the next phase, the `while` loop is iterated until a stopping criterion is met. At each iteration, a pair of consecutive weights $(\lambda_{\text{sup}}, \lambda_{\text{inf}})$ is subtracted randomly from L_i and used to calculate the new weight $\lambda = (\lambda_{\text{sup}} + \lambda_{\text{inf}})/2$. Then, procedure ChooseSeed uses this weight λ to choose a solution from the set of initial solutions Sd. To do so, first ChooseSeed finds the two non-dominated solutions that were obtained from scalarizations using the weights closest to λ:

$$\begin{aligned} s_{\text{inf}} &= \left\{ s_i \mid \max_{(s_i,\lambda_i)\in S_d} \{\lambda_i : \lambda_i < \lambda\} \right\} \\ s_{\text{sup}} &= \left\{ s_i \mid \min_{(s_i,\lambda_i)\in S_d} \{\lambda_i : \lambda_i > \lambda\} \right\} . \end{aligned} \tag{2}$$

Next, ChooseSeed calculates the scalar value of s_{sup} and s_{inf} according to the new weight λ following (1), and returns the solution with the smaller scalar value. This solution is the initial solution for SLS_Σ, the SLS algorithm used to tackle the scalarizations. This algorithm produces a new solution s', which is added to both the global *Archive* and (together with its corresponding weight) to the set of initial solutions Sd, from which any dominated solutions are removed. Finally, the set of weights for the next level L_{i+1} is extended with the new pairs $(\lambda_{\text{sup}}, \lambda)$ and $(\lambda, \lambda_{\text{inf}})$. This completes one iteration of the loop. If the current set of weights L_i is empty, a level of the search is complete, and the algorithm starts using pairs of weights from the next level L_{i+1}. In principle, this procedure may continue indefinitely, although larger number of scalarizations will lead to diminishing improvements in the approximation to the Pareto front.

3.2 Experimental analysis

In the original publication, we applied RA-TPLS to two bPFSPs [6]. In this paper, we perform a more comprehensive study by repeating the experiments on a larger set of instances for the bPFSPs and including results on the bTSP. The latter allows us to show that the shape of the Pareto front plays a fundamental role in the performance of the RA-TPLS strategy.

All algorithms evaluated in this paper were implemented in C++, compiled with gcc 4.4, and the experiments were run on a single core of Intel Xeon E5410 CPUs, running at 2.33 Ghz with 6MB of cache size under Cluster Rocks Linux version 4.2.1/CentOS 4.

3.2.1 Case study: bi-objective traveling salesman problem (bTSP)

Given a complete graph $G = (V, E)$ with $n = |V|$ nodes $\{v_1, \ldots, v_n\}$, a set of edges E, and a cost associated to each edge $c(v_i, v_j)$, the goal in the single-objective TSP is to find a Hamiltonian tour $p = (p_1, \ldots, p_n)$ that minimizes the total tour cost:

$$\text{minimize} \quad f(p) = c\left(v_{p_n}, v_{p_1}\right) + \sum_{i=1}^{n-1} c\left(v_{p_i}, v_{p_{i+1}}\right)$$

The single objective TSP may be directly extended to a multi-objective formulation by assigning a vector of costs to each edge, where each component corresponds to the cost of each objective. The goal then becomes to find the set of Hamiltonian tours that "minimizes" a vector of objective functions, where each objective is defined as above.

Here we focus on the bi-objective TSP (bTSP). We assume that the preferences of the decision maker are not known a priori. Hence, the goal is to find a set of feasible solutions that "minimizes" the bTSP in the sense of Pareto optimality. The bTSP is frequently used for testing algorithms and comparing their performance [9, 21]. Moreover, TPLS is a main component of the current state-of-the-art algorithm [17] for the bTSP.

Isometric and anisometric bTSP instances We created 10 Euclidean bTSP instances by generating for each instance two sets of 1,000 points with integer coordinates uniformly distributed in a square of side-length 10^5. We call these instances *isometric* because both distance matrices have similar range.

In addition, we generated other bTSP instances, where the first distance matrix (corresponding to the first objective) is Euclidean whereas the second matrix (corresponding to the second objective) is randomly generated with distance values in the range $[1, \max_{\text{dist}}]$, with $\max_{\text{dist}} \in \{5, 10, 25, 100\}$. Given the different range of both distance matrices, we call these instances *anisometric*. We generated 10 instances of 1,000 nodes for each value of $\max_{\text{dist}}$, that is, 40 *anisometric* bTSP instances in total.

Experimental setup for the bTSP The underlying single-objective algorithm for the TSP is an iterated local search (ILS) algorithm based on a first-improvement 3-opt algorithm [13].[1] In order to speed up the algorithm, we compute a new distance

[1] This algorithm is available online at http://www.sls-book.net/

matrix for each scalarization and we recompute the candidate sets used by the speed-up techniques of this ILS algorithm. For each scalarization, ILS runs for 1,000 ILS iterations (equal to the number of nodes in the instance). With our implementation and computing environment, 1,000 iterations require between 0.5 and 1 CPU seconds depending on the instances. Each of the two initial solutions is generated by running ILS for 2,000 iterations. Finally, each run of the multi-objective algorithms performs 30 scalarizations after generating the two initial solutions. The normalization of the objectives, necessary when solving a scalarization or calculating a weighted sum of the objectives, is performed by normalizing the two distance matrices to the same range.

We measure the quality of the results by means of the hypervolume unary measure [10, 25]. In the bi-objective space, the hypervolume measures the area of the objective space that is weakly dominated by the image of the solutions in a non-dominated set. This area is bounded by a reference point that is worse in all objectives than all points in all non-dominated sets measured. The larger is this area, the better is a non-dominated set. To compute the hypervolume, the objective values of all non-dominated solutions are normalized to the range $[1, 2]$, the values corresponding to the limits of the interval being the minimum and the maximum values ever found for each objective. We use $(2.1, 2.1)$ as the reference point for computing the hypervolume.

Experimental evaluation of RA-TPLS on bTSP instances We first study how the different TPLS strategies satisfy the *anytime* property by examining the quality of the Pareto front as the number of scalarizations increases. For each TPLS strategy, we plot the hypervolume value after each scalarization averaged across 15 independent runs. Figure 1 shows four exemplary plots comparing RA-TPLS, *1to2* and D-TPLS on two isometric bTSP instances, and two anisometric bTSP instances. We do not show the strategy *2to1* for isometric instances because it performs almost identical to *1to2* w.r.t. the hypervolume; however, for anisometric instances we include *2to1* due to its rather different behavior when compared to *1to2*. These plots are representative of the general results on other instances; the complete results are given as supplementary material [7].

For isometric bTSP instances, according to the top plots in Fig. 1, the three strategies reach similar final quality. However, there are strong differences in the development of the hypervolume during the execution of the algorithms. The hypervolume of the Pareto front approximations generated by RA-TPLS shows a quick initial increase, and for few scalarizations a much higher value than D-TPLS and *1to2*. Hence, if the algorithms are interrupted before completing the pre-defined number of scalarizations, RA-TPLS would clearly produce the best results.

For anisometric bTSP instances, where the first objective corresponds to a Euclidean distance matrix and the second objective corresponds to a randomly generated matrix with distance values in the range $[1, \max_{\text{dist}}]$, the bottom plots in Fig. 1 show a clear difference between strategies *1to2* and *2to1*. Moreover, the smaller the value of $\max_{\text{dist}}$, the larger is the difference. The value of $\max_{\text{dist}}$ also affects the anytime behavior and final performance of RA-TPLS. For small $\max_{\text{dist}}$, both *1to2* and D-TPLS seem to outperform RA-TPLS at various times. This can be explained as follows. Smaller values of $\max_{\text{dist}}$ result in a very large number of optimal solutions for the second objective. In such instances, the first scalarization of *2to1*, which

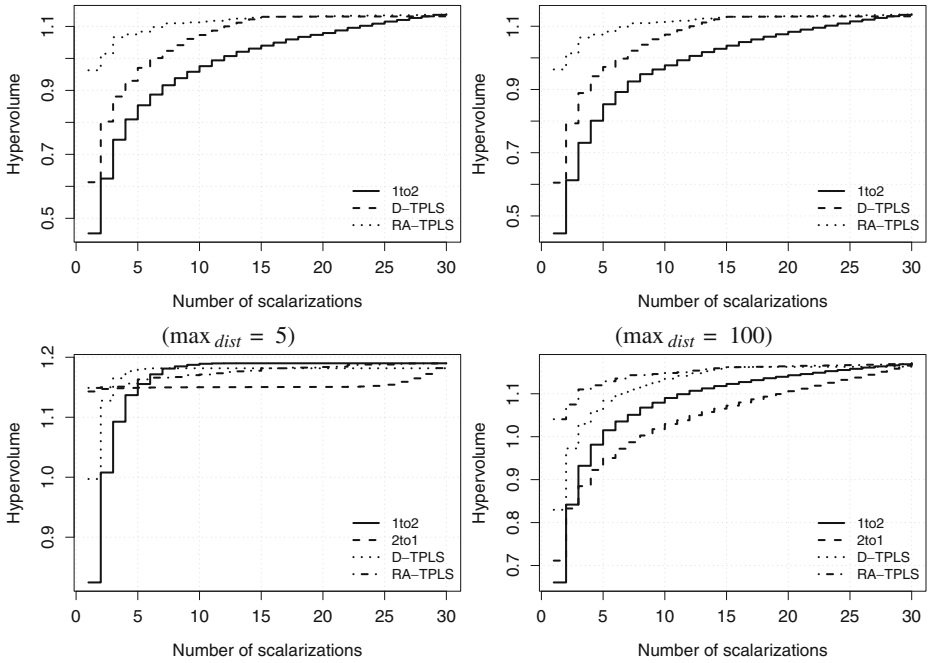

Fig. 1 Development of the hypervolume over the number of scalarizations for *1to2*, D-TPLS and RA-TPLS for two isometric (*top plots*) and two anisometric (*bottom plots*) bTSP instances. For anisometric instances we also plot the results of *2to1*, since they differ strongly from *1to2*. For isometric instances, there is almost no difference between *1to2* and *2to1*. Anisometric instances with intermediate values of $\max_{\text{dist}}$ (equal to 10 or 25) show a compromise trend between the two extreme values 5 and 100 (see supplementary material [7])

uses a nonzero weight for the first objective and a large weight for the second, generates a solution with a value of the second objective being still optimal and a good value of the first objective. This translates into a huge initial improvement of the hypervolume. The underlying reason is that the initial solution minimizing the second objective is weakly dominated by the solution returned for solving the first scalarization. Several subsequent scalarizations of *2to1* improve only slightly the value of the first objective, while keeping the optimal value of the second objective; therefore, the hypervolume improves very slowly. Only when the weight for the first objective grows large enough, a solution with a non-optimal value of the second objective is chosen, and the hypervolume starts improving in larger steps. On the other hand, when starting from the first objective in *1to2*, every scalarization finds non-dominated solutions closer to each other, and the hypervolume grows initially slower than what is observed for the first huge step in *2to1*. However, as soon as the weight of the second objective is large enough that only optimal values of the second objective solutions are accepted, the hypervolume quickly reaches its maximum. Finally, D-TPLS obtains better results because it progresses faster towards the second objective. All these behaviors show that equally distributing the computational effort in all directions does not pay off in these instances. It leads to a waste of scalarizations when being close to the optimum of the second objective, and

very slow progress when being close to the optimum of the first objective. This effect will be even stronger in the case of the bPFSP.

3.2.2 Case study: bi-objective permutation flow-shop scheduling problem

The flow-shop scheduling problem [14] models a very common type of environment in industry and it is therefore one of the most widely studied scheduling problems. In the flow-shop scheduling problem, 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 typical additional constraint is to forbid job passing, and, as a result, the processing sequence of the jobs is the same on all machines. Thus, any permutation of the jobs is a candidate solution. This formulation is called permutation flow-shop scheduling problem (PFSP).

In the PFSP, all processing times p_{ij} for a job J_i on a machine M_j are fixed, known in advance, and non-negative. Furthermore, we assume that all jobs are available at time 0. C_i denotes the completion time of a job J_i on the last machine M_m. The *makespan* ($C_{\max}$) is the completion time of the last job in the permutation. The PFSP with makespan minimization for more than two machines is $\mathcal{NP}$-hard in the strong sense [11].

The other objectives studied in this paper are the minimization of the *sum of completion times* and the minimization of the *total tardiness*. The sum of completion times is given by $\sum_{i=1}^{n} C_i$. The PFSP with sum of completion times minimization is strongly $\mathcal{NP}$-hard already for two machines [11]. Each job may have an additional associated due date d_i. The tardiness of a job J_i is defined as $T_i = \max\{C_i - d_i, 0\}$, and the total tardiness is given by $\sum_{i=1}^{n} T_i$. The PFSP with total tardiness minimization is strongly $\mathcal{NP}$-hard even for a single machine [4].

As a case study, in this paper we focus on two bPFSP variants:

1. *PFSP*-($C_{\max}$, $\sum C_i$) denotes the minimization of the makespan and the sum of completion times, and
2. *PFSP*-($C_{\max}$, $\sum T_i$) denotes the minimization of the makespan and the total tardiness.

Experimental setup for the bPFSP Recently, we developed a new, hybrid state-of-the-art SLS algorithm for these two bPFSPs [5, 8]. A crucial component in this hybrid algorithm is TPLS using effective iterated greedy (IG) algorithms [23] adapted to each objective and combinations thereof. We use here the same IG algorithms to implement the various TPLS variants. Concretely, each TPLS algorithm generates two initial solutions for each objective by running 1,000 iterations of the corresponding IG algorithm. Then, it performs 30 scalarizations, each scalarization running 500 iterations of the IG corresponding to the combination of objectives.

We generate 10 benchmark instances with $n = 50$ and $m = 20$ (`50 × 20`), and 10 instances with $n = 100$ and $m = 20$ (`100 × 20`), following the procedure described by Minella et al. [18]. These instances are available as supplementary material [7]. Given the large discrepancies in the range of the various objectives, all objectives are dynamically normalized using the maximum and minimum values found during each run for each objective. We compute and plot the evolution of the hypervolume as done earlier for the bTSP.

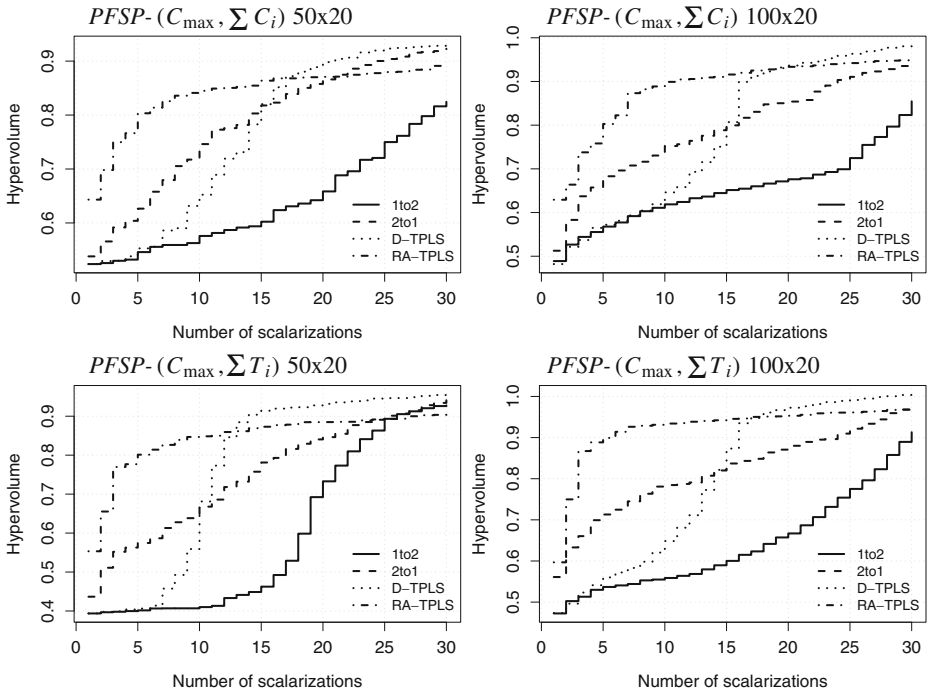

Fig. 2 Development of the average hypervolume over the number of scalarizations for *1to2*, *2to1*, D-TPLS, and RA-TPLS for bPFSP. Results are given for one instance of size `50 × 20` (*left column*) and one instance of size `100 × 20` (*right column*). The problems are *PFSP*-(C_{max}, $\sum C_i$) (*top plots*) and *PFSP*-(C_{max}, $\sum T_i$) (*bottom plots*)

Experimental evaluation of RA-TPLS on bPFSP instances We examine the quality of the result of each TPLS variant, *1to2*, *2to1*, D-TPLS, and RA-TPLS during the run of the algorithm. Figure 2 shows the development of the hypervolume of each TPLS variant, averaged across 15 independent runs. The plots show that the hypervolume value of *1to2*, *2to1*, and D-TPLS is rather poor up to the point that the sequence of weights reaches the other objective. On the other hand, RA-TPLS quickly reaches a high hypervolume in very few scalarizations. In terms of final quality, however, D-TPLS clearly performs better than RA-TPLS as soon as the former reaches half of its scalarizations and starts performing scalarizations back from the second to the first one. This fact and the differences between *1to2* and *2to1* strongly indicate that, for the bPFSPs considered here, the starting objective plays a significant role on both the anytime behavior and the final solution quality.

4 Adaptive TPLS

The TPLS variants discussed so far generate a sequence of weights that is determined by the number of scalarizations, and aims to allocate the same computational effort to all regions of the Pareto front. This strategy, however, may not be adequate when the underlying single-objective algorithm shows a different performance for each

objective, and the shape of the Pareto front is not regular in all search directions. Recently, we proposed an adaptive TPLS variant that dynamically generates weights in order to adapt the search to the shape of the Pareto front [6]. In this section, we explain this adaptive TPLS variant and discuss possible improvements that have not been considered before.

4.1 Adaptive anytime strategy

Our *adaptive* TPLS [6] is inspired by the *dichotomic* scheme proposed by Aneja and Nair [1] for exact algorithms and recently used for the approximate case by Lust and Teghem [17]. The *dichotomic* scheme does not define the weights in advance but determines them in dependence of the solutions already found. More formally, given a pair of solutions (s_1, s_2), the new weight λ is perpendicular to the segment (henceforth denoted by an overline) defined by s_1 and s_2 in the objective space, that is, assuming the range of the objectives are normalized or equal, we have

$$\lambda = \frac{f_2(s_1) - f_2(s_2)}{f_2(s_1) - f_2(s_2) + f_1(s_2) - f_1(s_1)}. \tag{3}$$

The *dichotomic* scheme used in these two earlier papers has a natural stopping criterion, and it progresses recursively depth-first. As a result, if stopped early, it would assign an uneven computational effort along the front, leading to a poor distribution of solutions and, hence, to poor anytime behavior. Moreover, Lust and Teghem [17] apply the *dichotomic* scheme as a *Restart* strategy that starts each scalarization from a newly generated initial solution. In the exact case, the algorithm of Aneja and Nair [1] is deterministic, and, hence, applying the same weight results in the same output. Also, the concept of seeding a scalarization is not considered. Our extension of the *dichotomic* strategy to the TPLS framework makes effective use of solutions found by previous scalarizations to seed later scalarizations and satisfies the *anytime* property [6]. We describe this *adaptive* TPLS strategy as Algorithm 3.

The main data structure is a set S of pairs of solutions found in previous scalarizations. This set is initialized with the solutions found by optimizing each single objective using $\mathsf{SLS}_1()$ and $\mathsf{SLS}_2()$. At each iteration, the algorithm selects the pair of solutions $(s_{\text{sup}}, s_{\text{inf}}) \in S$ whose images define the "largest gap" in the objective space, using a given norm $\|(\mathbf{f}(s), \mathbf{f}(s'))\|$ to compare every pair of solutions. The idea is to focus the search on the largest gap in the Pareto front in order to obtain a well-spread set of non-dominated solutions. This is different from the original *dichotomic* scheme, which explores segments recursively. We originally proposed to use as norm the Euclidean distance in the normalized objective space [6], and we use this norm in the following. However, we propose a new alternative in Section 6. After choosing the pair of solutions $(s_{\text{sup}}, s_{\text{inf}})$ according to the norm, the algorithm calculates a new weight λ perpendicular to the segment $\overline{\mathbf{f}(s_{\text{sup}})\mathbf{f}(s_{\text{inf}})}$ in the objective space, following (3). Next, the underlying single-objective SLS algorithm, SLS_Σ, is run using the weight λ either once, starting from either s_{sup} and s_{inf}, or twice, starting one time from solution s_{sup} and one time from solution s_{inf}. Which of these two possibilities is chosen, depends on the parameter `one_seed_case`.

In the last step of an iteration, procedure Update updates the set of initial solutions S using the new solutions found. If s' is a new solution, any single solution in S dominated by s' is replaced by s', and any pair of solutions (weakly) dominated by

Algorithm 3 Adaptive "anytime" TPLS strategy

```
1: s_1 := SLS_1()
2: s_2 := SLS_2()
3: Add s_1, s_2 to Archive
4: S := {(s_1, s_2)}
5: while not stopping criteria met do
6:    (s_sup, s_inf) := arg max_{(s,s')∈S} ‖(f(s), f(s'))‖
7:    Calculate λ perpendicular to f(s_sup)f(s_inf) following (3)
8:    if one_seed_case then
9:       s := ChooseRandomly(s_sup, s_inf)
10:      s' := SLS_Σ(s, λ)
11:      Add s' to Archive
12:      Update(S, s')
13:   else
14:      s'_sup := SLS_Σ(s_sup, λ)
15:      s'_inf := SLS_Σ(s_inf, λ)
16:      Add s'_sup and s'_inf to Archive
17:      Update(S, s'_sup)
18:      Update(S, s'_inf)
19:   end if
20: end while
21: RemoveDominated(Archive)
22: Output: Archive
```

s' is removed. The *dichotomic* scheme [1, 17] only accepts solutions for inclusion in S if they lie *within* the triangle defined in the objective space by the solutions s_{sup} and s_{inf}, and their local ideal point, which is the point $(f_1(s_{\text{sup}}), f_2(s_{\text{inf}}))$ (see Fig. 3). Heuristic algorithms may, however, generate solutions that are in the gray area *outside* the triangle. Solutions outside the gray area are either dominated or not supported (that is, non-dominated but not optimal for any scalarization); therefore, our *adaptive* strategy accepts *all* solutions in the gray area for inclusion in S. If a solution s' is accepted for inclusion in S, then the pair $(s_1, s_2) \in S$ with

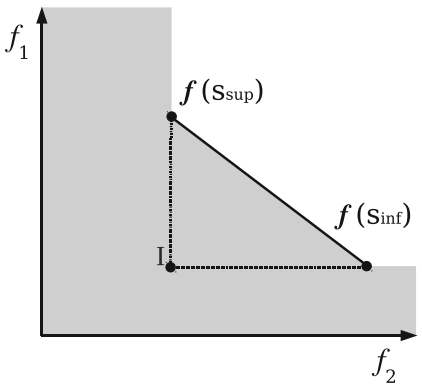

Fig. 3 Only solutions in the gray area are accepted as initial solutions for further scalarizations (see the text for details)

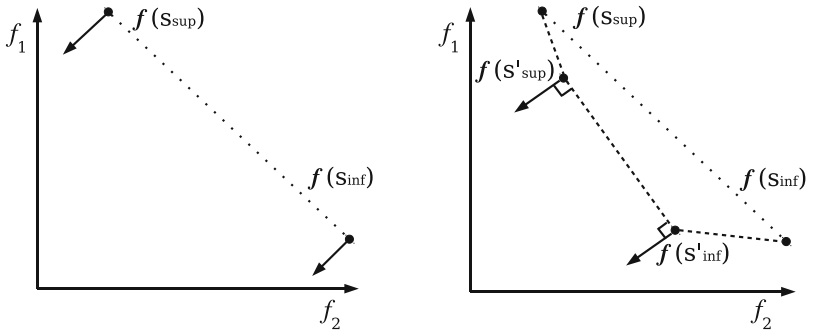

Fig. 4 A single iteration of the Adaptive "Anytime" TPLS strategy with two initial solutions (AN-TPLS-2seed). On the *left* the state before the iteration and on the *right* after S has been updated. The next segment that will be considered is (s'_{sup}, s'_{inf}) because of its larger distance

$f_1(s_1) < f_1(s') < f_1(s_2)$ is removed, and two new pairs (s_1, s') and (s', s_2) are added to S. Since each iteration produces at most two new solutions (s'_{sup} and s'_{inf}, or simply s'_1 in the one-seed variant), a maximum of three new pairs are added to S every iteration. Figure 4 shows an example of the update of S after one iteration of the *adaptive* algorithm. We call the algorithm that uses two initial solutions AN-TPLS-

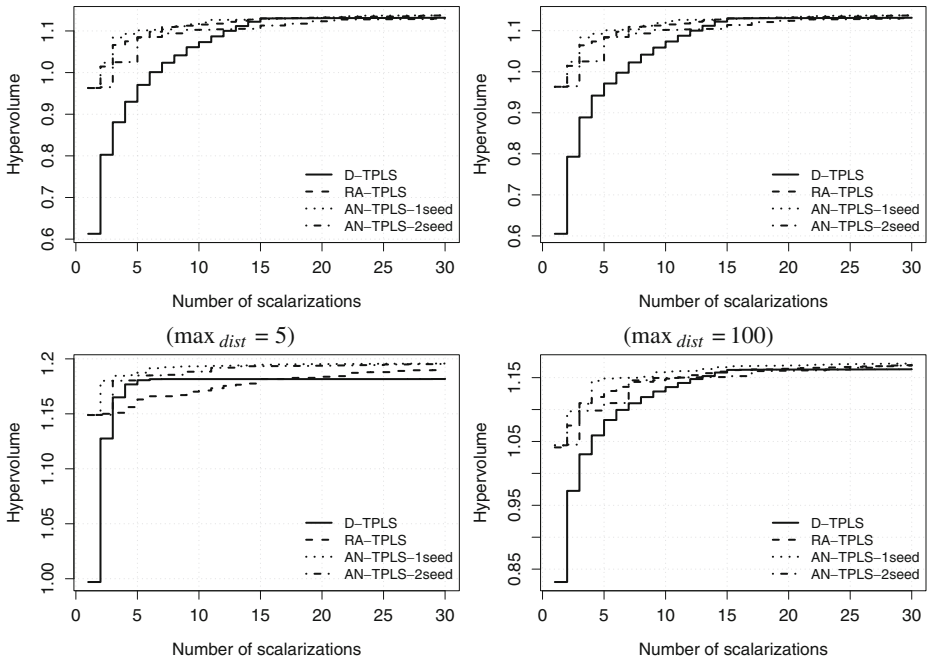

Fig. 5 Development of the hypervolume over the number of scalarizations for D-TPLS, RA-TPLS, AN-TPLS-2seed and AN-TPLS-1seed for two isometric TSP instances and two anisometric TSP instances. Anisometric instances with intermediate value of $\max_{dist}$ show a compromise trend between the two extremes shown here (see supplementary material [7])

2seed in what follows (for adaptive normal TPLS), and we call AN-TPLS-1seed its variant using only one initial solution (in the outline of Algorithm 3, this corresponds to `one_seed_case=true`).

4.2 Experimental evaluation of adaptive TPLS on bTSP instances

To evaluate the performance of AN-TPLS-2seed and its variant AN-TPLS-1seed, we use the same experimental setup described in Section 3. We present the average hypervolume evolution of AN-TPLS-2seed and AN-TPLS-1seed in Fig. 5, comparing it to D-TPLS and RA-TPLS.

For the two isometric instances, AN-TPLS-1seed appears to be better than AN-TPLS-2seed. By checking carefully the output, we noticed that the underlying ILS algorithm usually finds two solutions whose images are very close to each other in the objective space or possibly even the same. Hence, using two initial solutions gives a negligible improvement with respect to the hypervolume in comparison to using a single one.

For the two anisometric instances, AN-TPLS-1seed is again the best strategy. Interestingly, one can see that the higher the value of $\max_{\text{dist}}$, the closer is the performance of RA-TPLS to AN-TPLS-1seed. This is due to the fact that by

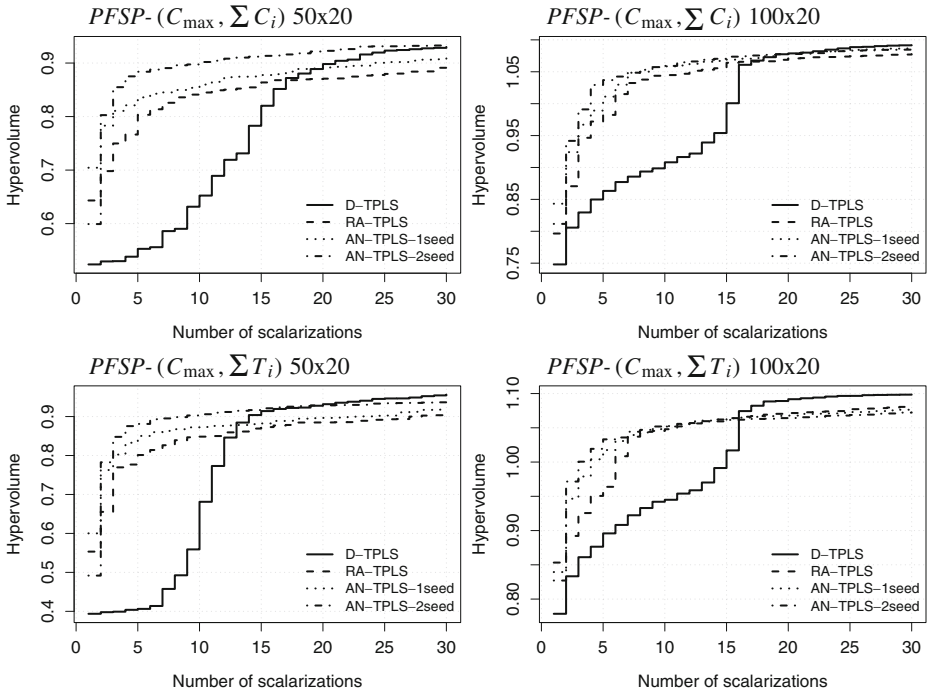

Fig. 6 Development of the hypervolume over the number of scalarizations for D-TPLS, RA-TPLS, AN-TPLS-2seed and AN-TPLS-1seed for bPFSP. Results are given for one instance of size `50 × 20` (*left column*) and one instance of size `100 × 20` (*right column*). The problems are *PFSP*-$(C_{\max}, \sum C_i)$ (*top plots*) and *PFSP*-$(C_{\max}, \sum T_i)$ (*bottom plots*)

increasing $\max_{dist}$, instances are "smoother" in the sense they resemble more the isometric ones and, therefore, there is less need to adapt the weights to the particular shape of the Pareto front.

4.3 Experimental evaluation of adaptive TPLS on bPFSP instances

To test on a problem that has a front resulting from objectives with different properties, we use again the bPFSPs that were already described in Section 3.2.2. Results are presented in Fig. 6. In contrast with the results on the bTSP, AN-TPLS-2seed clearly outperforms RA-TPLS and it is also significantly better than AN-TPLS-1seed. Still, D-TPLS shows on several instances better final performance than AN-TPLS-2seed according to the hypervolume indicator.

4.4 Further analysis of AN-TPLS-1seed and AN-TPLS-2seed

The results of the evaluation of Adaptive TPLS show that AN-TPLS-1seed is the best strategy for bTSPs while AN-TPLS-2seed is the best strategy for bPFSPs. In this section, we illustrate the fact that the different behavior of the underlying single-objective algorithms for each problem leads to such results. To do so, we examine in detail the scalarizations performed by AN-TPLS-2seed for the first three weights and each of the two seeds. We perform 15 independent runs of each scalarization, and we plot the objective vectors of the solutions obtained. Figure 7 presents the results for an isometric bTSP instance. The left plot shows the objective vectors of the solutions obtained with the first weight, using each of the two initial solutions as seed. The line indicates the direction of the search (the weight vector). The objective vector obtained by each of the 15 independent runs is represented by the same symbol as the seed used to initialize the corresponding run. In this case, all points overlap, that is, the resulting objective vectors fall into a tiny area of the objective space, independently of which initial solution was used as seed. The middle and right plots of Fig. 7 show the results of the same experiment for the second and the third weight, respectively. Hence, there is no significant advantage in using two different

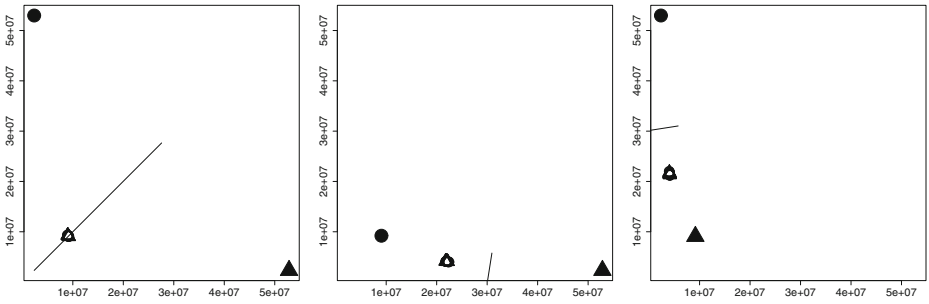

Fig. 7 Distribution of objective vectors in the objective space after the first (*left plot*), the second (*middle plot*) and third (*right plot*) weights, for an isometric bTSP instance. The two vectors plotted with filled, black symbols (*circle or triangle*) are the initial ones, used to define the weight (the direction is represented with a *line*) and used as seeds. The symbol of each point denotes the seed from which they have been obtained

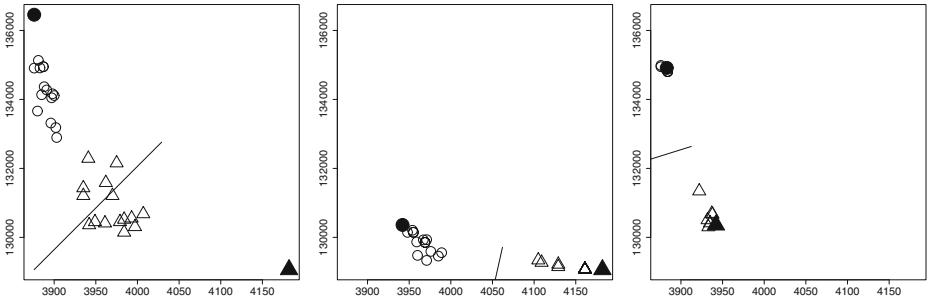

Fig. 8 Distribution of objective vectors in the objective space after the first (*left plot*), the second (*middle plot*) and third (*right plot*) weights, for a *PFSP*-($C_{\max}$, $\sum C_i$) instance. The two vectors plotted with filled, black symbols (*circle or triangle*) are the initial ones, used to define the weight (the direction is represented with a *line*) and used as seeds. The symbol of each point denotes the seed from which they have been obtained

initial solutions with the same weight with respect to increasing the hypervolume. We observed the same behavior on the anisometric bTSP instances.

We repeat the experiment for the bPFSP (using IG as the underlying single-objective algorithm) and present the results in Fig. 8. The situation is very different now. As the plots show, there is a larger variability on the location of the resulting objective vectors, even for those solutions obtained from the same seed. More importantly, the vectors resulting from the two different seeds are located in two clearly distinct clusters, despite all being solutions for the same scalarized problem. Hence, performing a scalarization from two different seeds is advantageous in the case of the bPFSP in order to obtain a better distribution of solutions along the Pareto front and, hence, a higher value of the hypervolume indicator.

This insight might be used to define when it is better to use one or two initial solutions. A simple way to choose automatically AN-TPLS-1seed or AN-TPLS-2seed would be to use AN-TPLS-2seed for the first weight, and consider the Euclidean distance between the objective vectors of the two solutions obtained. If the distance is large enough, it might be advantageous to continue with AN-TPLS-2seed; while if points are very close, it might be better to switch to AN-TPLS-1seed. Another possibility to choose the best strategy would be to launch both strategies, allowing two scalarizations for each. The search would then continue with the strategy that leads to the larger hypervolume. Evaluating these ideas is beyond the scope of this paper, and, hence, we let this question open for future investigation.

4.5 Adaptive focus TPLS

In AN-TPLS-2seed, when two adjacent segments in S are almost parallel, two scalarizations are solved using the same initial solution (the solution shared by the two segments) and very similar weights (because the two vectors perpendicular to the segments will again be almost parallel). A careful analysis of AN-TPLS-2seed showed that such situation actually occurs, and may result in negligible progress of the search. In order to avoid this problem, one can focus the search direction of each scalarization towards the center of each segment and further improve the results of AN-TPLS-2seed. We call this variant *adaptive focus* TPLS (AF-TPLS).

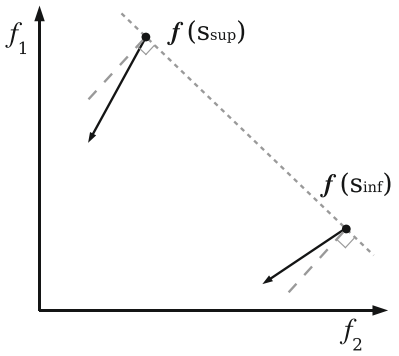

Fig. 9 AF-TPLS strategy: the two weights are "focused" to the center of the segment

Given a segment $\overline{\mathbf{f}(s_1)\mathbf{f}(s_2)}$, with $f_1(s_1) < f_1(s_2)$, AF-TPLS generates two weights λ_1 and λ_2 as

$$\lambda_1 = \lambda - \theta \cdot \lambda \quad \text{and} \quad \lambda_2 = \lambda + \theta(1 - \lambda), \tag{4}$$

where λ is the weight perpendicular to the segment computed by (3), and θ is a parameter that modifies λ towards the center of the segment (see Fig. 9).

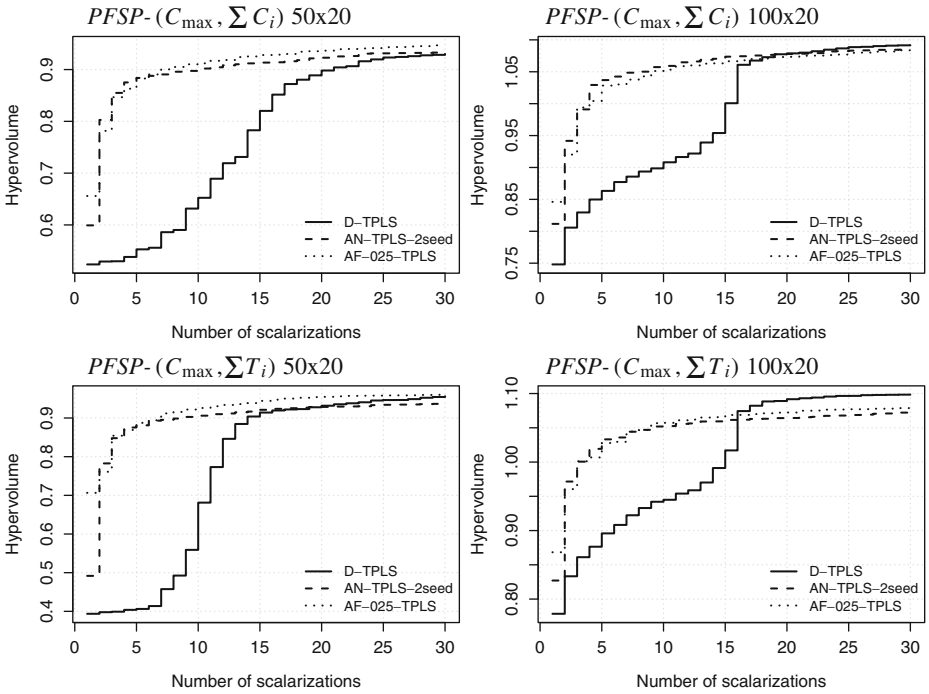

Fig. 10 Development of the hypervolume over the number of scalarizations for D-TPLS, AN-TPLS-2seed and AF-TPLS. Results are given for one instance of size `50 × 20` (*left column*) and one instance of size `100 × 220` (*right column*). The problems are *PFSP*-(C_{max}, $\sum C_i$) (*top plots*) and *PFSP*-(C_{max}, $\sum T_i$) (*bottom plots*)

These two new weights replace the weight λ in Algorithm 3, that is, the run of the SLS algorithm that uses s_1 as initial solution solves a scalarization according to weight λ_1, while the run starting from s_2 uses the weight λ_2. A value of $\theta = 0$ would reproduce the AN-TPLS-2seed strategy.

4.6 Experimental evaluation of adaptive focus on bPFSP instances

We test AF-TPLS on the bPFSP using different values of $\theta = \{0.05, 0.15, 0.25, 0.5\}$. Plots that present a comparison of AF-TPLS using these values are provided as supplementary material [7]. The values 0.25 and 0.15 show a similar performance, indicating a relative robustness of AF-TPLS with respect to the setting of θ. For the following comparisons, we consider only AF-TPLS using 0.25. In Fig. 10, we present a comparison of AF-TPLS, AN-TPLS-2seed and D-TPLS. AF-TPLS is at least as good as AN-TPLS-2seed, and it is often able to outperform it. Furthermore, AF-TPLS reaches a final quality often equal to or better than the one reached by D-TPLS. The instance `100 × 20_1` of problem *PFSP*-($C_{\max}$, $\sum T_i$) (bottom-right in Fig. 10) is actually the only instance of the 20 we tested where D-TPLS performs clearly better than AF-TPLS.

5 Statistical analysis

We have examined so far the results of the different approaches by comparing their performance in each instance. In order to assess the performance over the whole set of instances, we perform the following statistical analysis on each problem. The analysis is based on the Friedman test for analyzing non-parametric unreplicated complete block designs, and its associated post-test for multiple comparisons [3]. First, we calculate the mean hypervolume of the 15 runs of each algorithm for each instance. Then, we perform the Friedman test using the ten instances as the blocking factor, and the different strategies as the treatment factor. In most cases, the Friedman test rejects the null hypothesis with a p-value lower than 0.05. Then, we rank the strategies per instance according to the mean hypervolume, the lower rank the better. From this ranking we calculate the difference (ΔR) between the sum of ranks of each strategy and the best ranked one (with the lowest sum of ranks). Finally, we calculate the minimum difference between the sum of ranks of two strategies that is statistically significant (ΔR_α), given a significance level of $\alpha = 0.05$. We indicate in bold face the best strategy (the one having the lowest sum of ranks) and those that are not significantly different from the best one.

5.1 Results on the bTSP

We perform the statistical tests after the algorithms have performed 10, 20 and 30 scalarizations. We compare the strategies *1to2*, D-TPLS, RA-TPLS, AN-TPLS-2seed and AN-TPLS-1seed. We do not consider AF-TPLS since it leads to poor performance for bTSP instances (see the supplementary material [7]).

Results are given in Table 1 for isometric instances and in Table 2 for anisometric ones. When the value of the critical difference (ΔR_α) is equal to 0, the strategies have the same ranking over all instances. The numbers in parenthesis are the

Table 1 Statistical analysis for the isometric bTSP

N_{scalar}	ΔR_α	Strategies (ΔR)
10	0	**AN-TPLS-1seed**, RA-TPLS (10), AN-TPLS-2seed (20), D-TPLS (30), *1to2*(40)
20	0	**AN-TPLS-1seed**, RA-TPLS (10), D-TPLS (20), AN-TPLS-2seed (30), *1to2*(40)
30	0	***1to2***, RA-TPLS (10), AN-TPLS-1seed (20), D-TPLS (30), AN-TPLS-2seed (40)

For each number of scalarizations, strategies are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best strategy. The strategy that is significantly better than the other ones is indicated in bold face. For isometric bTSP the ordering of all strategies is the same on all instances and, hence, $\Delta R_\alpha = 0$

difference of ranks relative to the best strategy. For isometric instances, AN-TPLS-1seed is the best strategy before completion. However, when algorithms run until completion, *1to2* is significantly better than the other ones. For anisometric instances, we performed independent tests for each value of $\max_{\text{dist}}$ and we found that the results are consistent across the different values of $\max_{\text{dist}}$. AN-TPLS-1seed is always significantly better than all the other strategies. Hence, it is the strategy that should be used for anisometric instances, no matter the value of $\max_{\text{dist}}$.

5.2 Results on the bPFSPs

For the bPFSP, we perform the same procedure separately for each combination of objectives, each instance size `50 × 20` and `100 × 20`, and we measure the hypervolume after 10, 20 and 30 scalarizations. We compared D-TPLS, RA-TPLS, AN-TPLS-2seed, AN-TPLS-1seed and AF-TPLS (with $\theta = 0.25$). The results are given in Table 3.

Table 2 Statistical analysis for the anisometric bTSP

N_{scalar}	ΔR_α	Strategies (ΔR)
$\max_{\text{dist}} = 5$		
10	3.31	**AN-TPLS-1seed**, AN-TPLS-2seed (14), *1to2*(16), D-TPLS (30), RA-TPLS (40)
20	2.03	**AN-TPLS-1seed**, AN-TPLS-2seed (10), *1to2*(20), RA-TPLS (31), D-TPLS (39),
30	3.31	**AN-TPLS-1seed**, AN-TPLS-2seed (10), *1to2*(24), RA-TPLS (26), D-TPLS (40)
$\max_{\text{dist}} = 10$		
10	0	**AN-TPLS-1seed**, D-TPLS (10), AN-TPLS-2seed (20), RA-TPLS (30), *1to2*(40)
20	2.03	**AN-TPLS-1seed**, AN-TPLS-2seed (11), *1to2*(19), RA-TPLS (30), D-TPLS (40)
30	0	**AN-TPLS-1seed**, AN-TPLS-2seed (10), *1to2*(20), RA-TPLS (30), D-TPLS (40)
$\max_{\text{dist}} = 25$		
10	0	**AN-TPLS-1seed**, AN-TPLS-2seed (10), RA-TPLS (20), D-TPLS (30), *1to2*(40)
20	3.31	**AN-TPLS-1seed**, AN-TPLS-2seed (10), RA-TPLS (24), D-TPLS (26), *1to2*(40)
30	4.11	**AN-TPLS-1seed**, AN-TPLS-2seed (14), *1to2*(17), RA-TPLS (29), D-TPLS (40)
$\max_{\text{dist}} = 100$		
10	0	**AN-TPLS-1seed**, AN-TPLS-2seed (10), RA-TPLS (30), D-TPLS (20), *1to2*(40)
20	3.31	**AN-TPLS-1seed**, AN-TPLS-2seed (10), RA-TPLS (24), D-TPLS (26), *1to2*(40)
30	4.11	**AN-TPLS-1seed**, AN-TPLS-2seed (14), *1to2*(17), RA-TPLS (29), D-TPLS (40)

For each number of scalarizations, strategies are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best strategy. The strategy that is significantly better than the other ones is indicated in bold face

Table 3 Statistical analysis for the bPFSP

N_{scalar}	ΔR_α	Strategies (ΔR)
$(C_{\max}, \sum C_i)$ `50 × 20`		
10	5.41	**AF-TPLS**, AN-TPLS-2seed (6), AN-TPLS-1seed (12), RA-TPLS (26), D-TPLS (36)
20	7.65	**AN-TPLS-2seed, AF-TPLS (2)**, AN-TPLS-1seed (20), D-TPLS (21), RA-TPLS (32)
30	9.63	**AN-TPLS-2seed, AF-TPLS (3), D-TPLS (4)**, AN-TPLS-1seed (13), RA-TPLS (30)
$(C_{\max}, \sum C_i)$ `100 × 20`		
10	6.51	**AF-TPLS, AN-TPLS-2seed (4), AN-TPLS-1seed (5)**, RA-TPLS (23), D-TPLS (33)
20	9.91	**AF-TPLS, AN-TPLS-2seed (2)**, AN-TPLS-1seed (11), D-TPLS (18), RA-TPLS (29)
30	9.44	**D-TPLS, AF-TPLS (8)**, AN-TPLS-2seed (16), AN-TPLS-1seed (27), RA-TPLS (29)
$(C_{\max}, \sum T_i)$ `50 × 20`		
10	3.88	**AF-TPLS**, AN-TPLS-2seed (5), AN-TPLS-1seed (16), RA-TPLS (27), D-TPLS (37)
20	5.36	**AF-TPLS**, AN-TPLS-2seed (13), D-TPLS (23), AN-TPLS-1seed (24), RA-TPLS (40)
30	5.76	**AF-TPLS, D-TPLS (1)**, AN-TPLS-2seed (11), AN-TPLS-1seed (25), RA-TPLS (33)
$(C_{\max}, \sum T_i)$ `100 × 20`		
10	4.97	**AF-TPLS**, AN-TPLS-2seed (14), AN-TPLS-1seed (14), RA-TPLS (28), D-TPLS (39)
20	10.36	**AF-TPLS**, AN-TPLS-2seed (13), D-TPLS (19), AN-TPLS-1seed (22), RA-TPLS (31)
30	8.42	**D-TPLS, AF-TPLS (2)**, AN-TPLS-2seed (21), RA-TPLS (23), AN-TPLS-1seed (29)

For each number of scalarizations, strategies are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best strategy. Strategies that are not significantly different to the best one are indicated in bold face. See the text for details

For a low number of scalarizations, the *adaptive* strategies (AN-TPLS-2seed and AF-TPLS) are always superior to the classical TPLS strategies. Moreover, AF-TPLS is never significantly worse than D-TPLS, when the latter runs until completion (30 scalarizations), while the opposite is true two times. In conclusion, AF-TPLS would be the strategy of choice for bPFSP problems.

6 Optimistic hypervolume improvement as selection criterion

The main idea of the adaptive anytime strategies is to focus the search on the most promising region of the objective space for improving the quality of the Pareto front approximation. In this sense, the algorithm aims at filling the "largest gaps" in the Pareto front approximation. In order to measure the "size of the gap", we use a norm as described in line 6 of Algorithm 3 (Section 4), where the pair of solutions that maximizes it are selected as seeds for the next scalarization.

For all experiments presented so far, we have used as norm the Euclidean distance on the normalized objective space. Although this distance leads to a good "visual" distribution of solutions, it may not lead to the selection of the seeds with the maximum potential of improving quality. A measure of the quality of the current Pareto front approximation is the hypervolume, and therefore, we could select the pair of seeds that may lead to the largest improvement of the hypervolume. Assuming that the new solution found is within the rectangle defined in the objective space by the two seeds, the maximum improvement in terms of hypervolume is proportional to the area of this rectangle. Hence, using normalized objective values, we compute this norm as follows:

$$\|(s, s')\|_{\mathcal{HV}} = |((f_1(s) - f_1(s')) \cdot ((f_2(s) - f_2(s'))| \tag{5}$$

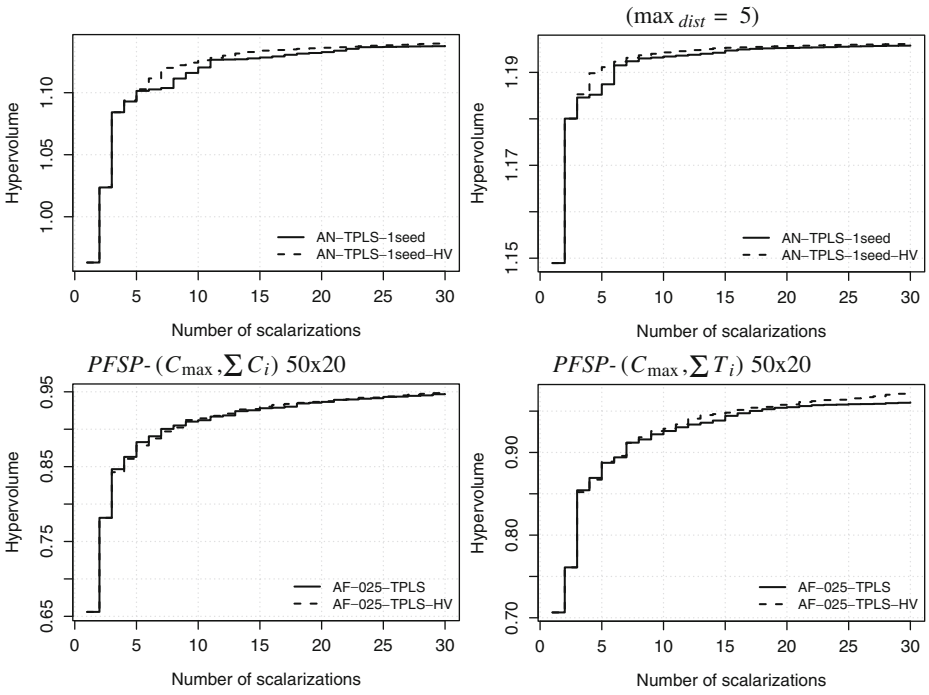

Fig. 11 Development of the hypervolume over the number of scalarizations for AN-TPLS-1seed using Euclidean distance and $\|(s, s')\|_{\mathcal{HV}}$ for one isometric TSP instance (*top-left*), one anisometric TSP instance (*top-right*), and one instance of bPFSP with the two different combinations of objectives

This optimistic hypervolume improvement is different from measuring the contribution to the hypervolume of each solution for the purposes of selecting or discarding some solutions [2, 15].

We compare this optimistic hypervolume improvement with the Euclidean distance as the selection criterion in AN-TPLS-1seed, which is the best adaptive TPLS strategy for the isometric and the anisometric bTSP, and AF-TPLS, which is the best adaptive TPLS strategy for the bPFSP.

Figure 11 shows the development of the hypervolume of the resulting adaptive TPLS variants. The version of AN-TPLS-1seed that uses the $\|(s, s')\|_{\mathcal{HV}}$ norm is

Table 4 Statistical analysis for the isometric bTSP

N_{scalar}	ΔR_α	Strategies (ΔR)
10	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), *1to2*(20)
20	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), *1to2*(20)
30	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, *1to2*(10), AN-TPLS-1seed (20)

For each number of scalarizations, strategies are ordered according to the rank obtained

Table 5 Statistical analysis for the anisometric bTSP

N_{scalar}	ΔR_α	Strategies (ΔR)
$\max_{\text{dist}} = 5$		
10	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
20	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
30	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
$\max_{\text{dist}} = 10$		
10	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
20	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
30	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
$\max_{\text{dist}} = 25$		
10	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
20	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
30	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
$\max_{\text{dist}} = 100$		
10	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
20	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)
30	0	**AN-TPLS-1seed**$_{\mathcal{HV}}$, AN-TPLS-1seed (10), D-TPLS (20)

For each number of scalarizations, strategies are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best strategy. The strategy that is significantly better than the other ones is indicated in bold face

slightly but consistently better than the one that uses the Euclidean distance. For AF-TPLS on the bPFSP, results are not as consistent as for the bTSP (additional plots are available as supplementary material [7]).

Table 6 Statistical analysis for the bPFSP

N_{scalar}	ΔR_α	Strategies (ΔR)
$(C_{\max}, \sum C_i)$ `50 × 20`		
10	3.87	**AF-TPLS, AF-TPLS**$_{\mathcal{HV}}$ **(3)**, D-TPLS (16.5)
20	6.80	**AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS (3)**, D-TPLS (13.5)
30	8.23	**AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS (4)**, D-TPLS (11)
$(C_{\max}, \sum C_i)$ `100 × 20`		
10	3.96	**AF-TPLS**$_{\mathcal{HV}}$, AF-TPLS (4), D-TPLS (17)
20	7.07	**AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS (2)**, D-TPLS (13)
30	pval>0.05	**D-TPLS, AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS**
$(C_{\max}, \sum T_i)$ `50 × 20`		
10	4.54	**AF-TPLS**$_{\mathcal{HV}}$, AF-TPLS (4), D-TPLS (17)
20	4.54	**AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS (4)**, D-TPLS (17)
30	pval>0.05	**AF-TPLS**$_{\mathcal{HV}}$**, AF-TPLS, D-TPLS**
$(C_{\max}, \sum T_i)$ `100 × 20`		
10	3.96	**AF-TPLS, AF-TPLS**$_{\mathcal{HV}}$ **(6)**, D-TPLS (18)
20	6.86	**AF-TPLS, AF-TPLS**$_{\mathcal{HV}}$ **(10)**, D-TPLS (14)
30	pval>0.05	**AF-TPLS, D-TPLS, AF-TPLS**$_{\mathcal{HV}}$

For each number of scalarizations, strategies are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best strategy. Strategies that are not significantly different to the best one are indicated in bold face. See the text for details

To assess the statistical significance of the differences between the two selection criteria over all instances, we perform the same statistical analysis as in the previous section. For the isometric bTSP, we compare in Table 4 the quality of AN-TPLS-1seed, its variant that uses the $\|(s, s')\|_{\mathcal{HV}}$ norm (AN-TPLS-1seed$_{\mathcal{HV}}$), and *1to2*, which outperformed all other strategies in terms of final quality. We compare in Table 5 AN-TPLS-1seed, AN-TPLS-1seed$_{\mathcal{HV}}$, and D-TPLS, for the anisometric bTSP. The results are consistent for the two types of instances, and all values of $\max_{\text{dist}}$. AN-TPLS-1seed$_{\mathcal{HV}}$ is the best-ranked strategy and it is always significantly better than all the other ones, including *1to2*. In the case of the bPFSP, Table 6 compares the two adaptive strategies AN-TPLS-2seed and AF-TPLS, their variants using the $\|(s, s')\|_{\mathcal{HV}}$ norm, and D-TPLS. The improvement is not as consistent as for

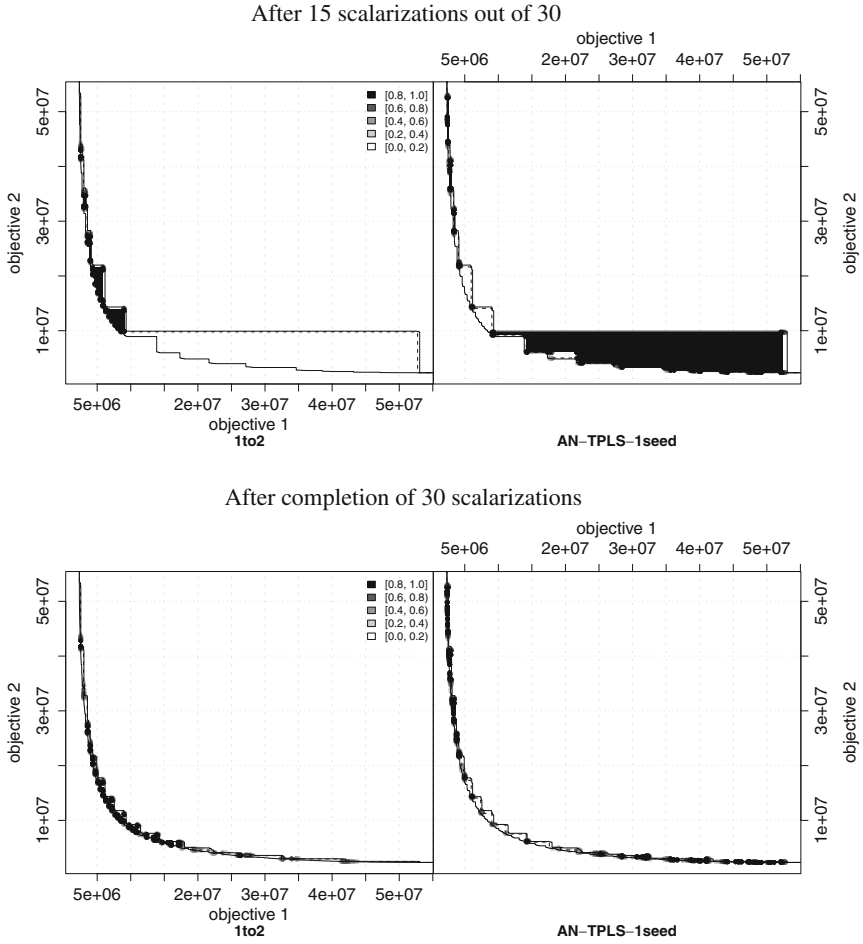

Fig. 12 EAF differences for one isometric TSP instance, after 15 scalarizations (*top*) and after 30 scalarizations (*bottom*). Strategies are *1to2* (*left*) and AN-TPLS-1seed (*right*). Plots for other instances are available as supplementary material [7]

the bTSP. However, AF-TPLS$_{\mathcal{HV}}$ is most often the best-ranked strategy and never significantly worse than the best ranked one.

7 Graphical analysis based on the EAF differences

We further explore the differences between RA-TPLS, AF-TPLS and D-TPLS by examining the empirical attainment functions (EAF) of the final results after 30 scalarizations. The EAF of an algorithm provides the probability, estimated from several runs, of an arbitrary point in the objective space being attained by (dominated by or equal to) a solution obtained by a single run of the algorithm [12]. Examining the differences between the EAFs of two algorithms allows us to identify regions of

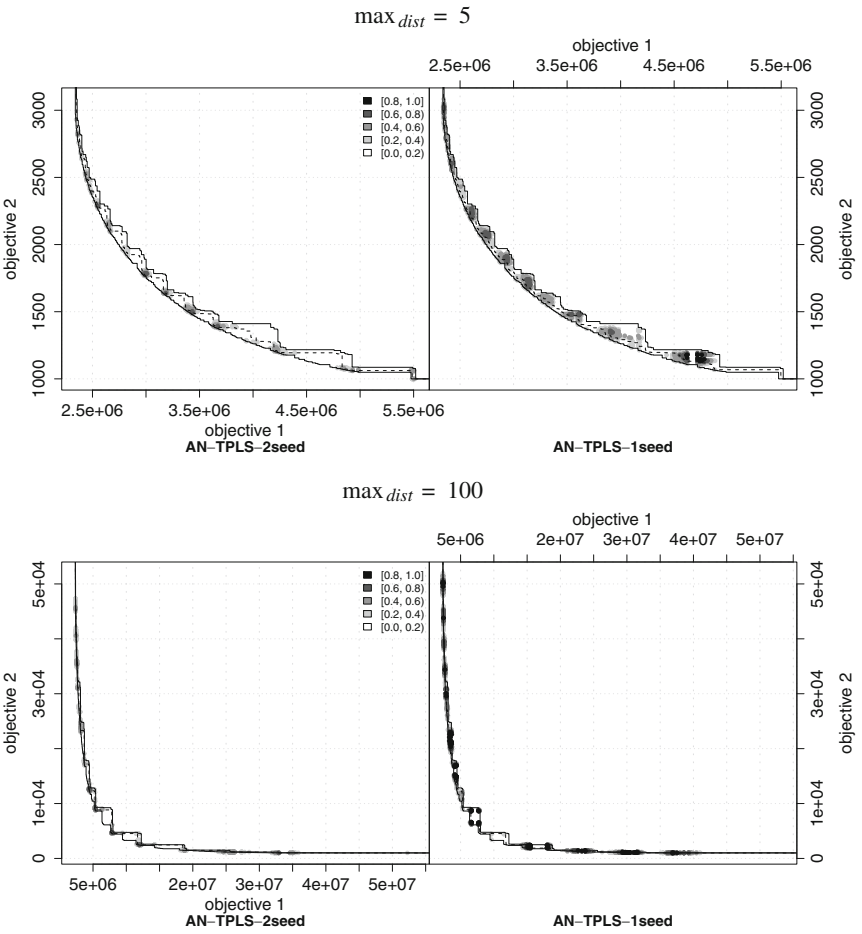

Fig. 13 EAF differences for two anisometric TSP instances ($\max_{\text{dist}} = 5$, *top*, and $\max_{\text{dist}} = 100$, *bottom*), after completing 30 scalarizations. Strategies are AN-TPLS-2seed (*left*) and AN-TPLS-1seed (*right*). Other values of $\max_{\text{dist}}$ show similar trends. Plots for other instances are available as supplementary material [7]

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. For more details, we refer to López-Ibáñez et al. [16].

7.1 Graphical analysis on bTSP instances

For the isometric bTSP, we compare the best strategy AN-TPLS-1seed, with the second best, *1to2*. Figure 12 shows the differences in the EAFs of these two strategies for one isometric instance. In the top plot, we show the differences after 15 scalarizations out of 30, whereas the bottom plot compares the final quality. The EAF differences after 15 scalarizations show that *1to2* simply does not cover a significant part of the objective space, whereas AN-TPLS-1seed covers the front equally in all directions.

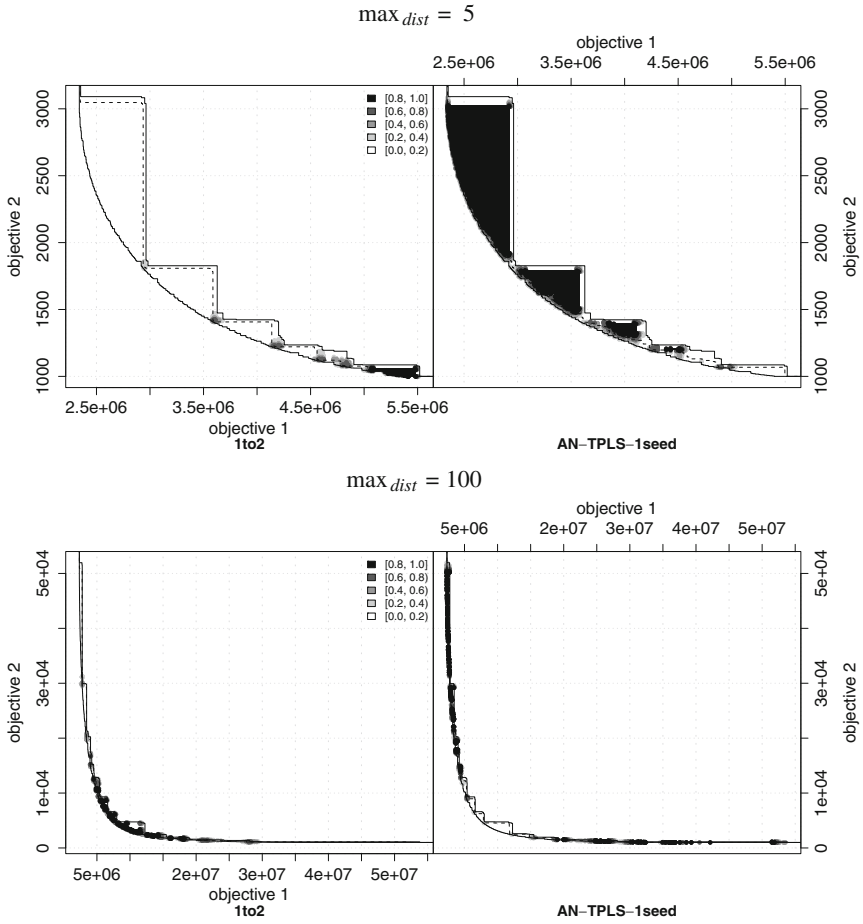

Fig. 14 EAF differences for two anisometric TSP instances ($\max_{\text{dist}} = 5$, *top*, and $\max_{\text{dist}} = 100$, *bottom*), after completing 30 scalarizations. Strategies are *1to2* (*left*) and AN-TPLS-1seed (*right*). Plots for other instances are available as supplementary material [7]

Here we color in black the region of the objective space attained by more than 80% of the runs of each algorithm, to help to visualize this behavior. This plot and the ones for other instances [7] show very clearly the lack of the anytime property in *1to2*, and the much better anytime behavior of AN-TPLS-1seed. The EAF differences after completion of the 30 scalarizations show differences in favor of both algorithms along the whole Pareto front. In this case, *1to2* appears to be better in the center of the Pareto front, whereas the adaptive TPLS finds better solutions along the extremes. Similar results are observed for other instances [7].

For anisometric instances, we first give in Fig. 13 plots that show the EAF differences between AN-TPLS-2seed and AN-TPLS-1seed. These plots support the conclusion from the statistical test, namely that AN-TPLS-1seed is better than AN-

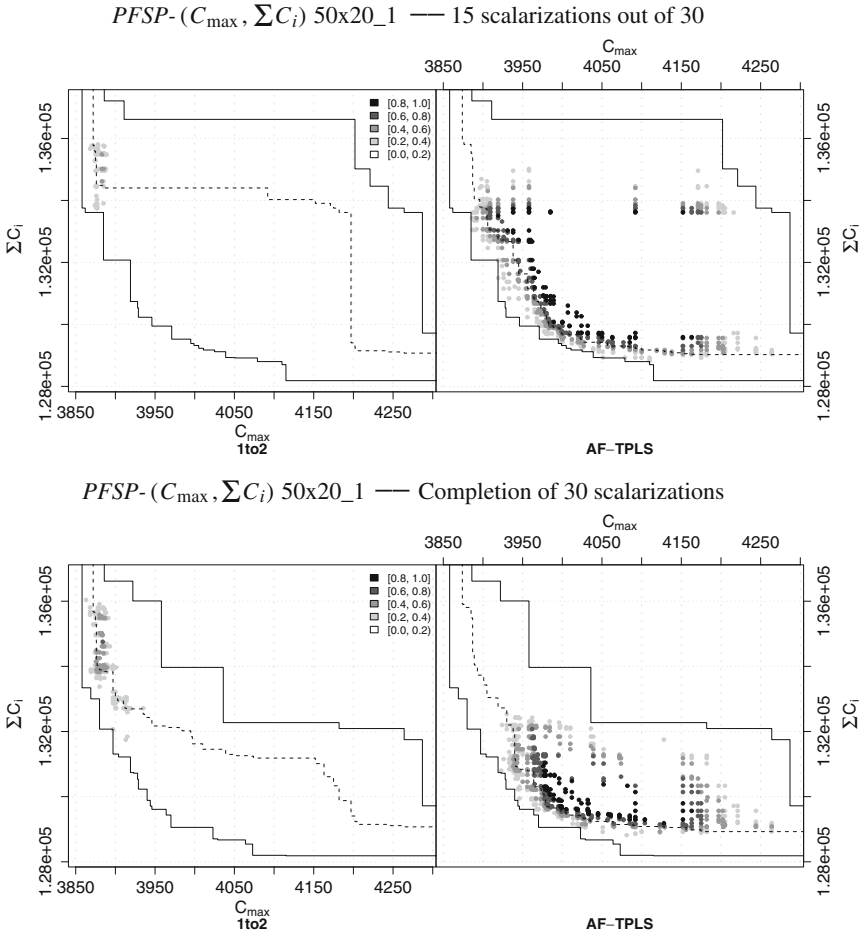

Fig. 15 EAF differences for one bPFSP instance, after 15 scalarizations out of 30 (*top plot*) and after completing the 30 scalarizations (*bottom plot*). Strategies are *1to2* (*left*) and AF-TPLS (*right*). The instance shown is `50 × 220_1` and the combination of objectives is *PFSP*-($C_{\max}$, $\sum C_i$). Plots for other instances are available as supplementary material [7]

TPLS-2seed for this problem, which is true also when varying $\max_{\text{dist}}$. In comparison with *1to2* (the best among the classical TPLS strategies for this problem), AN-TPLS-1seed is clearly better for a small values of $\max_{\text{dist}}$ (top plot of Fig. 14), whereas for large values of $\max_{\text{dist}} = 100$ (bottom plot of Fig. 14), the results are similar to the ones obtained in the isometric bTSP instances.

7.2 Graphical analysis on bPFSP

Figure 15 illustrates the EAF differences between AF-TPLS and *1to2* on one instance for *PFSP*-($C_{\max}$, $\sum C_i$), after 15 scalarizations out of 30 and after completing the 30 scalarizations. In both cases, there are strong differences in favor of AF-TPLS.

8 Conclusion

TPLS is a key component of effective bi-objective optimization algorithms [5, 8, 17]. However, the originally proposed TPLS framework has an important drawback: it requires to know in advance the available computation time to distribute appropriately the computational effort and to reach high quality results. Stopping the algorithm earlier than scheduled would lead to poor performance, as we have shown in this paper. Therefore, the original TPLS framework has poor *anytime* behavior.

In this paper, we have addressed this weakness. We have proposed new ways to define the weights used to start new scalarizations and the order in which these weights are considered.

Our first proposal, RA-TPLS, improves strongly the anytime behavior of classical TPLS strategies and, thus, outperforms these if they are stopped before completion. However, the final quality of the Pareto front approximations obtained by the classical TPLS strategies, for example D-TPLS, is better than that of RA-TPLS. Therefore, we have proposed adaptive TPLS variants that define the scalarizations adaptively in dependence of the solutions obtained in previous scalarizations.

Our adaptive TPLS variants are inspired by the dichotomic scheme proposed for exact algorithms [1]. Yet, exact algorithms require prohibitively high computation time for the problems considered here. We studied various variants of adaptive TPLS algorithms that differ in the number of initial solutions (one or two) that are used per weight, variants for focusing the search towards the center of a segment, and different ways of choosing the region of the objective space where to intensify the search. Our experimental results have unambiguously shown that (i) the adaptive TPLS variants have better anytime behavior than the non-adaptive anytime TPLS variant and (ii) the best adaptive TPLS variants typically also improve over the final quality of the approximations to the Pareto front reached by the best classical TPLS variants. Hence, our results suggest that the new adaptive TPLS variants should replace the classical variants in future TPLS applications.

Acknowledgements 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–2009–238819. 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*".

References

1. Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Manag. Sci. **25**(1), 73–78 (1979)
2. Beume, N., Naujoks, B., Emmerich M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
3. Conover, W.J.: Practical Nonparametric Statistics, 3rd edn. John Wiley & Sons, New York (1999)
4. Du, J., Leung, J.Y.T.: Minimizing total tardiness on one machine is NP-hard. Math. Oper. Res. **15**(3), 483–495 (1990)
5. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Effective hybrid stochastic local search algorithms for biobjective permutation flowshop scheduling. In: Blesa, M.J., Blum C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol. 5818, pp 100–114. Springer, Heidelberg (2009)
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Adaptive "anytime" two-phase local search. In: Blum, C., Battiti, R. (eds.) Learning and Intelligent Optimization, 4th International Conference, LION 4, Lecture Notes in Computer Science, vol. 6073, pp. 52–67. Springer, Heidelberg (2010)
7. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Supplementary Material: Improving the Anytime Behavior of Two-Phase Local Search. http://iridia.ulb.ac.be/supp/IridiaSupp2010-012 (2010)
8. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Comput. Oper. Res. **38**(8), 1219–1236 (2011)
9. Ehrgott, M., Gandibleux, X.: Approximative solution methods for combinatorial multicriteria optimization. TOP **12**(1), 1–88 (2004)
10. Fonseca, C.M., Paquete, L., López-Ibáñez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006), pp. 1157–1163. IEEE Press, Piscataway (2006)
11. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**, 117–129 (1976)
12. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler, E., Deb, K., Thiele, L., Coello, C.A., Corne, D. (eds.) Evolutionary Multi-criterion Optimization (EMO 2001). Lecture Notes in Computer Science, vol. 1993, pp. 213–225. Springer, Heidelberg (2001)
13. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco (2005)
14. Johnson, D.S.: Optimal two- and three-stage production scheduling with setup times included. Nav. Res. Logist. Q. **1**, 61–68 (1954)
15. Knowles, J.D., Corne, D.: Properties of an adaptive archiving algorithm for storing nondominated vectors. IEEE Trans. Evol. Comput. **7**(2), 100–116 (2003)
16. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 209–222. Springer, Berlin (2010)
17. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics **16**(3), 475–510 (2010)
18. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS J. Comput. **20**(3), 451–471 (2008)
19. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca. C.M., et al. (eds.) Evolutionary Multi-criterion Optimization (EMO 2003). Lecture Notes in Computer Science, vol. 2632, pp. 479–493. Springer, Heidelberg (2003)
20. 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, pp. 29–1—29–15. Chapman & Hall/CRC, Boca Raton (2007)

21. Paquete, L., Stützle, T.: Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Comput. Oper. Res. **36**(9), 2619–2631 (2009)
22. 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. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 177–200. Springer(2004)
23. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **177**(3), 2033–2049 (2007)
24. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Mag. **17**(3), 73–83 (1996)
25. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto evolutionary algorithm. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)