SCISPACE
formerly Typeset

Open access • Journal Article • DOI:10.1007/S10472-011-9235-0

Improving the anytime behavior of two-phase local search — Source link

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

Institutions: Université libre de Bruxelles

Published on: 01 Feb 2011 - Annals of Mathematics and Artificial Intelligence (Springer Netherlands)

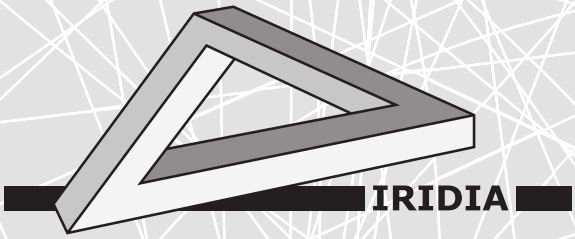
Topics: Local search (optimization)

Related papers:

- Performance assessment of multiobjective optimizers: an analysis and review
- Two-phase Pareto local search for the biobjective traveling salesman problem
- A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems
- Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study
- Improving the Anytime Behavior of

Share this paper:

View more about this paper here: https://typeset.io/papers/improving-the-anytime-behavior-of-two-phase-local-search-zhvil0mxdh



Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Improving the Anytime Behavior of Two-Phase Local Search

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

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2010-022

October 2010

Submitted to *Annals of Mathematics and Artificial Intelligence*

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2010-022

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

Improving the Anytime Behavior of Two-Phase Local Search

Jérémie Dubois-Lacoste jeremie.dubois-lacoste@ulb.ac.be

Manuel López-Ibáñez manuel.lopez-ibanez@ulb.ac.be

Thomas Stützle stuetzle@ulb.ac.be

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

October 2010

Abstract

Algorithms based on the two-phase local search (TPLS) framework are a powerful method to efficiently tackle bi-objective combinatorial optimization problems. TPLS algorithms solve a sequence of scalarizations, that is, weighted sum aggregations, of the bi-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, with the aim of filling the "largest gap" in the current approximation to the Pareto front. The experimental setup considers problems with strong differences in the shape of the Pareto front, and the analysis indicates which adaptive strategy is the best for each shape. The results presented here show that the best adaptive TPLS variants are superior to the "classical" TPLS strategies in terms of anytime behavior, and also match, and often surpass, them in terms of final quality, even if the latter run until completion.

1 Introduction

Two-phase local search (TPLS) [16] is a stochastic local search (SLS) method for tackling multi-objective combinatorial optimization problems in terms of Pareto-optimality [17]. 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) [14], its variant with three objectives [18], and the bi-objective permutation flow-shop scheduling problem (bPFSP) [4].

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 [16]. 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 [20] 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 [5]. 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 since, in some sense, they are not equally hard. 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. Our intuition suggests that the performance of the different variants depends strongly on the shape of the Pareto front and the distribution of solutions along it. We therefore study two classes of bTSP instances. In isometric bTSP instances, the distance matrices of both objectives are generated in the same way and have a similar range of values. Hence, the distribution of solutions along the Pareto front is similar in both objectives. In anisometric bTSP instances, the two distance matrices are of different type and with very different range of values. As a result, both objectives are not equally difficult, and the distribution of solutions varies strongly along the Pareto front. Our experiments demonstrate the importance of dynamically adapting the weights when the difficulty of the objectives is not similar and the shapes of the resulting Pareto fronts show strong "tails" in the distribution of the solutions.

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. This new method is based on an optimistic estimate of the improvement in the hypervolume indicator, instead of using the Euclidean distance between solutions as in [5], and it 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, an alternative to the original weight setting strategies of TPLS that fulfills the anytime property. 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 these strategies. In Section 6, we propose the use of an optimistic estimation of the hypervolume 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. Finally, 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 focusing on two objective function.

2.1 Bi-objective Combinatorial Optimization

In bi-objective combinatorial optimization problems, candidate solutions are evaluated according to an *objective function vector* $\vec{f} = (f_1, f_2)$. 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 $\vec{u}, \vec{v} \in \mathbb{R}^2$, we say that $\vec{u}$ *dominates* $\vec{v}$ ($\vec{u} \prec \vec{v}$) iff $\vec{u} \neq \vec{v}$ and $u_i \leq v_i,\ i = 1, 2$. When $\vec{u} \not\prec \vec{v}$ and $\vec{v} \not\prec \vec{u}$, we say that $\vec{u}$ and $\vec{v}$ are mutually *non-dominated*. For simplicity, we extend the dominance criteria to solutions, that is, a solution s dominates another one s' iff $\vec{f}(s) \prec \vec{f}(s')$. If no s' exists such that $\vec{f}(s') \prec \vec{f}(s)$, 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" $\vec{f}$ in the sense of Pareto optimality. Hence, the goal is to determine the set of all Pareto-optimal solutions, called the *Pareto front*. From this Pareto optimal set, the decision maker may choose a final solution *a posteriori*. Since this goal is in many cases computationally intractable [7], in practice the goal becomes to find the best possible approximation to the Pareto-optimal set within a specific time limit. Any set of mutually non-dominated solutions provides such an approximation, but some approximations are better than others.

Algorithm 1 Two-Phase Local Search

1: $\pi_1 := \mathsf{SLS}_1()$
2: $\pi_2 := \mathsf{SLS}_2()$
3: Add π_1, π_2 to Archive
4: **if** *1to2* **then** $\pi' := \pi_1$ **else** $\pi' := \pi_2$
5: **for** each weight λ **do**
6: $\quad \pi' := \mathsf{SLS}_\Sigma(\pi', \lambda)$
7: $\quad$ Add π' to Archive
8: **end for**
9: Filter(Archive)
10: **Output:** Archive

2.2 Two-Phase Local Search

Two-phase local search (TPLS) [16] 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 scalarizations in TPLS are defined by a sequence of weight vectors. In a bi-objective problem, a normalized weight vector is of the form $\vec{\lambda} = (\lambda, 1-\lambda)$, $\lambda \in [0,1] \subset \mathbb{R}$, and the scalar value of a solution s with objective function vector $\vec{f}(s) = (f_1(s), f_2(s))$ is computed as

$$f_\lambda(s) = \lambda \cdot f_1(s) + (1-\lambda) \cdot f_2(s). \tag{1}$$

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 denote weight vectors by their first component, the second component can be easily derived from Eq. 1.) In *2to1* the sequence is reversed. Two drawbacks of this simple strategy are that (i) the direction chosen can give a clear 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. 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. Different from the initial proposal [16], in our implementation we first generate a very good solution for each single objective problem because we have high performing algorithms for them. However, we use only one of the solutions as a starting solution for further scalarizations.

Double strategy. We denote as Double TPLS (D-TPLS) [16] 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 first TPLS pass are alternated with the weights used for the second 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 [16, 18]. 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 [5]. 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 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, in some sense filling the gaps in the Pareto front. 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.

Algorithm 2 RA-TPLS

1: $s_1 := \mathsf{SLS}_1()$
2: $s_2 := \mathsf{SLS}_2()$
3: Add s_1, s_2 to *Archive*
4: $L_0 := \{(1,0)\}$; $L_i := \emptyset \quad \forall i > 0$
5: $Sd := \{(s_1,1),(s_2,0)\}$
6: $i := 0$
7: **while** not stopping criterion met **do**
8: $(\lambda_{\text{sup}}, \lambda_{\text{inf}}) :=$ extract randomly from L_i
9: $L_i := L_i \setminus (\lambda_{\text{sup}}, \lambda_{\text{inf}})$
10: $\lambda := (\lambda_{\text{sup}} + \lambda_{\text{inf}})/2$
11: $s := \mathsf{ChooseSeed}(Sd, \lambda)$
12: $s' := \mathsf{SLS}_\Sigma(s, \lambda)$
13: Add s' to *Archive*
14: $Sd := Sd \cup (s', \lambda)$
15: $\mathsf{Filter}(Sd)$
16: $L_{i+1} := L_{i+1} \cup (\lambda_{\text{sup}}, \lambda) \cup (\lambda, \lambda_{\text{inf}})$
17: **if** $L_i = \emptyset$ **then** $i := i + 1$
18: **end while**
19: $\mathsf{Filter}(Archive)$
20: **Output:** *Archive*

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, $\mathsf{SLS}_1()$ 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 following 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 $\mathsf{ChooseSeed}$ uses this weight λ to choose a solution from the set of initial solutions Sd. To do so, first $\mathsf{ChooseSeed}$ finds the two non-dominated

solutions that were obtained from scalarizations using the weights closest to λ:

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

Next, ChooseSeed calculates the scalar value of s_{sup} and s_{inf} according to the new weight λ following Eq. 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, which is filtered to keep only non-dominated ones. 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 [5]. 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 $G = (V, A)$ with $n = |V|$ nodes $\{v_1, \ldots, v_n\}$, a set of arcs A, and a cost associated to each arc $c(v_i, v_j)$, 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 arc, 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 for each cost component.

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 [7, 18]. Moreover, TPLS is a main component of the current state-of-the-art algorithm [14] for the bTSP.

Isometric and Anisometric bTSP Instances

We created 10 Euclidean bTSP instances by generating two sets of 1000 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 several 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 $\max_{\text{dist}} \in \{5, 10, 25, 100\}$, as in [18]. Given the different range of both distance matrices, we call these instances *anisometric*. We generated 10 instances of 1000 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) based on a first-improvement 3-opt algorithm [11].[1] In order to speed up the algorithm, we compute a new distance matrix for each scalarization and we recompute the candidate sets used by the speed-up techniques of this ILS algorithm. Each scalarization runs for 1000 ILS iterations (equal to the number of nodes in the instance). With our implementation and computing environment, 1000 ILS iterations require roughly 0.5 CPU seconds. Each of the two initial solutions is generated by running ILS for 2000 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 [8, 21]. In the bi-objective space, the hypervolume measures the area of the objective space weakly dominated by 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.

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

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 [6]. For isometric bTSP instances, according to these plots, 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, there is a clear difference between strategies *1to2* and *2to1*, and 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. Smaller values of $\max_{\text{dist}}$ result in a larger number of optimal solutions for the second objective. Hence, the first scalarization of *2to1* will return one of these optima and a very good value of the first objective, producing a huge initial improvement of the hypervolume. Several subsequent scalarizations of *2to1* will only slightly improve 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 accepted, 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 minimum of the second objective, and very slow progress when being close to the minimum of the first objective. This effect will be even stronger in the case of the bPFSP.

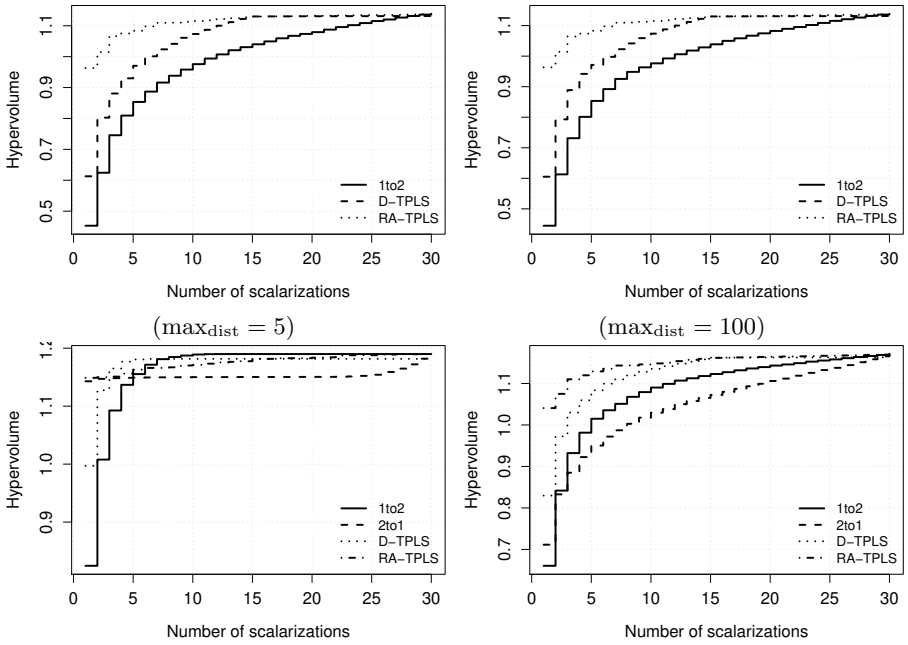


Figure 1: Development of the hypervolume over the number of scalarizations for *1to2*, D-TPLS and RA-TPLS for two isometric (top plots) and two anisometric (bottom plots) bTSP instances. For anisometric instances we also plot the results of *2to1*, since they differ strongly from *1to2*. For isometric 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 [6]).

3.2.2 Case Study: Bi-objective Permutation Flow-shop Scheduling Problem

The flow-shop scheduling problem [12] models a very common type of environment in the industry and 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 [9].

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 [9]. 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 [3].

As a case study, in this paper we focus on 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 sum of completion times 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 [4]. A crucial component in this hybrid algorithm is TPLS using effective iterated greedy (IG) algorithms [19] 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 ($\lambda = 1$ and $\lambda = 0$) by running 1 000 iterations of the corresponding IG algorithm. Then, it performs 30 scalarizations, each scalarization running for 500 IG iterations.

We generate 10 benchmark instances with $n = 50$ and $m = 20$ (`50x20`), and 10 instances with $n = 100$ and $m = 20$ (`100x20`), following the procedure

described by Minella et al. [15]. These instances are available as supplementary material [6]. 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.

Experimental Evaluation of RA-TPLS on bPFSP Instances

As for the bTSP, 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 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 two objectives have different difficulty 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 [5]. 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 [5] 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 [14]. 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 defined by s_1 and s_2 in the objective space, that is:

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

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

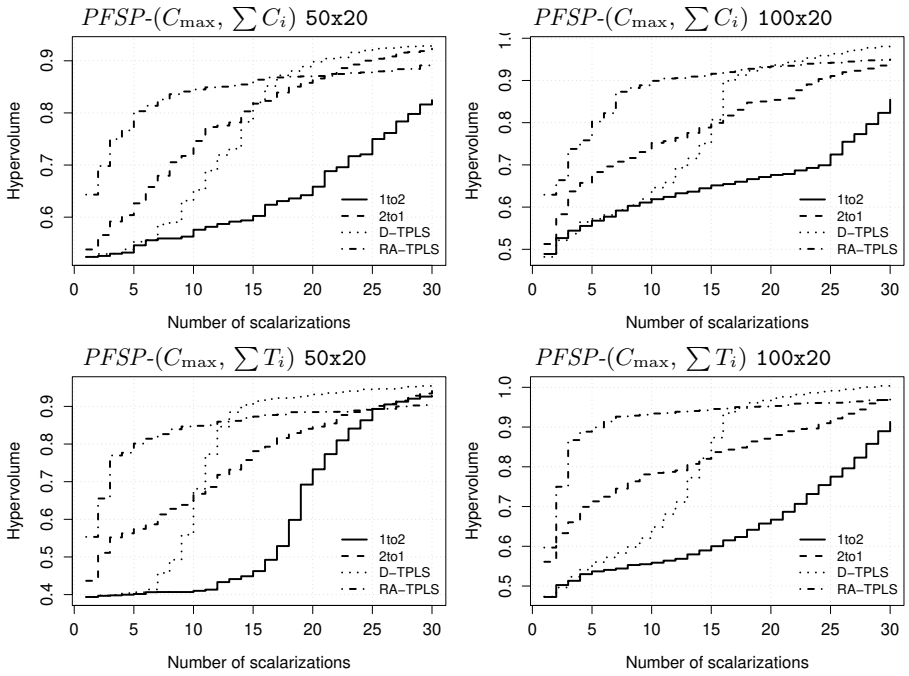


Figure 2: Development of the average hypervolume over the number of scalarizations for *1to2*, *2to1*, D-TPLS, and RA-TPLS for bPFSP. Results are given for one instance of size 50x20 (left column) and one instance of size 100x20 (right column). The problems are *PFSP*-($C_{\max}$, $\sum C_i$) (top plots) and *PFSP*-($C_{\max}$, $\sum T_i$) (bottom plots).

Algorithm 3 Adaptive "Anytime" TPLS Strategy

$s_1 := \mathsf{SLS}_1()$
$s_2 := \mathsf{SLS}_2()$
Add s_1, s_2 to *Archive*
$S := \{(s_1, s_2)\}$
while not stopping criteria met **do**
  $(s_{\text{sup}}, s_{\text{inf}}) := \arg\max_{(s,s')\in S} \|(s, s')\|$
  Calculate λ perpendicular to $\vec{f}(s_{\text{sup}})\vec{f}(s_{\text{inf}})$ following Eq. 3
  if one_seed_case **then**
    $s := \mathsf{ChooseRandomly}(s_{\text{sup}}, s_{\text{inf}})$
    $s' := \mathsf{SLS}_\Sigma(s, \lambda)$
    Add s' to *Archive*
    $\mathsf{Update}(S, s')$
  else
    $s'_{\text{sup}} := \mathsf{SLS}_\Sigma(s_{\text{sup}}, \lambda)$
    $s'_{\text{inf}} := \mathsf{SLS}_\Sigma(s_{\text{inf}}, \lambda)$
    Add s'_{sup} and s'_{inf} to *Archive*
    $\mathsf{Update}(S, s'_{\text{sup}})$
    $\mathsf{Update}(S, s'_{\text{inf}})$
  end if
end while
$\mathsf{Filter}(\mathit{Archive})$
Output: *Archive*

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 [14] 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 solution. 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 [5]. 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 $(s_{\text{sup}}, s_{\text{inf}}) \in S$ that defines the "largest gap" in the objective space, using a given norm $\|(s, s')\|$ 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 [5], and we use this norm in the following. However, we propose a new alternative in

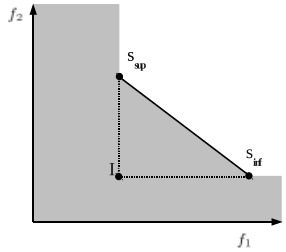


Figure 3: Only solutions in the gray area are accepted as initial solutions for further scalarizations
(See the text for details).

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 defined by s_{sup} and s_{inf} in the objective space, following Eq. 3. Next, the underlying single-objective SLS algorithm, SLS_Σ, is run either once, using the weight λ and 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 with s', and any pair of solutions (weakly) dominated by s' is removed. The *dichotomic* scheme [1, 14] only accepts solutions for inclusion in S if they lie *within* the triangle defined by the solutions s_{sup} and s_{inf}, and their local ideal point (see Figure 3). Solutions outside the gray area are either dominated or not supported (not optimal for any scalarization). Heuristic algorithms may, however, generate supported solutions that are in the gray area *outside* the triangle; 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 segment $(s_1, s_2) \in S$ with $f_1(s_1) < f_1(s') < f_1(s_2)$ is removed, and two new segments (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 segments 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 this algorithm AN-TPLS-2seed in what follows (for adaptive normal TPLS), and we call AN-TPLS-1seed its variant using only one initial solution.

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 Figure 5,

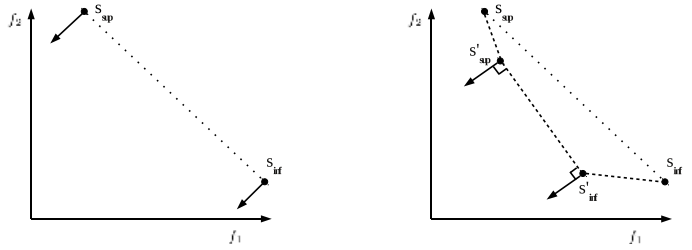


Figure 4: A single iteration of the AN-TPLS-2seed algorithm. On the left the state before the iteration and on the right after S has been updated. The next segment that will be considered is (s'_1, s'_2) because of its larger distance.

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 that are very close to each other 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 increasing $\max_{\text{dist}}$, 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 Figure 6. For the bPFSP, 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

We examine the differences between AN-TPLS-1seed and AN-TPLS-2seed in more detail. In particular, we examine the distribution of the objective vectors obtained after each scalarization in AN-TPLS-2seed for the bTSP and the bPFSP.

We first run AN-TPLS once and record the solutions and weights generated.

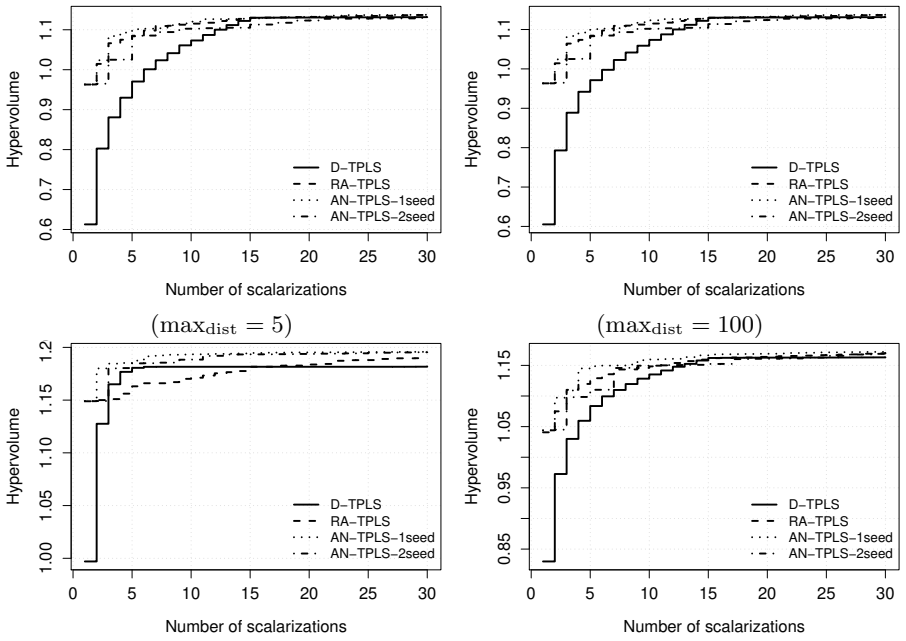


Figure 5: Development of the hypervolume over the number of scalarizations for D-TPLS, RA-TPLS, AN-TPLS-2seed and AN-TPLS-1seed for two isometric TSP instances and two anisometric TSP instances. Anisometric instances with intermediate value of $\max_{\text{dist}}$ show a compromise trend between the two extremes shown here (see supplementary material [6]).

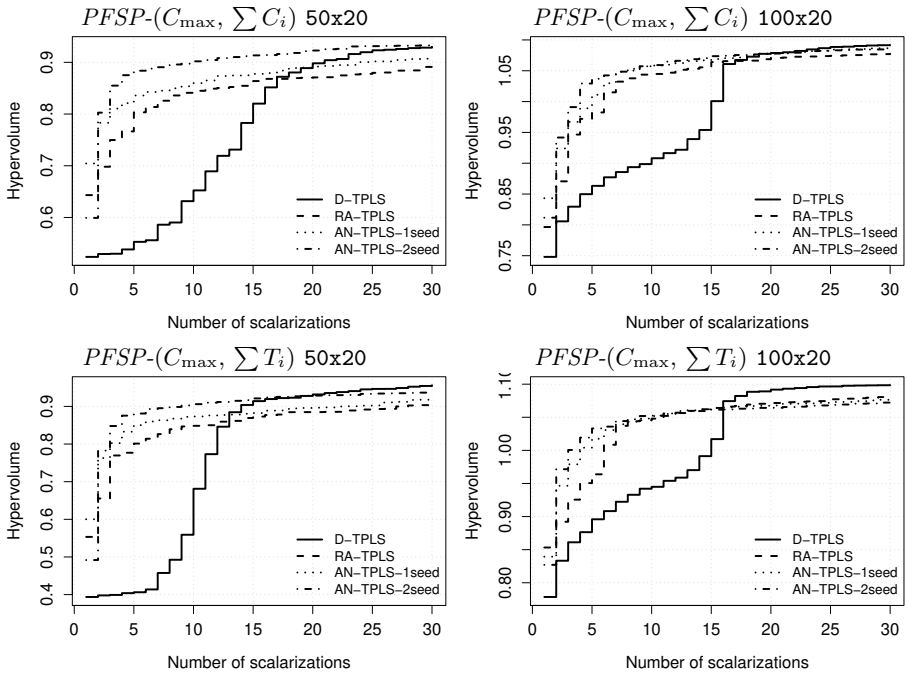


Figure 6: Development of the hypervolume over the number of scalarizations for D-TPLS, RA-TPLS, AN-TPLS-2seed and AN-TPLS-1seed for bPFSP. Results are given for one instance of size `50x20` (left column) and one instance of size `100x20` (right column). The problems are *PFSP*-($C_{\max}$, $\sum C_i$) (top plots) and *PFSP*-($C_{\max}$, $\sum T_i$) (bottom plots).

Then, we use the two initial solutions and the first weight to perform 15 independent scalarizations starting from each of the two initial solutions. Results are plotted on the left column of Figure 7. The line indicates the direction of the scalarization (defined by the weight). The initial solutions of the scalarizations are shown in gray, and the symbols indicate the initial solution from which each new candidate solution was obtained. Next, we use the second and third weight and the third and fourth initial solution pairs to simulate 15 scalarizations for each combination of initial solution and weight. The resulting solutions after tackling the scalarized problems are shown on the right column of Figure 7. Solutions that have an additional small cross inside originate from the scalarization shown with a dotted line. These plots clearly show that for the bTSP, all resulting objective vectors for the same scalarization fall into a very narrow area of the objective space, independent from which initial solution they started. Hence, there is no evident advantage in using two different initial solutions with the same weight. The situation is very different for the bPFSP. As it is clearly visible, there is a large variability of the resulting objective vectors even for the same initial solution. More importantly, the objective vectors resulting from two different initial solutions, even when tackling the same scalarized problem, result in two clearly distinct clusters. Hence, tackling a scalarization from two different initial solutions, is advantageous in the case of the bPFSP.

This insight might be used to define when it is better to use one or two initial solutions by, for example, taking into account the distance among objective vectors after the first scalarization of AN-TPLS-1seed, and dynamically deciding whether to perform an additional scalarization starting from the other initial solution. However, we do not explore these possibilities in this paper.

4.5 Adaptive Focus TPLS

If two adjacent segments in S are almost parallel, in AN-TPLS-2seed two scalarizations will be 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 situations actually occur 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* (AF-TPLS).

Given a segment $(s_1, s_2) \in S$, 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 Eq. 3, and θ is a parameter that modifies λ towards the center of the segment (see Figure 8).

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

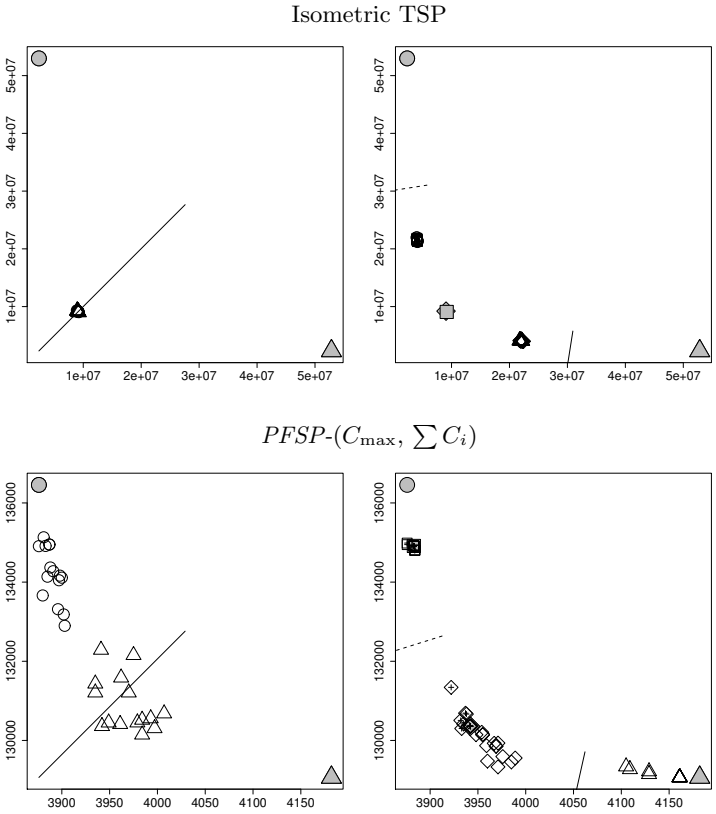


Figure 7: Distribution of solutions in the objective space after the first (left plots), and the second and third (right plots) scalarizations. The top plots show an isometric bTSP instance, and the bottom ones show an instance for the *PFSP*-($C_{\max}$, $\sum C_i$). Other instances show the same trend, which demonstrate strong differences between the two problems.

according to weight λ_1, while the run starting from with 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 [6]. 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 Figure 9, 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 out-

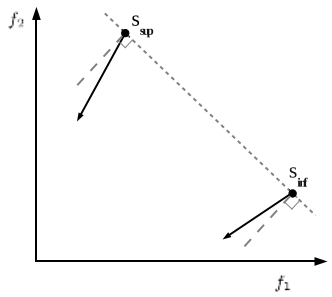


Figure 8: AF-TPLS strategy: the two weights are "focused" to the center of the segment.

perform it. Furthermore, AF-TPLS reaches a final quality often equal to or better than the one reached by D-TPLS. (The instance `100x20_1` of problem *PFSP*-($C_{\max}$, $\sum T_i$) (bottom-right in Fig. 9) is actually the only instance 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 [2]. 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, and 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 [6]).

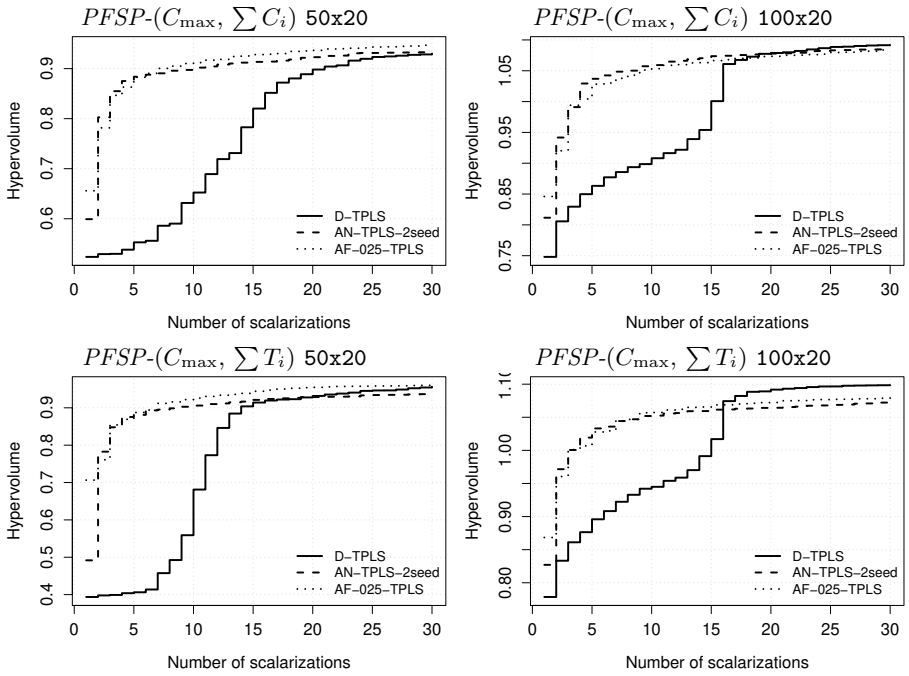


Figure 9: Development of the hypervolume over the number of scalarizations for D-TPLS, AN-TPLS-2seed and AF-TPLS. Results are given for one instance of size `50x20` (left column) and one instance of size `100x20` (right column). The problems are *PFSP*-($C_{\max}$, $\sum C_i$) (top plots) and *PFSP*-($C_{\max}$, $\sum T_i$) (bottom plots).

Table 1: Statistical analysis for the isometric bTSP. 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 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, $\Delta R_\alpha = 0$.

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)

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 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 for $\max_{\text{dist}}$. 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 `50x20` and `100x20`, 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 (using $\theta = 0.25$). The results are given in Table 3.

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

Table 2: Statistical analysis for the anisometric bTSP. 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 significantly better than the other ones is indicated in bold face.

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)

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 quality improvement. A measure of the quality of the current Pareto front approximation is the hypervolume, and therefore, selecting the pair of seeds that may lead to the largest improvement of hypervolume should intuitively lead to improving the quality of the approximation. Assuming that the new solution found is within the square defined by the two seeds, the maximum improvement in terms of hypervolume is proportional to the area of the square. Hence, using normalized objective values, we compute this norm as follows:

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

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 10 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 slightly, however consistently, better than the one that uses the Eu-

Table 3: Statistical analysis for the bPFSP. 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 which are not significantly different to the best one are indicated in bold face. See the text for details.

N_{scalar}	ΔR_α	**Strategies** (ΔR)
($C_{\max}$, $\sum C_i$) 50x20		
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)
($C_{\max}$, $\sum C_i$) 100x20		
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)
($C_{\max}$, $\sum T_i$) 50x20		
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)
($C_{\max}$, $\sum T_i$) 100x20		
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)

Table 4: Statistical analysis for the isometric bTSP. For each number of scalarizations, strategies are ordered according to the rank obtained. For this problem, the first strategy is always the first one, the second is always the second one, and so on.

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)

clidean distance. For AF-TPLS on the bPFSP, results are not as consistent as for the bTSP (additional plots are available as supplementary material [6]).

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

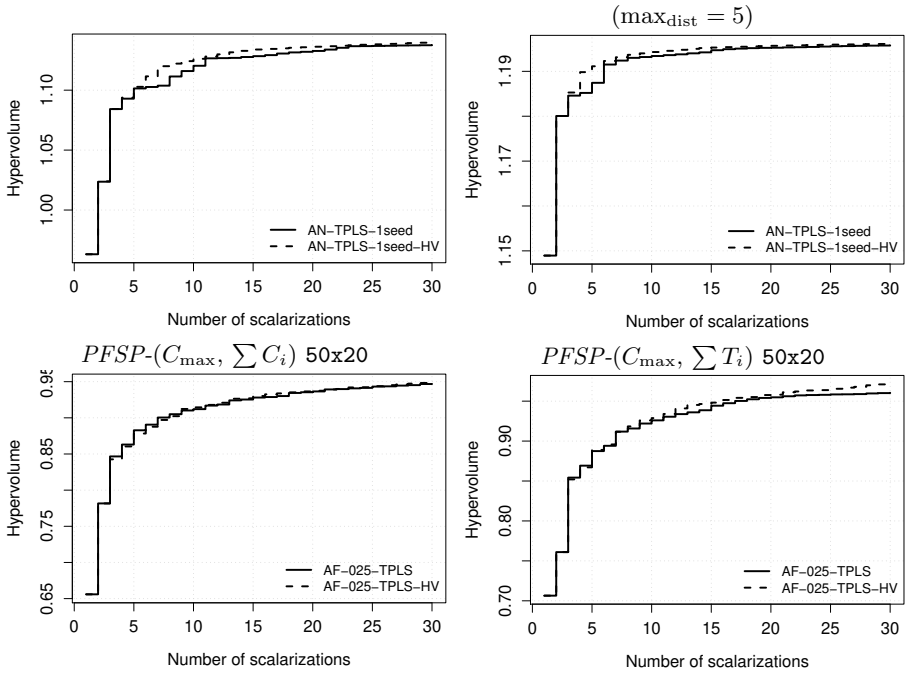


Figure 10: Development of the hypervolume over the number of scalarizations for AN-TPLS-1seed using Euclidean distance and $\|(s, s')\|_{\mathcal{HV}}$ for one isometric TSP instance (top-left), one anisometric TSP instance (top-right), and one instance of bPFSP with the two different combinations of objectives.

Table 5: Statistical analysis for the anisometric bTSP. 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 significantly better than the other ones is indicated in bold face.

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)

D-TPLS. The improvement is not as consistent as for the bTSP. However, AF-TPLS$_{\mathcal{HV}}$ is most often the best-ranked strategy, never being 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 [10]. Examining the differences between the EAFs of two algorithms allows us to identify regions of the objective space where one algorithm performs better than another. Given a pair of algorithms, the differences in favor of each algorithm are plotted side-by-side and the magnitude of the difference is encoded in gray levels. For more details, we refer to López-Ibáñez et al. [13].

Table 6: Statistical analysis for the bPFSP. 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 which are not significantly different to the best one are indicated in bold face. See the text for details.

N_{scalar}	ΔR_α	**Strategies** (ΔR)
($C_{\max}$, $\sum C_i$) `50x20`		
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)
($C_{\max}$, $\sum C_i$) `100x20`		
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**
($C_{\max}$, $\sum T_i$) `50x20`		
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**
($C_{\max}$, $\sum T_i$) `100x20`		
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$_{\mathcal{HV}}$**

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 11 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. 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 [6] 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 [6].

For anisometric instances, we first give in Fig. 12 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-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

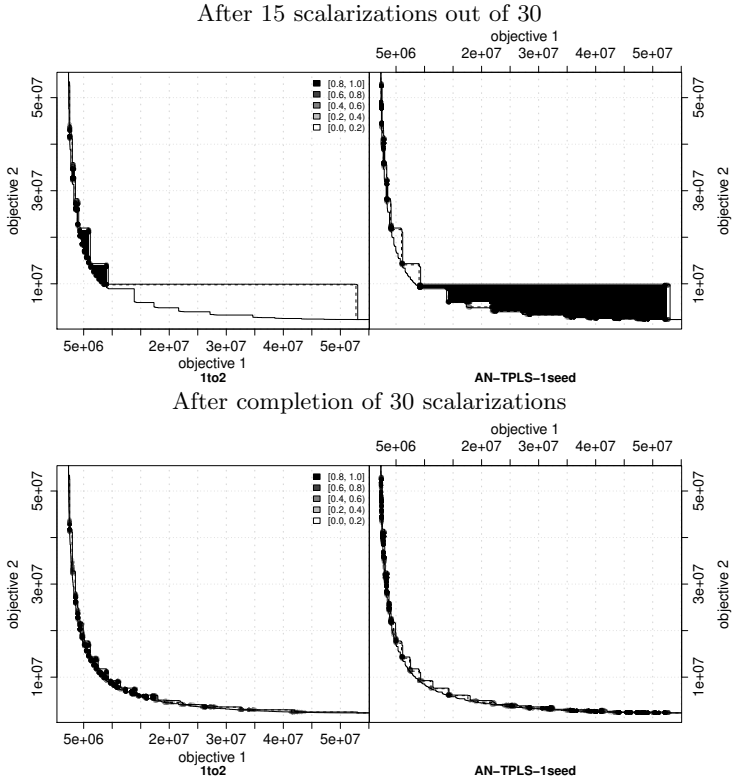


Figure 11: EAF differences for one isometric TSP instance, after 15 scalarizations (top) and after 30 scalarizations (bottom). Strategies are *1to2* (left) and AN-TPLS-1seed (right). Plots for other instances are available as supplementary material [6].

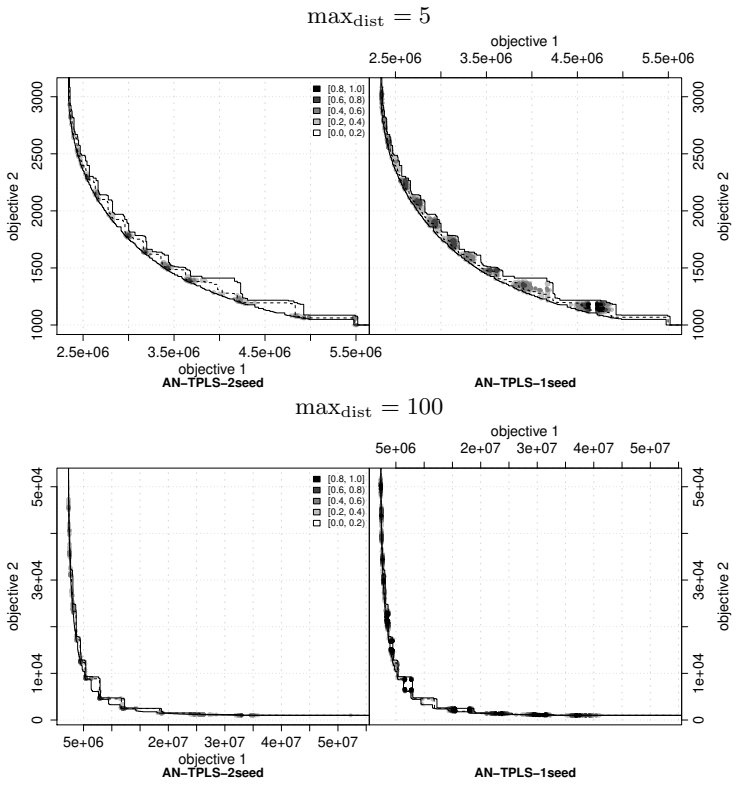


Figure 12: EAF differences for two anisometric TSP instances, after completing 30 scalarizations. Strategies are AN-TPLS-2seed (left) and AN-TPLS-1seed (right). Other values of $\max_{\text{dist}}$ show similar trends. Plots for other instances are available as supplementary material [6].

problem), AN-TPLS-1seed is clearly better for a small values of $\max_{\text{dist}}$ (top plot of Fig. 13), whereas for large values of $\max_{\text{dist}} = 100$ (bottom plot of Fig. 13), the results are similar to the ones obtained in the isometric bTSP instances.

7.2 Graphical analysis on bPFSP

Figure 14 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.

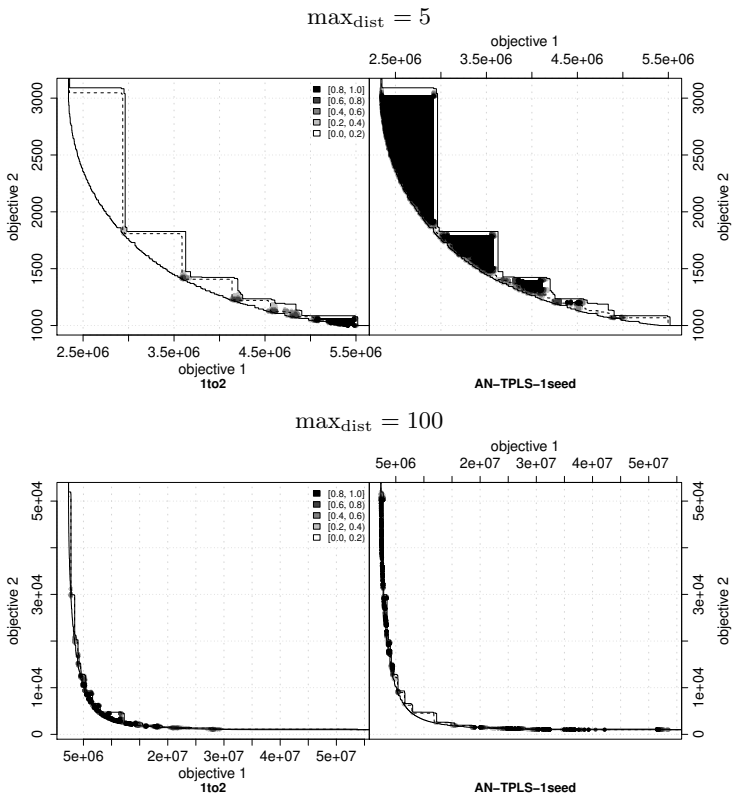


Figure 13: EAF differences for two anisometric TSP instances, after completing 30 scalarizations. Strategies are *1to2* (left) and AN-TPLS-1seed (right). Plots for other instances are available as supplementary material [6].

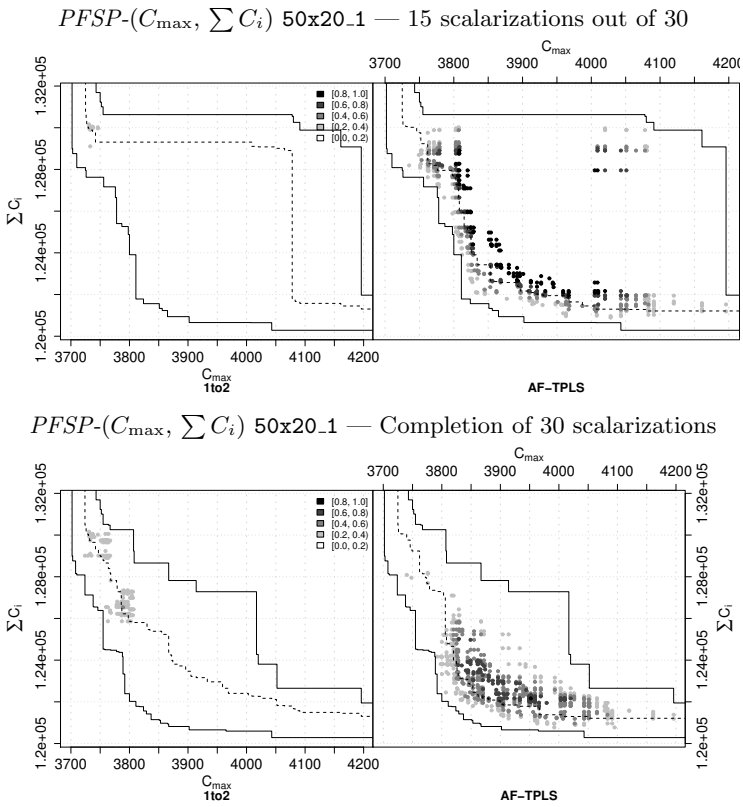


Figure 14: EAF differences for one bPFSP instance, after 15 scalarizations out of 30 (top plot) and after completing the 30 scalarizations (bottom plot). Strategies are *1to2* (left) and AF-TPLS (right). The instance shown is `50x20_1` and the combination of objectives is *PFSP*-($C_{\max}$, $\sum C_i$). Plots for other instances are available as supplementary material [6].

8 Conclusion

TPLS is a key component for effective bi-objective optimization algorithms [4, 14]. 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 show in this paper. Therefore, the original TPLS framework has poor *anytime* behavior. In this paper, we address this weakness. We propose new ways to define the weights used to start new scalarizations and the order in which these weights are considered. Our new strategies are designed with the main goal of improving substantially the anytime behavior of TPLS. However, according to our analysis, the best of our proposed strategies also improve upon the quality of the Pareto front approximations reached by the "classical" TPLS strategies. These improvements are experimentally shown using as benchmarks two well-known problems, the bi-objective Traveling Salesman Problem (bTSP) and the bi-objective Permutation Flow-Shop Problem (bPFSP) using different types of instances (in the case of the bTSP) and different combinations of objectives (in the case of the bPFSP). These two problems are important because of the amount of research they have already attracted, and because TPLS is an essential part of the current state-of-the-art algorithms for these problems—any improvements made to the TPLS framework would further improve those algorithms.

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 of, for example, D-TPLS is better than that of RA-TPLS. However, when choosing adaptively, in dependence of the distribution of already obtained points, the scalarizations, further improvements over RA-TPLS are possible. Our adaptive TPLS schemes are inspired by the dichotomic scheme proposed for exact algorithms [1]. We studied various variants of adaptive TPLS algorithms that differ in the number of initial solutions (one or two) that are used to tackle a scalarization, schemes 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 unambiguously show that (i) the adaptive schemes show better anytime behavior that a regular scheme and (ii) the best adaptive schemes typically also improve over the final quality of the approximations to the Pareto front reached by the best classical TPLS schemes. Hence, our results suggest that the new adaptive TPLS schemes should replace the classical schemes in future TPLS applications.

Acknowledgments.

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium, and by the MIBISOC network, an Initial Training Network

funded by the European Commission, grant PITN–GA–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 YP, Nair KPK (1979) Bicriteria transportation problem. Management Science 25(1):73–78

[2] Conover WJ (1999) Practical Nonparametric Statistics, 3rd edn. John Wiley & Sons, New York, NY

[3] Du J, Leung JYT (1990) Minimizing total tardiness on one machine is NP–hard. Mathematics of Operations Research 15(3):483–495

[4] Dubois-Lacoste J, López-Ibáñez M, Stützle T (2009) Effective hybrid stochastic local search algorithms for biobjective permutation flowshop scheduling. In: Blesa MJ, Blum C, Di Gaspero L, Roli A, Sampels M, Schaerf A (eds) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol 5818, Springer, Heidelberg, Germany, pp 100–114

[5] Dubois-Lacoste J, López-Ibáñez M, Stützle T (2010) Adaptive "anytime" two-phase local search. In: Blum C, Battiti R (eds) Learning and Intelligent Optimization, 4th International Conference, LION 4, Springer, Heidelberg, Germany, Lecture Notes in Computer Science, vol 6073, pp 52–67

[6] Dubois-Lacoste J, López-Ibáñez M, Stützle T (2010) Supplementary material: Improving the Anytime Behavior of Two-Phase Local Search. http://iridia.ulb.ac.be/supp/IridiaSupp2010-012

[7] Ehrgott M, Gandibleux X (2004) Approximative solution methods for combinatorial multicriteria optimization. TOP 12(1):1–88

[8] Fonseca CM, Paquete L, López-Ibáñez M (2006) An improved dimension -sweep algorithm for the hypervolume indicator. In: IEEE Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, pp 1157–1163

[9] Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1:117–129

[10] Grunert da Fonseca V, Fonseca CM, Hall AO (2001) Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler E, Deb K, Thiele L, Coello CA, Corne D (eds) Evolutionary Multicriterion Optimization (EMO 2001), Lecture Notes in Computer Science, vol 1993, Springer, Heidelberg, Germany, pp 213–225

[11] Hoos HH, Stützle T (2005) Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA

[12] Johnson DS (1954) Optimal two- and three-stage production scheduling with setup times included. Naval Research Logistics Quarterly 1:61–68

[13] López-Ibáñez M, Paquete L, Stützle T (2010) 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, Springer, Berlin, Germany, pp 209–233

[14] Lust T, Teghem J (2010) Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 16(3):475–510

[15] Minella G, Ruiz R, Ciavotta M (2008) A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20(3):451–471

[16] Paquete L, Stützle T (2003) A two-phase local search for the biobjective traveling salesman problem. In: Fonseca CM, et al (eds) Evolutionary Multi-criterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol 2632, Springer, Heidelberg, Germany, pp 479–493

[17] Paquete L, Stützle T (2007) Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In: Gonzalez TF (ed) Handbook of Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC, Boca Raton, FL, pp 29–1—29–15

[18] Paquete L, Stützle T (2009) Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Computers & Operations Research 36(9):2619–2631

[19] Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3):2033–2049

[20] Zilberstein S (1996) Using anytime algorithms in intelligent systems. AI Magazine 17(3):73–83

[21] Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. IEEE Transactions on Evolutionary Computation 3(4):257–271