Pareto Local Search Algorithms for Anytime Bi-objective Optimization

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

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

Abstract. Pareto local search (PLS) is an extension of iterative improvement methods for multi-objective combinatorial optimization problems and an important part of several state-of-the-art multi-objective optimizers. PLS stops when all neighbors of the solutions in its solution archive are dominated. If terminated before completion, it may produce a poor approximation to the Pareto front. This paper proposes variants of PLS that improve its anytime behavior, that is, they aim to maximize the quality of the Pareto front at each time step. Experimental results on the bi-objective traveling salesman problem show a large improvement of the proposed anytime PLS algorithm over the classical one.

1 Introduction

Multi-objective optimization deals with problems for which solutions are evaluated according to multiple criteria. These criteria are often in conflict and, hence, different solutions can show different possible trade-offs between the objectives that must be optimized. If there is no *a priori* knowledge about the preferences of the decision maker, multi-objective problems are usually tackled by determining the set of all Pareto optimal solutions from which the decision maker can select, *a posteriori*, one solution or extract some conclusions.

Pareto local search (PLS) is an extension of iterative improvement algorithms for single-objective problems to the multi-objective case [13]. PLS produces high-quality results when used as a stand-alone algorithm [13], and even better results when used as a component of hybrid algorithms, where PLS starts from a good set of initial solutions. In fact, PLS is a crucial component of the best algorithms known for the multi-objective traveling salesman problem [12] with two and three objectives, various bi-objective permutation flowshop problems [4], and the bi-objective multi-dimensional knapsack problem [11].

Even when starting from a good approximation to the Pareto front, PLS can require a long time to terminate [13,4]. Hence, many applications define an additional termination criterion, either in terms of a maximum number of solutions to be explored or computation time. In many real-life situations, however, the available computation time may be unknown, and it is therefore desirable to obtain the best possible output whenever the algorithm is stopped. Unfortunately, PLS was originally designed without taking into account its *anytime behavior* [15].

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 206–217, 2012.

Although there is much work on anytime algorithms for single-objective optimization problems [15,9], there is little research on anytime multi-objective optimization algorithms [5]. Some papers on multi-objective evolutionary algorithms report the quality obtained along time among other evaluations of the performance, to show the convergence ability of the algorithms, but their goal is not to obtain a good anytime behavior. By focusing on improving the anytime behavior of PLS, similarly to [5], our aim is also to better formalize the notion of anytime behavior for multi-objective algorithms.

In this paper, we study alternatives to the algorithmic components of the classical PLS with the goal of improving its anytime behavior. In particular, we propose variants that switch strategies during the run of the algorithm. We consider the bi-objective traveling salesman problem (bTSP) as a case study. The traveling salesman problem (TSP) is a well-known $\mathcal{NP}$-hard combinatorial problem widely used to assess the performance of optimization algorithms and meta-heuristics. In its bi-objective variant, two cost values are assigned to each edge of a graph, and each of the two objective functions is computed with respect to the corresponding cost value. The bTSP is a prominent benchmark problem in the study of multi-objective optimization algorithms [7,14,5].

2 Anytime Pareto Local Search

PLS is an iterative improvement method for solving multi-objective combinatorial optimization problems. The acceptance criterion in PLS is based on Pareto dominance. Given two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$, we say that $\boldsymbol{u}$ *dominates* $\boldsymbol{v}$ ($\boldsymbol{u} \prec \boldsymbol{v}$) iff $\boldsymbol{u} \neq \boldsymbol{v}$ and $u_i \leq v_i,\ i = 1, 2$, assuming, without loss of generality, that both objectives must be minimized. Algorithm 1 illustrates the PLS framework. Given is an initial set of mutually nondominated solutions, called *archive*. The solutions are initially marked as unexplored (line 2). PLS then iteratively applies the following steps. First, a solution s is chosen randomly among all unexplored ones (*selection step*, line 5). Then, the neighborhood of s, $\mathcal{N}(s)$, is fully explored and all neighbors that are nondominated w.r.t. s and any solution in the archive $\mathcal{A}$ are added to $\mathcal{A}$ (lines 6 to 11). Solutions in $\mathcal{A}$ that are dominated by the newly added solutions are removed (procedure Update in line 9). Once the neighborhood of s has been fully explored, s is marked as explored (line 10). When all solutions have been explored, and no more nondominated solutions can be discovered, the algorithm stops. Next we discuss the three main algorithmic components of PLS.

Selection Step. In many combinatorial optimization problems, solutions that are close to each other in the solution space are also close in the objective space. Transferring this fact to multi-objective problems, we may expect to fill existing "gaps" around the image of a solution in the Pareto front by exploring the neighborhood of it. In the original PLS, the next solution to be explored is selected randomly among the ones not explored, without considering possibly existing "gaps". Instead, PLS could select the solution whose neighborhood has the largest potential of improving the current Pareto front approximation.

Algorithm 1. Pareto Local Search

```
1: Input: An initial set of nondominated solutions A_0
2: explored(s) := FALSE   ∀s ∈ A_0
3: A := A_0
4: repeat
5:    s := select randomly a solution from A_0      // Selection step
6:    for each s' ∈ N(s) do      // Neighborhood exploration
7:       if s ≠ s' then      // Acceptance criterion
8:          explored(s') := FALSE
9:          A := Update(A, s')
10:   explored(s) := TRUE
11:   A_0 := {s ∈ A | explored(s) = FALSE}
12: until A_0 = ∅
13: Output: A
```

A measure of the quality of a single solution is given by its hypervolume contribution. The hypervolume measures the volume of the objective space weakly dominated by a set of solutions [17]. The larger the hypervolume, the better the quality of the set. The hypervolume contribution of a single solution measures how much a solution contributes to the total hypervolume of a set. However, in the selection step of PLS we are not concerned about the hypervolume contribution of the solutions in the current archive, but on the potential hypervolume contribution of solutions that are generated by exploring the neighborhood of solutions in the archive. Since the actual hypervolume contribution that can be generated by neighborhood exploration is unknown, we "optimistically" estimate it. Given two solutions s and s' (in the biobjective case), we define the *optimistic hypervolume contribution* (ohvc) as the hypervolume contribution of the local ideal point defined by s and s' in the objective space:

$$\mathsf{ohvc}(s, s') = (f_1(s) - f_1(s')) \cdot (f_2(s') - f_2(s)), \tag{1}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are the normalized objective values for the first and second objective, respectively.

We expect to find new nondominated solutions in the region between the current solution and its closest neighbors in the objective space, and, thus, we define the *optimistic hypervolume improvement* (OHVI) of a solution s as

$$\mathsf{OHVI}(s) = \begin{cases} 2 \cdot \mathsf{ohvc}(s, s_{\text{sup}}) & \text{if } \nexists s_{\text{inf}}, \\ 2 \cdot \mathsf{ohvc}(s_{\text{inf}}, s) & \text{if } \nexists s_{\text{sup}}, \\ \mathsf{ohvc}(s, s_{\text{sup}}) + \mathsf{ohvc}(s_{\text{inf}}, s) & \text{otherwise,} \end{cases} \tag{2}$$

where s_{sup} and s_{inf} are the closest neighbors of s in the objective space from the current archive $\mathcal{A}_0$ defined as

$$\begin{aligned} s_{\text{sup}} &= \arg\min_{s_i \in \mathcal{A}_0} \{ f_2(s_i) \mid f_2(s_i) > f_2(s) \} \\ s_{\text{inf}} &= \arg\max_{s_i \in \mathcal{A}_0} \{ f_2(s_i) \mid f_2(s_i) < f_2(s) \}. \end{aligned} \tag{3}$$

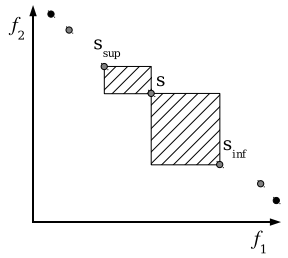

Fig. 1. Representation of the normalized objective space. The optimistic hypervolume improvement (OHVI) of a solution s is the sum of the two areas between s and its closest neighbors in the objective space, s_{inf} and s_{sup}.

Either s_{sup} or s_{inf} may not exist if s is the best solutions for $f_2()$ or $f_1()$, respectively. In such a case, we define the OHVI to be two times the existing ohvc, in order to avoid a strong bias against extreme solutions. Figure 1 graphically illustrates the computation of the OHVI of a solution.

When using OHVI for selection, the unexplored solution with the maximum OHVI value is chosen. It is, however, important to note that the above definition of OHVI makes some implicit assumptions. First, it assumes that by exploring the neighborhood of a solution new solutions are obtained only between the selected solution and the closest solutions in the objective space. This may not be necessarily the case and if the correlation between the distance of solutions in the solution space and the distance in the objective space is small, OHVI may not work better than random selection. Second, the above definition does not consider the area that dominates the current solution, and, hence, it assumes that no solutions are found that dominate the current solution. Again, this is not necessarily true, but given a good enough initial set, it is more common to find nondominated solutions than dominating ones.

Although the proposed OHVI selection is specifically designed for PLS, the concept of ohvc was previously proposed for improving the anytime behavior of Two-Phase Local Search [5]. Moreover, ohvc is related, but different to the hypervolume contribution used in some multi-objective evolutionary algorithms (MOEAs), e.g. in [1] where the hypervolume contribution of each solution with respect to the current archive is used to select or discard solutions. By contrast, ohvc estimates the potential contribution of solutions that *could* be found by the algorithm in the next iteration. The proposed OHVI selection is related to diversity measures used in other MOEAs, such as crowding distance [3] and the distance to the k-nearest neighbor [16], since it favors the exploration of the less crowded regions of the objective space.

Acceptance Criterion. The original PLS accepts any new nondominated solution into its archive. This *nondominating acceptance criterion* favors exploration but it also leads to a fast archive growth, which in turn slows down the algorithm and makes it more difficult to select the best solutions for further exploration. By contrast, a *dominating acceptance criterion*, that is, accepting only neighbors that dominate the currently explored solution, may allow PLS to reach faster the

optimal Pareto front. There are two reasons for this. First, the size of the archive stays more limited; second, the search becomes more focused on moving the current archive closer to the optimal Pareto front. The downside of a dominating acceptance criterion is that exploration is limited: nondominated solutions that may fill gaps between other solutions are not accepted and possible paths to other dominating or nondominated solutions are cut.

To avoid poor quality solutions caused by early termination of PLS when using the dominating acceptance criterion, we propose to change between the two acceptance criteria in the following manner. Initially, the dominating acceptance criterion is used as described above. If no dominating neighbor is found, the neighborhood of the current solution is explored again, this time accepting nondominated solutions. In this way, a switch from the dominating to the nondominated acceptance criterion happens on an adaptive, per-solution basis.

Neighborhood Exploration. The neighborhood exploration component decides when to stop exploring the neighborhood of the current solution. In the original PLS, the neighborhood of a solution is always fully explored. Alternatively, the exploration of the neighborhood may be stopped as soon as an "improved" solution is found, where the meaning of "improved" is defined by the acceptance criterion (see above) as being either a dominating or a nondominated solution. Different levels of neighborhood exploration may be defined by allowing a partial exploration of the neighborhood beyond finding a first acceptable solution. Here, we focus on the two extreme cases of either a *full* exploration (as in the original PLS) or a *first-accepted* exploration. By first-accepted exploration one cannot find as many solutions around the current one as with full exploration, but it terminates earlier and, thus, may allow to move close to the Pareto front faster. Independent of which of the two rules is applied, in PLS a solution becomes marked as explored, as soon as its exploration is stopped.

When using first-accepted exploration, one may find improving solutions by completing the neighborhood exploration of solutions in the archive, even if they are marked as explored. To account for this possibility, we propose the following combination of the two rules: When all solutions in the archive have been explored using the first-acceptance rule, the whole archive is marked as unexplored and the algorithm switches to full neighborhood exploration.

Related Work. To the best of our knowledge, there has been no study of the anytime behavior of PLS subject to some algorithmic variations. Liefooghe et al. [8] study the effect on the quality of restricting the overall exploration by limiting the number of solutions to be selected for exploration, and by limiting the neighborhood exploration itself. Nonetheless, there are significant differences with our proposals here. First, they do not distinguish between acceptance criterion and neighborhood exploration. In particular, they do not examine the combination of the dominating acceptance criterion and full exploration. Second, when using the dominating acceptance criterion and the first-accepted neighborhood exploration, they propose to also add all nondominated solutions found during exploration to the current archive. Our dominating acceptance criterion is more aggressive, keeps a smaller archive size, and focuses the search on getting

closer to the Pareto front. Third, different from us they do not consider variants proposed here that switch strategies during a single run of PLS, such as from dominating to nondominated acceptance criterion.

3 Experimental Analysis

3.1 Experimental Setup and Performance Assessment

We test the proposed PLS variants on the bTSP. We generate 10 bTSP instances of size 200, 300, and 500. The two distance matrices of each instance are generated independently of each other and correspond to symmetric, Euclidean TSP instances [5]. Due to the stochasticity of the algorithms, we repeat each experiment 10 times using a different random seed leading to different initial sets for PLS. The algorithms are 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 a 6 MB cache under Cluster Rocks Linux version 4.2.1/CentOS 4. We use the hypervolume unary indicator as a measure of the quality of the Pareto front approximations. As the upper bounds used for normalization of the objective values, we sample 10 000 random solutions, and we record the worst objective value produced from this sampling. Note that we also check that the upper bounds are never attained in any result we obtained. The lower bounds are the optimal solutions for each distance matrix obtained from the exact Concorde solver, release 03.12.19. In order to study the anytime behavior of the algorithms, we store the approximation of the Pareto front at 100 steps during the run of each algorithm, where each step is the CPU time in seconds at the following points in time: $t_i = \exp(i \cdot \ln(1001)/100) - 1, \quad i \in 1, \ldots, 100$. We examine the anytime behavior of each variant by plotting the hypervolume of its archive at each time step as follows. First, we normalize all the objectives values to the range $[1, 2]$, using the bounds as described above. Next, we compute the hypervolume of each Pareto front approximation using the reference point $[2.1, 2.1]$. Finally, for each strategy, each instance and each time step, we plot the hypervolume averaged over the 10 independent runs of each variant and the 95% confidence intervals around the mean as a gray area.

PLS starts from a given set of solutions, and this set has a large impact on PLS' final results. We consider three common alternatives for this initial set:

1. One random solution, corresponds to use PLS as a stand-alone algorithm.
2. Two solutions, each of high quality for one objective. This corresponds to a scenario where high-performing single-objective algorithms are available for each objective. Here, each solution is generated by running the iterated local search (ILS) algorithm using a 3-opt neighborhood for 2 seconds.
3. A nondominated set of high-quality solutions. This setting corresponds to a scenario where we have an algorithm for generating a high-quality approximation to the Pareto front and PLS further improves this approximation. Here, we generate a set of 5 high-quality solutions by running the anytime two-phase local search (TPLS) algorithm [5]. Each weighted sum aggregation of the objective functions is tackled by the ILS algorithm during 2 seconds.

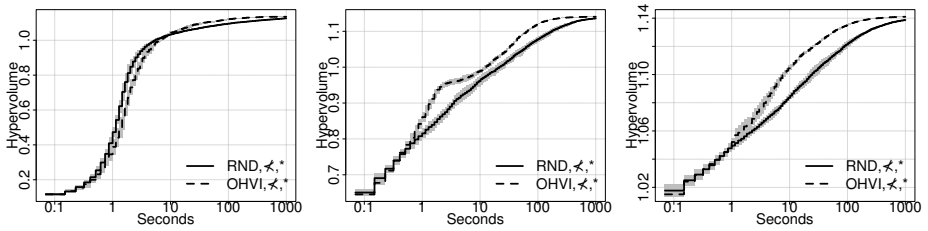

Fig. 2. PLS⟨RND⟩ vs PLS⟨OHVI⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right), respectively. Note the different ranges on the y-axis.

3.2 Experimental Results

First, we graphically explore the impact of one component at a time. Later, we compare the best combinations on larger instances. For reasons of space, we only provide results on one instance for each comparison; other instances show very similar behavior, and their plots are available as supplementary material [6]. As a last step, we carry out a statistical analysis over all instances.

Selection Step. We call the variant of PLS using a random selection PLS⟨RND⟩, and the variant using the OHVI for selection PLS⟨OHVI⟩. We compare in Fig. 2 these two variants. When starting from a random solution (*left*) the two variants show a very similar evolution; no curve clearly dominates the other. When the initial set is either two (*middle*) or a set of high-quality solutions (*right*), the curve corresponding to PLS⟨OHVI⟩ is most of the time above the one corresponding to PLS⟨RND⟩, which shows that PLS⟨OHVI⟩ generates a significantly better approximation set during most of the run time. Therefore, we choose OHVI as the selection step in the remainder of the paper.

Acceptance Criterion. We call the nondominated acceptance criterion PLS⟨$\nprec$⟩, the dominating one PLS⟨$\succ$⟩, and the combination of both PLS⟨$\succ\nprec$⟩. All variants use OHVI as the selection step. We compare these three variants in Fig. 3. When the initial set is a random solution (*left*), PLS⟨$\succ$⟩ improves the quality of the archive in a very short time, after which it stops because the whole archive has been explored. On the other hand, PLS⟨$\nprec$⟩ improves its archive at a slower rate, but it does not stop prematurely and it yields much better final results. Interestingly, the combination of the two strategies for the acceptance criterion used by PLS⟨$\succ\nprec$⟩ outperforms both individual strategies clearly: it combines the fast initial improvement of PLS⟨$\succ$⟩ with the much better final performance of PLS⟨$\nprec$⟩. When starting from two (*middle*) or a set of high-quality solutions (*right*), the behavior of PLS⟨$\succ$⟩ is particularly bad: it is not able to improve the initial set at all, which results in the flat line at the bottom of the plots. On the other hand, PLS⟨$\nprec$⟩ is able to significantly improve the initial solutions. The fact that the anytime behavior of PLS⟨$\succ\nprec$⟩ is better than the one of PLS⟨$\nprec$⟩ shows that the adaptive switching of the acceptance criteria on a per solution basis in PLS⟨$\succ\nprec$⟩ is highly beneficial.

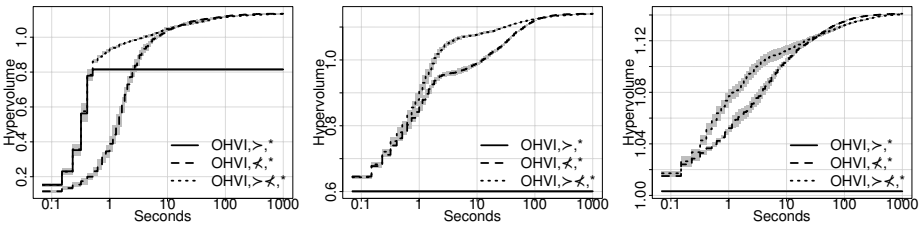

Fig. 3. PLS⟨OHVI, ≻⟩ vs PLS⟨OHVI, ≺⟩ vs PLS⟨OHVI, ≻≺⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right), respectively. Note the different ranges on the y-axis.

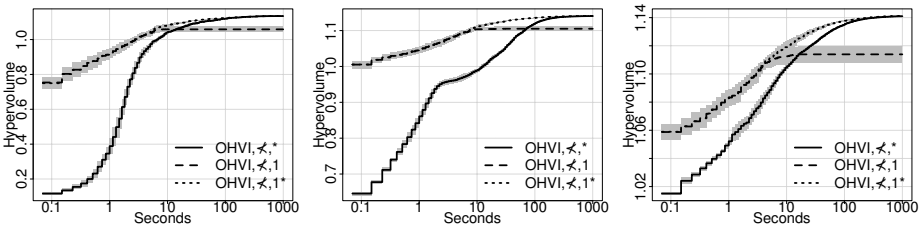

Fig. 4. PLS⟨∗⟩ vs PLS⟨1⟩ vs PLS⟨1∗⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right). Note the different ranges on the y-axis.

Neighborhood Exploration. In the following, PLS⟨∗⟩ denotes the full neighborhood exploration in PLS, PLS⟨1⟩ the first-accepted neighborhood exploration, and PLS⟨1∗⟩ the PLS variant that switches from the first-accepted to the full neighborhood exploration. All variants use OHVI selection and nondominating acceptance criterion. Fig. 4 shows that the anytime behavior of these three variants is very consistent, independently of the kind of initial set. PLS⟨1⟩ improves quickly the quality of the current archive in the first ten seconds, but then it terminates. PLS⟨∗⟩ requires more than ten seconds to reach the same quality as PLS⟨1⟩, but then improves further until the computation time limit. Finally, PLS⟨1∗⟩ initially matches the behavior of PLS⟨1⟩ (being actually the same algorithm) but then continues to progress due to the switch in the rule of the neighborhood exploration. Hence, PLS⟨1∗⟩ has a much better anytime behavior than each individual strategy for exploring the neighborhood.

Interactions between PLS Components. We now compare PLS variants that differ in more than one component in order to explore interactions between components. PLS⟨RND, ≺, ∗⟩ denotes the classical PLS using random selection, nondominated acceptance criterion and full neighborhood exploration; PLS⟨OHVI, ≻≺, ∗⟩ denotes the best variant obtained above when analyzing the acceptance criterion; PLS⟨OHVI, ≺, 1∗⟩ the best variant obtained above when analyzing the neighborhood exploration rule; and PLS⟨OHVI, ≻≺, 1∗⟩ the variant that combines the PLS⟨≻≺⟩ and PLS⟨1∗⟩ strategies. These four PLS variants are compared in Fig. 5. The results obtained by PLS⟨OHVI, ≻≺, ∗⟩ and PLS⟨OHVI, ≺, 1∗⟩ confirm that the newly proposed algorithms improve the anytime behavior of PLS w.r.t. to the

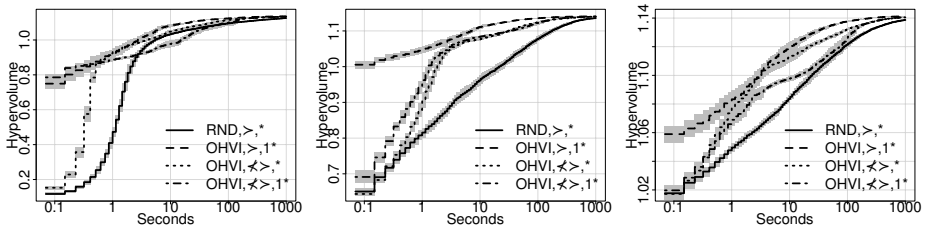

Fig. 5. PLS⟨RND, $\not\prec$, ∗⟩ vs PLS⟨OHVI, $\not\prec$, 1∗⟩ vs PLS⟨OHVI, $\succ\not\prec$, ∗⟩ and PLS⟨OHVI, $\succ\not\prec$, 1∗⟩, starting from a random seed (left), 2 solutions (middle), a set of solutions (right)

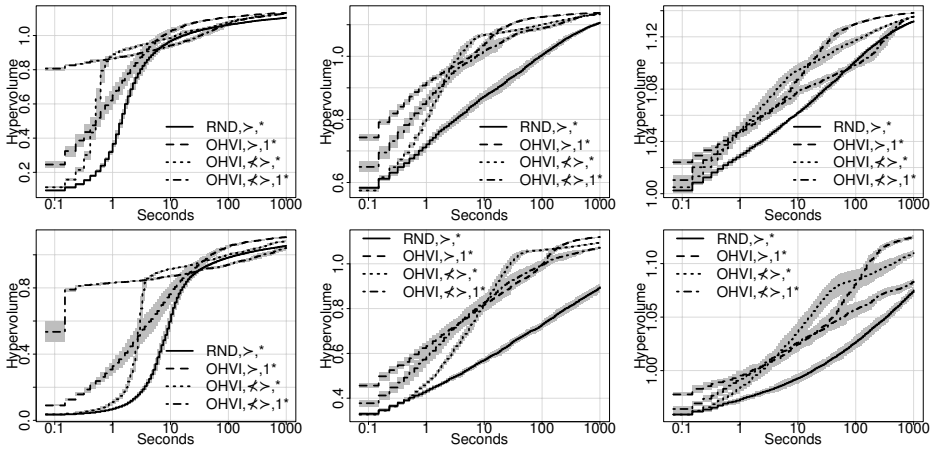

Fig. 6. PLS⟨RND, $\not\prec$, ∗⟩ vs PLS⟨OHVI, $\not\prec$, 1∗⟩ vs PLS⟨OHVI, $\succ\not\prec$, ∗⟩ and PLS⟨OHVI, $\succ\not\prec$, 1∗⟩, starting from a random seed (left), 2 solutions (middle), a set of solutions (right), and for two instance sizes: 300 (top) and 500 cities (bottom)

classical PLS⟨RND, $\not\prec$, ∗⟩. However, there is a somewhat surprising interaction between the components of PLS⟨OHVI, $\succ\not\prec$, 1∗⟩, since its anytime behavior is worse than that of PLS⟨OHVI, $\not\prec$, 1∗⟩ in general, and also worse than PLS⟨OHVI, $\succ\not\prec$, ∗⟩ for a high-quality initial set (right plot). Further work would be necessary to completely understand this behavior.

We also compare these four variants on larger bTSP instances, 10 instances of sizes 300 and 500, respectively. For tackling these larger instances, we use *candidate lists* obtained by Pareto-ranking of the edges to speed-up the neighborhood exploration [10]. We tested candidate lists of size 25, 50 and 100; they resulted in similar trends, and, here we only present results with size 50. The results provided by Fig. 6 do not allow us to declare an overall winner. When the initial solution is random (left plots), PLS⟨OHVI, $\succ\not\prec$, 1∗⟩ progresses much faster than other variants, but it is eventually outperformed by other variants. When the initial set are two or more high-quality solutions, PLS⟨OHVI, $\not\prec$, 1∗⟩ obtains the best hypervolume in the earlier and final stages. However, there is a range

Table 1. Statistical analysis of the best variants of PLS at different time steps. We chose 10.2 and 101.4 as they are the closest time steps from 10 and 100 seconds, respectively. For each time, PLS variants are ordered according to the sum of ranks obtained across all instances. The numbers in parenthesis are the differences of the sum of ranks relative to the best variant. PLS variants that are statistically significantly better than the best one are indicated in bold face. ΔR_α gives the difference of the sum of ranks that is significant.

Time	ΔR_α	**Strategies** (ΔR)
		Instance size 200, initial set: random solution.
1.0	12.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (34), ⟨OHVI,≻≯,1*⟩ (161), ⟨RND,≯,*⟩ (265)
10.2	13.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (116), ⟨RND,≯,*⟩ (192), ⟨OHVI,≻≯,1*⟩ (284)
101.4	13.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (145), ⟨OHVI,≻≯,1*⟩ (158), ⟨RND,≯,*⟩ (293)
1000.0	19.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (81.5), ⟨OHVI,≻≯,*⟩ (97), ⟨RND,≯,*⟩ (259.5)
		Instance size 200, initial set: two high-quality solutions.
1.0	3.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (103), ⟨OHVI,≻≯,*⟩ (197), ⟨RND,≯,*⟩ (300)
10.2	11.6	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (139), ⟨OHVI,≻≯,*⟩ (158), ⟨RND,≯,*⟩ (299)
101.4	10.6	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (131), ⟨OHVI,≻≯,*⟩ (169), ⟨RND,≯,*⟩ (300)
1000.0	13.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (115.5), ⟨OHVI,≻≯,*⟩ (157.5), ⟨RND,≯,*⟩ (291)
		Instance size 200, initial set: high-quality set.
1.0	17.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (117), ⟨OHVI,≻≯,1*⟩ (138), ⟨RND,≯,*⟩ (277)
10.2	10.7	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (76), ⟨OHVI,≻≯,1*⟩ (194), ⟨RND,≯,*⟩ (278)
101.4	15.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (130), ⟨OHVI,≻≯,1*⟩ (191), ⟨RND,≯,*⟩ (279)
1000.0	16.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (111.5), ⟨OHVI,≻≯,*⟩ (125.5), ⟨RND,≯,*⟩ (279)
		Instance size 300, initial set: random solution.
1.0	7.1	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (78), ⟨OHVI,≯,1*⟩ (189), ⟨RND,≯,*⟩ (289)
10.2	14.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (84), ⟨RND,≯,*⟩ (171), ⟨OHVI,≻≯,1*⟩ (277)
101.4	16.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (122), ⟨OHVI,≻≯,1*⟩ (220), ⟨RND,≯,*⟩ (254)
1000.0	12.7	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (142), ⟨OHVI,≻≯,*⟩ (146), ⟨RND,≯,*⟩ (296)
		Instance size 300, initial set: two high-quality solutions.
1.0	12.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (102), ⟨OHVI,≻≯,*⟩ (169), ⟨RND,≯,*⟩ (289)
10.2	11.5	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (137), ⟨OHVI,≻≯,1*⟩ (160), ⟨RND,≯,*⟩ (299)
101.4	3.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (102), ⟨OHVI,≻≯,1*⟩ (198), ⟨RND,≯,*⟩ (300)
1000.0	10.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (129.5), ⟨OHVI,≻≯,*⟩ (170.5), ⟨RND,≯,*⟩ (300)
		Instance size 300, initial set: high-quality set.
1.0	22.3	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (35), ⟨OHVI,≻≯,1*⟩ (37), ⟨RND,≯,*⟩ (224)
10.2	13.4	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (101), ⟨OHVI,≻≯,1*⟩ (175), ⟨RND,≯,*⟩ (284)
101.4	9.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨RND,≯,*⟩ (225), ⟨OHVI,≻≯,1*⟩ (275)
1000.0	10.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (135.5), ⟨OHVI,≻≯,1*⟩ (164.5), ⟨RND,≯,*⟩ (300)
		Instance size 500, initial set: random solution.
1.0	0	⟨**OHVI,≻≯,1***⟩, ⟨OHVI,≯,1*⟩ (100), ⟨OHVI,≻≯,*⟩ (200), ⟨RND,≯,*⟩ (300)
10.2	12	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (106), ⟨OHVI,≯,1*⟩ (193), ⟨RND,≯,*⟩ (285)
101.4	6.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (103), ⟨RND,≯,*⟩ (196), ⟨OHVI,≻≯,1*⟩ (297)
1000.0	16	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (120), ⟨OHVI,≻≯,1*⟩ (223), ⟨RND,≯,*⟩ (257)
		Instance size 500, initial set: two high-quality solutions.
1.0	9.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (71), ⟨OHVI,≻≯,*⟩ (186), ⟨RND,≯,*⟩ (283)
10.2	22.4	⟨**OHVI,≯,1***⟩, ⟨**OHVI,≻≯,***⟩ **(18)**, ⟨OHVI,≻≯,1*⟩ (46), ⟨RND,≯,*⟩ (220)
101.4	13.8	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (101), ⟨OHVI,≻≯,1*⟩ (157), ⟨RND,≯,*⟩ (286)
1000.0	0	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨OHVI,≻≯,1*⟩ (200), ⟨RND,≯,*⟩ (300)
		Instance size 500, initial set: high-quality set.
1.0	14.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (65), ⟨OHVI,≻≯,*⟩ (154), ⟨RND,≯,*⟩ (273)
10.2	19.4	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (63), ⟨OHVI,≯,1*⟩ (105), ⟨RND,≯,*⟩ (256)
101.4	11.7	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (59), ⟨OHVI,≻≯,1*⟩ (169), ⟨RND,≯,*⟩ (276)
1000.0	8.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨OHVI,≻≯,1*⟩ (216), ⟨RND,≯,*⟩ (284)

of time where other variants, in particular PLS$\langle$OHVI, $\succ\not\prec, *\rangle$, perform better. Nonetheless, the plots clearly show that the proposed PLS variants consistently outperform the classical PLS$\langle$RND, $\not\prec, *\rangle$.

Statistical Tests. We perform statistical tests to assess the behavior over the whole set of 10 instances of each size. We apply the Friedman test for analyzing non-parametric, unreplicated, complete block designs. Each block is a run using a particular seed (out of ten) on a single instance, and the different PLS variants are the treatment factors. Next, we rank the PLS variants per block according to the hypervolume, the lower the rank the better, and we calculate the difference (ΔR) between the sum of ranks of each variant and the best ranked one (with the lowest sum of ranks). Finally, we calculate the minimum difference between the sum of ranks of two variants that is statistically significant (ΔR_α), given a significance level of $\alpha = 0.05$, using the Friedman post-test for multiple comparisons [2]. We perform this test on the output of the algorithms at various time steps, for each instance size and for each type of initial set. Table 1 summarizes the result of one independent test in each row. We indicate in bold face the best variant (the one having the lowest sum of ranks) and those that are not significantly different from the best one. The tests confirm our conclusions that PLS$\langle$OHVI, $\not\prec, 1*\rangle$ is the best strategy overall for size 200. It is also often the best for sizes 300 and 500, but not always. Moreover, it shows that the classical PLS$\langle$RND, $\not\prec, *\rangle$ performs quite poorly when compared with the anytime PLS variants proposed here, being in some cases completely outranked (with a sum of ranks close to 300) by the other variants.

4 Conclusions

In this paper, we proposed alternative choices for algorithmic components of PLS that improve substantially its anytime behavior. In addition, we have proposed novel approaches that are based on switching strategies for the neighborhood exploration and the acceptance criterion; these switching strategies have proven to be essential for improving the anytime behavior. As a result, replacing the original PLS with one of our proposed variants in hybrid algorithms is likely to further improve the current state of the art in multi-objective optimization. However, none of the variants is clearly superior to all others, and further research is needed to understand in more detail the behavior of the proposed variants. In the future, we will extend the analysis to problems with more than 2 objectives, to other problems such as MKP [11] and permutation flowshop [4], and investigate other possibilities to improve anytime behavior.

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 and Manuel López-Ibáñez acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Associate and a Postdoctoral researcher, respectively.

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
2. Conover, W.J.: Practical Nonparametric Statistics. John Wiley & Sons (1999)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 181–197 (2002)
4. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38(8), 1219–1236 (2011)
5. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Improving the anytime behavior of two-phase local search. Annals of Mathematics and Artificial Intelligence 61(2), 125–154 (2011)
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Supplementary Material: Pareto Local Search Variants for Anytime Bi-Objective Optimization (2012), `http://iridia.ulb.ac.be/supp/IridiaSupp2012-004`
7. Ehrgott, M., Gandibleux, X.: Approximative solution methods for combinatorial multicriteria optimization. TOP 12(1), 1–88 (2004)
8. Liefooghe, A., Mesmoudi, S., Humeau, J., Jourdan, L., Talbi, E.G.: On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. Journal of Heuristics (2011)
9. Loudni, S., Boizumault, P.: Combining VNS with constraint programming for solving anytime optimization problems. European Journal of Operational Research 191, 705–735 (2008)
10. Lust, T., Jaszkiewicz, A.: Speed-up techniques for solving large-scale biobjective TSP. Computers & Operations Research 37(3), 521–533 (2010)
11. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. Arxiv preprint arXiv:1007.4063 (2010)
12. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 16(3), 475–510 (2010)
13. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation. LNEMS, vol. 535, pp. 177–200. Springer (2004)
14. Paquete, L., Stützle, T.: Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Computers & Operations Research 36(9), 2619–2631 (2009)
15. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control, pp. 95–100. CIMNE, Barcelona (2002)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)