Anytime Pareto Local Search

Jérémie Dubois–Lacoste*, Manuel López–Ibáñez, Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, 50 Av. F. Roosevelt, 1050 Brussels, Belgium

Abstract

Pareto Local Search (PLS) is a simple and effective local search method for tackling multi-objective combinatorial optimization problems. It is also a crucial component of many state-of-the-art algorithms for such problems. However, PLS may be not very effective when terminated before completion. In other words, PLS has poor *anytime behavior*. In this paper, we study the effect that various PLS algorithmic components have on its anytime behavior. We show that the anytime behavior of PLS can be greatly improved by using alternative algorithmic components. We also propose *Dynagrid*, a dynamic discretization of the objective space that helps PLS to converge faster to a good approximation of the Pareto front and continue to improve it if more time is available. We perform a detailed empirical evaluation of the new proposals on the bi-objective traveling salesman problem and the bi-objective quadratic assignment problem. Our results demonstrate that the new PLS variants not only have significantly better anytime behavior than the original PLS, but also may obtain better results for longer computation time or upon completion.

Keywords: Pareto local search, anytime optimization, multi-objective optimization, traveling salesman problem, quadratic assignment problem

1. Introduction

Optimization problems are ubiquitous in our society and of crucial importance in numerous fields of high social, environmental or economic relevance. In particular, many relevant situations involve problems that are evaluated according to several criteria. These criteria are often conflicting and there is no solution that is best for all of them at the same time. In this paper, we consider problems where no information is given about the decision maker's preferences. In this case, it is common to evaluate solutions by means of Pareto dominance. The goal is to find the set of Pareto optimal solutions (those not dominated by any other feasible solution), or at least a close approximation to it. The decision maker can then choose *a posteriori* the preferred one.

We focus on algorithms to tackle multi-objective combinatorial optimization problems (MCOPs), many of which are problems computationally intractable or, more formally, $\mathcal{NP}$-hard [14]. Pareto Local Search (PLS) [41] is a heuristic algorithm for tackling $\mathcal{NP}$-hard MCOPs in the Pareto sense. It is an extension of iterative improvement algorithms for single-objective problems [19] to the multi-objective case with the goal to approximate the Pareto-optimal set. PLS can produce, upon completion, high-quality approximations when used as a stand-alone algorithm [38, 41]. PLS is also a crucial component of hybrid algorithms, where PLS is used to improve upon a set of good solutions that have been obtained by approaches, for example, based on scalarizations. Examples of such hybrid algorithms can be found for the

*Corresponding author.

Email addresses: jeremie.dubois-lacoste@ulb.ac.be (Jérémie Dubois–Lacoste), manuel.lopez-ibanez@ulb.ac.be (Manuel López–Ibáñez), stuetzle@ulb.ac.be (Thomas Stützle)

multi-objective traveling salesman problem [33, 34] with two and three objectives, various bi-objective permutation flowshop problems [7], and the bi-objective multi-dimensional knapsack problem [35].

A disadvantage of PLS and, by extension, a limitation of these hybrid algorithms, is that it requires a possibly long time to reach high-quality approximations to the Pareto front (e.g., several hours for large biobjective traveling salesman problem (bTSP) instances tackled in the literature [34]). Moreover, if stopped early, PLS may produce a poor approximation when compared to the final approximation reached upon completion. For this reason, we say that PLS has a poor *anytime behavior*. By contrast, an algorithm with good anytime behavior aims to deliver the best possible result at any time during its execution by quickly reaching high-quality solutions and by continuously improving them [29, 45].

Our goal in this paper is to study and improve the anytime behavior of PLS. In a preliminary work [8], we empirically examined the impact that algorithmic components of PLS have on its anytime behavior, and proposed alternative variants that improve its anytime behavior for the bTSP. In this paper, we extend that preliminary work in four ways. First, we repeat the experiments reported in our preliminary study but considering longer computation time limits, which allows us to study the potential final quality reached by standard PLS versus the anytime PLS variants. Second, our preliminary work used only the bTSP as a benchmark problem. In this paper, we also consider the bi-objective quadratic assignment problem (bQAP). The bTSP and the bQAP are both fundamental problems that arise in a number of real-world situations. They are also among the most studied MCOPs in meta-heuristics research [4, 5, 12, 23, 30, 33, 40]. Third, we propose a dynamic grid mechanism (*Dynagrid*) that dynamically limits the number of unexplored solutions accepted by PLS. Dynagrid extends a previous static epsilon-grid mechanism [1] by discretizing the objective space using a decrease of epsilon at run-time. This allows PLS to converge faster to a well-spread Pareto-front approximation, while avoiding premature convergence if the initial discretization was excessively coarse. Dynagrid improves both the anytime behavior and the final quality achieved by PLS. Finally, we dedicate Section 7 to a survey of algorithmic components from the literature that are related to those proposed by us in this paper, and we discuss in detail the similarities and differences between the various proposals.

Our experimental analysis examines scenarios when PLS is used as a stand-alone algorithm and when PLS is part of a hybrid algorithm. Therefore, we are certain that the proposed anytime variants of PLS further improve the state-of-the-art local search algorithm for the bTSP [34], not only in terms of anytime behavior but also on final quality after completion. Moreover, our proposals can be applied to any bi-objective combinatorial problem and they may also turn out to be useful for other algorithms than PLS.

This paper is organized as follows. In Section 2, we first define multi-objective optimization and related concepts, and then we present the original PLS algorithm and its decomposition into distinct algorithmic components. In Section 3, we discuss the poor anytime behavior of the original PLS algorithm. Section 4 recalls the alternative algorithmic components proposed in a preliminary work [8] and presents new results that further confirm the improvements in anytime behavior reported earlier. Section 5 proposes Dynagrid, a dynamic discretization of the objective space that further improves the anytime behavior of PLS. In Section 6, we compare the original PLS with the best PLS variants obtained from the alternative algorithmic components and the objective space discretization. We deliberately chose to review the literature that is relevant to our study at the end of the paper (Section 7), since this allows us to discuss in more detail the similarities and differences between our proposals and previous ones. Finally, we conclude in Section 8 and highlight promising directions for future research.

2. Pareto Local Search

In this section, we first introduce basic concepts of multi-objective optimization. Next, we explain the original PLS algorithm and how we decompose it into different algorithmic components.

2.1. Multi-Objective Optimization

Many real-world problems require optimizing more than one objective. Often, these objectives are conflicting and, therefore, there exists no solution that is optimal for all objectives at the same time. If no a priori assumptions upon the decision maker's preferences can be made, the goal typically becomes to find a set of solutions that are optimal in the sense of Pareto optimality.

More formally, candidate solutions are evaluated according to an objective function vector $\vec{f} = (f_1, \ldots, f_d)$ with d objectives. Pareto dominance defines a partial order on the set of feasible solutions. Without loss of generality, we consider here only problems where all objectives are to be minimized. If $\vec{u}$ and $\vec{v}$ are vectors in $\mathbb{R}^d$, $\vec{u}$ is said to *dominate* $\vec{v}$ ($\vec{u} \prec \vec{v}$) iff $\vec{u} \neq \vec{v}$ and $u_i \leq v_i,\ i = 1, \ldots, d$. $\vec{u}$ and $\vec{v}$ are said to be mutually *non-dominated* iff $\vec{u} \nprec \vec{v}$, $\vec{v} \nprec \vec{u}$ and $\vec{u} \neq \vec{v}$. For simplicity, we extend the notion of dominance to solutions and we say that a solution s dominates another one s' iff $\vec{f}(s) \prec \vec{f}(s')$. If a set $\mathcal{A}$ does not contain any solution that dominates a solution s, and does not contain any solution s' with $\vec{f}(s') = \vec{f}(s)$, we write $\mathcal{A} \nprec s$.

If no s' exists such that $\vec{f}(s') \prec \vec{f}(s)$, s is called *Pareto optimal*. The goal when tackling MCOPs in the Pareto sense is to find the set of Pareto-optimal solutions. Their image in the objective space is called the Pareto front. Since this goal is in many cases computationally intractable, heuristic algorithms are used to find an as good as possible approximation to the Pareto front [11].

2.2. The Original Pareto Local Search Algorithm

PLS is a direct extension of the iterative improvement algorithm from single-objective problems to the multi-objective case. PLS explores the neighborhood of solutions in its archive, inserts new solutions in the archive that are non-dominated, and filters the archive to remove dominated solutions. PLS was originally proposed by Paquete et al. [41]; independently, a similar algorithm was proposed by Angel et al. [1].

Algorithm 1 illustrates the PLS framework. The input to PLS is an initial set of solutions $\mathcal{A}_0$ that are mutually non-dominated. The solutions in $\mathcal{A}_0$ are initially marked as unexplored (line 2). PLS updates an archive of nondominated solutions $\mathcal{A}$, which is initially equal to $\mathcal{A}_0$, by applying the following steps. First, a solution s is chosen randomly among all unexplored ones (*selection step* on line 5) and its neighborhood, $\mathcal{N}(s)$, is fully explored (line 6). All neighbors that are non-dominated w.r.t. the solutions in $\mathcal{A}$ (line 7) are considered for addition to the archive $\mathcal{A}$. The procedure Update (line 9) adds a candidate solution $s' \in \mathcal{N}(s)$ to the archive if s' is not dominated by any solution in the archive and removes all solutions from the archive that become dominated by s'. Once the neighborhood of s has been fully explored, s is marked as explored (line 12). When $\mathcal{A}$ contains only solutions that have been fully explored, the algorithm stops in a Pareto local optimum [42], that is, a set of nondominated solutions for which any neighbor solution is dominated or equal to a solution in the set.

2.3. Algorithmic Components of PLS

We decompose PLS into three main algorithmic components for which we will later describe and test alternative variants. We explain these components in detail below:

Selection step. This component determines how to select the next solution for neighborhood exploration. In the original PLS, this solution is selected uniformly at random among the unexplored ones.

Acceptance criterion. This component describes the conditions under which a new solution enters the archive. The original PLS accepts all solutions identified in the neighborhood exploration that are non-dominated.

Algorithm 1 Pareto Local Search

```
1:  **Input:** An initial set of non-dominated solutions $\mathcal{A}_0$
2:  $\mathsf{explored}(s) := \textsc{false} \quad \forall s \in \mathcal{A}_0$
3:  $\mathcal{A} := \mathcal{A}_0$
4:  **repeat**
5:      $s :=$ select randomly a solution from $\mathcal{A}_0$        // *Selection step*
6:      **for each** $s' \in \mathcal{N}(s)$ **do**        // *Neighborhood exploration*
7:          **if** $\mathcal{A} \nprec s'$ **then**        // *Acceptance criterion*
8:              $\mathsf{explored}(s') := \textsc{false}$
9:              $\mathcal{A} := \mathsf{Update}(\mathcal{A}, s')$
10:         **end if**
11:     **end for**
12:     $\mathsf{explored}(s) := \textsc{true}$
13:     $\mathcal{A}_0 := \{s \in \mathcal{A} \mid \mathsf{explored}(s) = \textsc{false}\}$
14: **until** $\mathcal{A}_0 = \emptyset$
15: **Output:** $\mathcal{A}$
```

Neighborhood exploration. This component performs the neighborhood exploration of the selected solution. In particular, it defines the part of the neighborhood that will be explored before switching to a different solution. The original PLS algorithm always explores the entire neighborhood of a solution.

In the next section, we focus on the algorithmic components of PLS, test possible variants of them and examine their impact on the anytime behavior of the resulting algorithms.

3. Anytime Behavior of Pareto Local Search

PLS has a natural stopping criterion, that is, it stops when all solutions in the archive have been fully explored. Previous works have shown that PLS can reach a high-quality approximation of the Pareto front in many combinatorial problems if it runs until completion [34, 36]. However, PLS may take a long time to stop. In the worst case, it may take an exponential time to reach completion, since for some problems the Pareto front may contain an exponential number of solutions and, in the original PLS, the archive size is unlimited. In practical problems, runtimes of hours for typical instance sizes have been reported [34]. Therefore, it is desirable in some situations to stop PLS before completion, but still return an as high-quality approximation of the Pareto front as possible. An *anytime algorithm* aims to obtain as high-quality results as possible at any time during its execution and continuously improves the quality of the result as computation time increases [29, 45]. Similarly, an algorithm that produces better results than another for any termination criterion of practical interest is said to have a better *anytime behavior*.

The original PLS does not make any attempt to quickly obtain a high-quality approximation of the Pareto front and, hence, if stopped early, the approximation of the Pareto front may be poor (see Fig. 1). Thus, we can say that PLS, as originally proposed, is not an anytime algorithm in the sense that it does not show a good anytime behavior.

As in previous work [6, 8], in this paper we use the hypervolume indicator [46] to compare the anytime behavior of multi-objective optimizers by graphically plotting the evolution of the hypervolume over time. The hypervolume computes the size of the objective space dominated by a given non-dominated set and, thus, higher hypervolume corresponds to better quality. In the case of PLS, we compute the hypervolume of the current archive $\mathcal{A}$ of non-dominated solutions. Figure 2 illustrates this analysis; it shows the development of the obtained hypervolume over time for two algorithms. The plain line shows

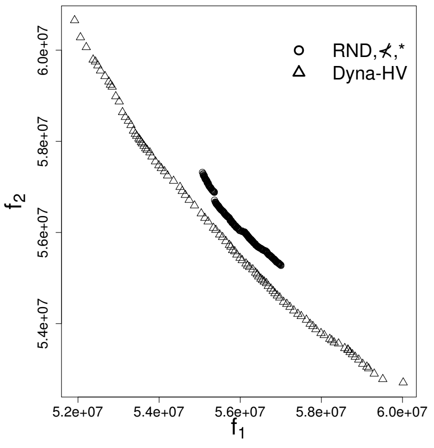

Figure 1: Two non-dominated sets obtained by two PLS variants, the original PLS (denoted by RND,$\not\prec$,$*$) and an anytime PLS variant proposed in this paper (denoted by Dyna-HV). Each set corresponds to the output of one single run, stopped after 20 seconds, on a bQAP instance with correlation -0.75. The set obtained by the anytime PLS variant completely dominates the set obtained by the original PLS.

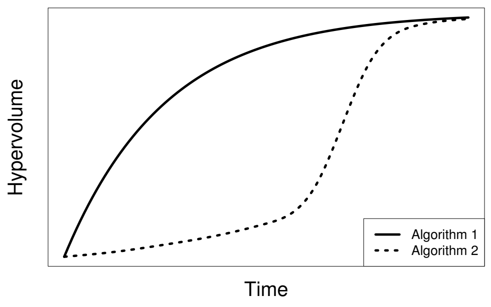

Figure 2: An example of the hypervolume attained by two algorithms over time. Algorithm 1 shows a good anytime behavior, while Algorithm 2 shows worse anytime behavior.

an algorithm with good anytime behavior; the algorithm produces a quick increase of the quality at the beginning, and then continuously improves until the end of the execution. The dotted line shows an algorithm with a poor anytime behavior; its quality improves slowly and it is worse than the quality of the other algorithm at any time.

In the following sections, we examine how the different components of PLS affect its anytime behavior. In addition, we propose variants of PLS that greatly improve its anytime behavior. These new variants allow stopping PLS in times shorter by several orders of magnitude than its natural completion time and obtain much better approximations to the Pareto front (Fig. 1 is a typical example).

4. Alternative Algorithmic Components for Anytime Optimization

We have recently shown [8] that the anytime behavior of PLS can be largely improved by using alternative variants for each of its algorithmic components. In particular, we proposed the optimistic hypervolume improvement (OHI) as an alternative to random selection; and alternative variants for the acceptance criterion and neighborhood exploration. In this section, we first recall these proposals in detail, since we will use these concepts throughout the paper. Next, we report new experimental results. In particular, we consider longer computation time limits than those considered in our preliminary study [8], which gives a more accurate notion of the potential final quality reached by standard PLS versus the new anytime PLS variants. Moreover, we include the bQAP as an additional benchmark problem.

Alternative for selection step. PLS does not make use of any information on the current state of the archive in the selection step but uses a simple random choice. An alternative approach is to select solutions whose exploration may have the largest potential to improve the current archive. Given two solutions s and s' (in the biobjective case), we define the *optimistic hypervolume contribution* (ohvc) as the potential contribution to the hypervolume of the archive by the local ideal point defined by s and s' in the objective space. In the bi-objective space, we can define it as:

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

The ohvc is based on the assumption that solutions that are close in the solution space are close in the objective space. This common assumption is known to often hold in the single objective case. Intuitively, if this is true for each objective of a multi-objective problem, it will be true for the multi-objective problem itself. This assumption in the multi-objective case is supported empirically by some studies [6, 15, 21, 37, 44], and to some extent by the results of this paper.

Hence, by exploring the neighborhood of a given solution, one can expect to find new non-dominated solutions in the region between the current solution and its closest neighbors in the objective space. The ohvc indicator can be used to select a pair of adjacent solutions. In the case of PLS, we define an additional indicator for the selection of a single solution s, called Optimistic Hypervolume Improvement (OHI), as follows:

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

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

$$s_{\text{sup}} = \arg\min_{s_i \in \mathcal{A}} \{f_2(s_i) \mid f_2(s_i) > f_2(s)\} \quad \text{and} \quad s_{\text{inf}} = \arg\max_{s_i \in \mathcal{A}} \{f_2(s_i) \mid f_2(s_i) < f_2(s)\}.$$

Either s_{sup} or s_{inf} may not exist if s is the best solution for f_1 or f_2, respectively (often called extreme solution). In such a case, we define the OHI to be two times the optimistic hypervolume contribution of the existing solution in order to avoid a strong bias against extreme solutions. Fig. 3 shows a graphical representation of the OHI indicator. The OHI value is a heuristic to measure the "gap" around solutions in the objective space, in order to select for exploration a solution that has the highest potential of increasing the hypervolume. This is different from measuring the actual hypervolume contribution of the solution [3, 24]. Computing the OHI of a set of solutions in the bi-objective space can be done in linear time, if the set is sorted with respect to one objective. For this reason, when adding or removing solutions, we always keep the set sorted. The OHI could be applied to more than two objectives, but an efficient implementation would require a data structure to find efficiently the nearest solutions to a given one. We leave this extension to future work.

As an alternative to the random selection in the original PLS, denoted by $\langle\mathsf{RND}\rangle$, the OHI selection, denoted by $\langle\mathsf{OHI}\rangle$, selects the unexplored solution with the highest OHI value.

Alternatives for acceptance criterion. The original PLS algorithm accepts any non-dominated solution for inclusion in the archive. We call this component $\langle\nprec\rangle$ for "non-dominated". However, different criteria have been used in the literature, particularly more restrictive ones that avoid a large

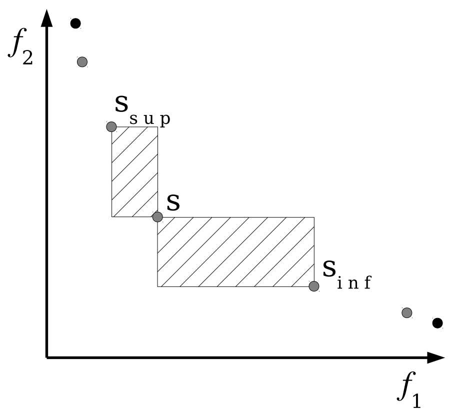

Figure 3: Representation of the normalized objective space. The OHI of solution s is the sum of the two hatched areas that lie between s and its two closest neighbors in the objective space, s_{inf} and s_{sup}. The OHI of extreme solutions (in black) is twice the area between them and their closest neighbor.

number of solutions in the archive [28] (a detailed discussion of how our proposals relate to the existing literature is given in Section 7). In particular, accepting only neighbors that dominate the current solution may allow a quick convergence to the Pareto front at the price of a possible loss of quality. We call this component $\langle \succ \rangle$ for "dominating". We can also switch from one rule to another: if a solution that dominates the current one is found, only such solution is accepted, and if no dominating solution can be found, the acceptance criterion switches to accepting solutions that are non-dominated. We call this component $\langle \succ \nprec \rangle$.

Alternatives for neighborhood exploration. PLS explores in its original version all neighboring solutions. This is also done in iterative improvement algorithms that use a *best-improvement* pivoting rule and, hence, we refer to this way of neighborhood exploration in PLS also as *best-improvement* and refer to it as component $\langle * \rangle$. In single-objective local search, a *first-improvement* neighborhood exploration often leads faster to local optima [19]. In the multi-objective case, *first-improvement* corresponds to stopping the neighborhood exploration as soon as one neighboring solution can be accepted. We refer to this alternative as $\langle 1 \rangle$. In PLS with *first-improvement* neighborhood exploration a solution is marked as explored when a neighbor is accepted into the archive even if there could be more acceptable neighbors if the full neighborhood were explored. We also proposed switching from one alternative to the other in the following way: when all solutions in the archive have been marked as explored using the *first-improvement* rule, the algorithm will mark all solutions as unexplored and explore them again using the *best-improvement* rule. Ideally, the first-improvement phase will quickly converge to a good approximation of the Pareto front, while the best-improvement phase will add any remaining neighbors to the archive and, thus, further improve its quality. We denote this alternative with $\langle 1* \rangle$.

In what follows, we denote each variant of the original PLS algorithm by specifying which alternative components it is using. For instance, PLS$\langle$OHI, $\succ \nprec$, $1*\rangle$ denotes the variant that

- uses OHI for the selection step;
- uses a rule for the acceptance criterion that switches from dominating to non-dominated;
- uses a rule for the neighborhood exploration that switches from stopping after the first accepted neighbor to stopping only after considering all accepted neighbors.

Following this notation, the original PLS is denoted by PLS$\langle$RND, $\nprec$, $*\rangle$.

4.1. Experimental Setup

4.1.1. Initial Sets to Start PLS

In our experimental analysis, we cover possible uses of PLS by using three different initial conditions, that is, sets of solutions that PLS starts from.

- **High-quality sets (HQS).** State-of-the-art algorithms for several problems [7, 10, 33, 34] use PLS in a second phase to refine a set of high-quality solutions obtained from a first phase, which often is based on scalarizations of the problem. To cover this usage of PLS, we consider an initial set composed of five high-quality solutions, that is, solutions very close to the Pareto-optimal front. These five solutions are also well-spread over the objective space, since they are obtained by an algorithm based on scalarizations that tends to distribute solutions as evenly as possible in terms of the hypervolume of the set [6]. We chose five in order to have one very good solution for each objective (on the extremes of the Pareto front), one solution that is an equal trade-off between the two extremes, and two solutions that represent unequal trade-offs between the two extremes. Since the initial solutions are nearly-optimal and well-spread, the task of PLS becomes to fill the gaps between these solutions. A higher number of initial solutions would make this task easier, but it would make harder to assess the anytime behavior since the difference in terms of hypervolume between the initial set and the final one would be small. Another drawback of starting from more solutions is that they take time to be generated (since a single-objective algorithm is launched to tackle each scalarization one by one), during which the anytime behavior would be rather poor: stopping during that phase would bring few solutions. In some preliminary experiments, we observed that the anytime behavior of different PLS variants was relatively similar, even if the absolute values are different (since the initial set has a higher hypervolume).

- **Two high-quality solutions (TS).** It is possible that there is no algorithm available to solve the scalarized problems, but an algorithm is available to solve the two individual objectives of the bi-objective problem. In this case, PLS starts from one nearly-optimal solution for each objective. These two initial solutions are on the extremes of the Pareto front, and, hence, the main task of PLS is to quickly reach the center of the Pareto front.

- **Random solution (RS).** Finally, it is possible that no algorithm is available to solve any single-objective version of the problem, i.e., one must rely only on PLS to tackle the problem. In this case, PLS starts from a random initial solution, which is likely to be far away from the Pareto front. Thus, the main task of PLS is to quickly reach the Pareto front, while keeping a wide spread set of solutions.

4.1.2. Performance Assessment of Non-Dominated Sets

Similarly to solutions that can be mutually non-dominated, different approximations to the Pareto front can be *incomparable*. In fact, this is often the case when comparing the output of different algorithms. It is therefore common to use indicators to measure the quality of these approximations. In this paper, we use the hypervolume (HV) indicator, a widely-used unary indicator that is consistent with the Pareto optimality principle [46]. In the bi-objective space, the hypervolume measures the area of the objective space that is weakly dominated by (i.e., dominated by or equal to) the image of the solutions in a non-dominated set. This area is bounded by a reference point that is worse in all objectives than all points in all non-dominated sets. The larger is this area, the better is a non-dominated set.

Computing the hypervolume of non-dominated sets requires to normalize the objective values of the solutions in them, to avoid favoring one objective over another if the objectives do not have the same range. For the bTSP, we found the lower bound for normalization using the exact Concorde solver [2], release 03.12.19, and the upper bound for normalization by taking the worst solution value of 100 000 solutions that are sampled uniformly at random. For the bQAP, we do not use the same procedure since no

Table 1: Cut-off times used for each problem and type of instances.

Problem	Instance type	Cut-off time (s)
bTSP	Size = 500	10 000
bQAP	Size = 100, Correlation = −0.75	1 200
	Size = 100, Correlation = −0.5	100

exact solver is available that solves our instances within reasonable computation time. Therefore, we ran the original PLS algorithm three times, using all initial conditions, and we record the best and the worst value obtained over all results. We then define the lower bound as being the *best value* multiplied by 0.95, and the upper bound as being the *worst value* multiplied by 1.05. We check during the experimental analysis that the bounds are never exceeded by any result we obtained. All objective values are then normalized into the range $[1, 2]$, and we use the coordinates $(2.1, 2.1)$ as the reference point for computing the hypervolume of the normalized sets.

We assess graphically the anytime behavior by plotting the average hypervolume (over 25 runs) of the normalized sets over computation time. To do so, we define a priori a sequence of time steps, at which we normalize the current non-dominated set obtained so far by the algorithm, we compute its hypervolume and we record it. The computation time required to perform these steps is not counted in the overall computation time that is measured and reported in our results, since these steps are required to observe the behavior of the algorithms but are not actually part of the algorithms. By using an exponential scale for the sequence of time steps (and for the plot) we can observe the behavior of the algorithms both after short periods of time and at larger scales. More precisely, we use 100 time steps, computed as $t_i = (\texttt{cutoff\_time} + 1)^{(i/100)} - 1, \quad i \in 1, \ldots, 100$, where `cutoff_time` is a cut-off time (see Table 1), determined from preliminary runs of the algorithms. In the case of the bTSP, we use a cut-off time ten times longer than the one used in our preliminary study [8]. We also show graphically the variance of each algorithm across the multiple runs by plotting in gray the confidence intervals corresponding to each curve.

4.1.3. Experimental Benchmark, Computational Environment and Neighborhood Operators

For the bTSP, we generated three instances with 500 cities. The two distance matrices of each instance are generated independently of each other and correspond to those of symmetric, Euclidean TSP instances [6, 40]. For the bQAP, we used the instance generator proposed in [23]. We generated three instances with correlations between the flow matrices in $\{-0.75, -0.5, 0, 0.5, 0.75\}$. The lower the correlation, the higher are the run times of PLS to reach completion, since the number of non-dominated solutions increases strongly with negative correlation [38]. On the other hand, PLS requires a very short time to terminate for instances with correlation zero or larger. Hence, these instances are not useful for illustrating improvements in anytime behavior, and we only present in this paper results with correlation −0.75 and −0.5.

The algorithms are implemented in `C++`, compiled with gcc 4.4.6, and the experiments were run on a single core of AMD Opteron 6272 CPUs, running at 2.1 Ghz with a 16 MB cache under Cluster Rocks Linux version 6/CentOS 6.3, 64bits.

For the bTSP, we use a neighborhood operator based on 2-exchange moves, and we use delta-evaluations to compute the objective value of new candidate solutions faster [19]. For the bQAP, we use a neighborhood operator based on exchanging two components [19]. The initial solutions, when starting from two or more high-quality solutions are obtained solving scalarized problems (see Section 4.1.1). For the bTSP, the single-objective algorithm used to tackle these problems is an Iterated Local Search algorithm based on 3-opt moves [19]. For the bQAP, we use a Simulated Annealing algorithm [20]. For

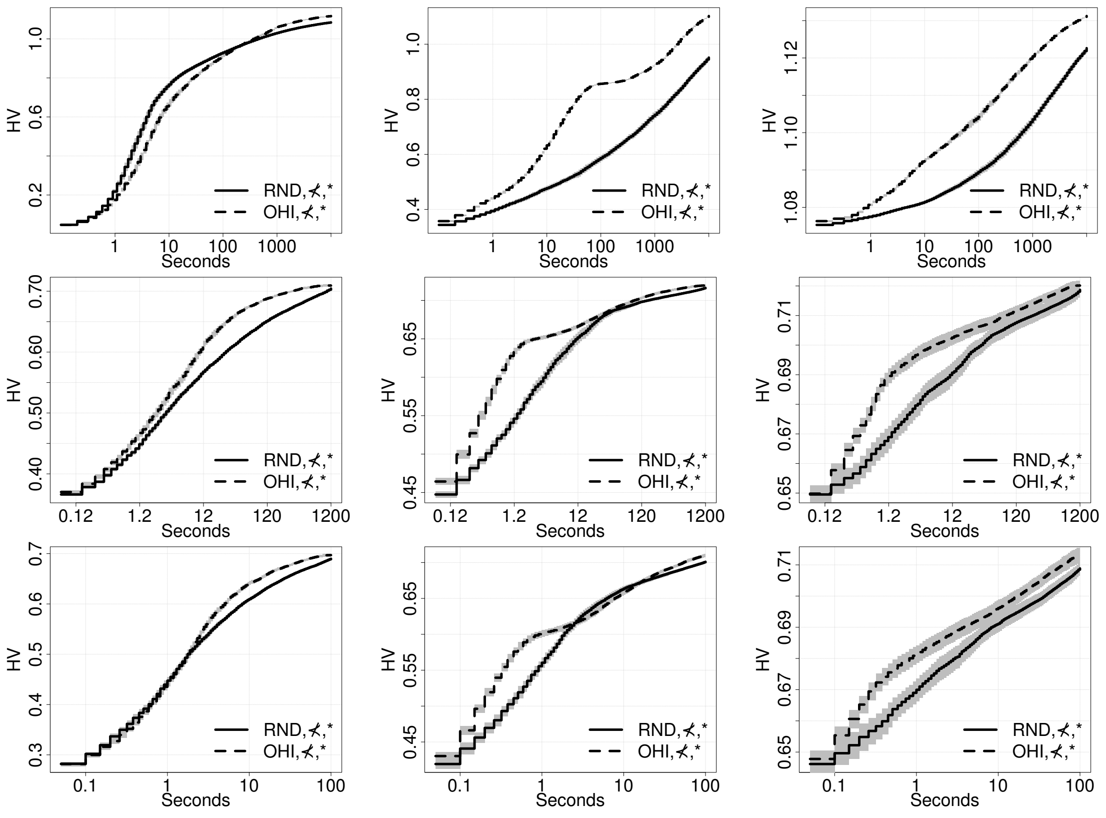

Figure 4: **Selection Step:** PLS⟨RND, $\not\prec$, $*$⟩ vs PLS⟨OHI, $\not\prec$, $*$⟩ for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). Initial conditions are `RS` (left), `TS` (middle) and `HQS` (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

both problems, the single-objective algorithms were given two seconds for each scalarization.

4.2. Experimental Evaluation of Alternative Components

We now present the experimental evaluation of the anytime behavior of the PLS variants. To make the presentation concise, we present in this paper plots for only one bTSP instance, one bQAP instance with correlation −0.5 and another bQAP instance with correlation −0.75. Other instances show remarkably similar results, and the conclusions drawn in the paper are true for all instances we tested. The plots for all additional instances are available on-line as supplementary material [9].

4.2.1. Selection Step

We present in Fig. 4 the evaluation of the selection component that uses the optimistic hypervolume improvement (OHI) against the original random selection (RND). The plots on each row show the results obtained with different instances: one bTSP instance (top) and two bQAP instances with correlation −0.75 (middle) and correlation −0.5 (bottom). Each column of plots corresponds to a different initial scenario, that is, starting with a random solution (left), two high-quality solutions (middle) and a set of high-quality solutions (right). In most plots, PLS⟨OHI⟩ outperforms PLS⟨RND⟩ at any moment of the search, sometimes by a large gap. This result indicates that the selection of the most promising objective regions for further improvement actually helps reaching better results in less computation time. On the other hand, both strategies obtain similar results on a few cases, such as the top-left plot that presents

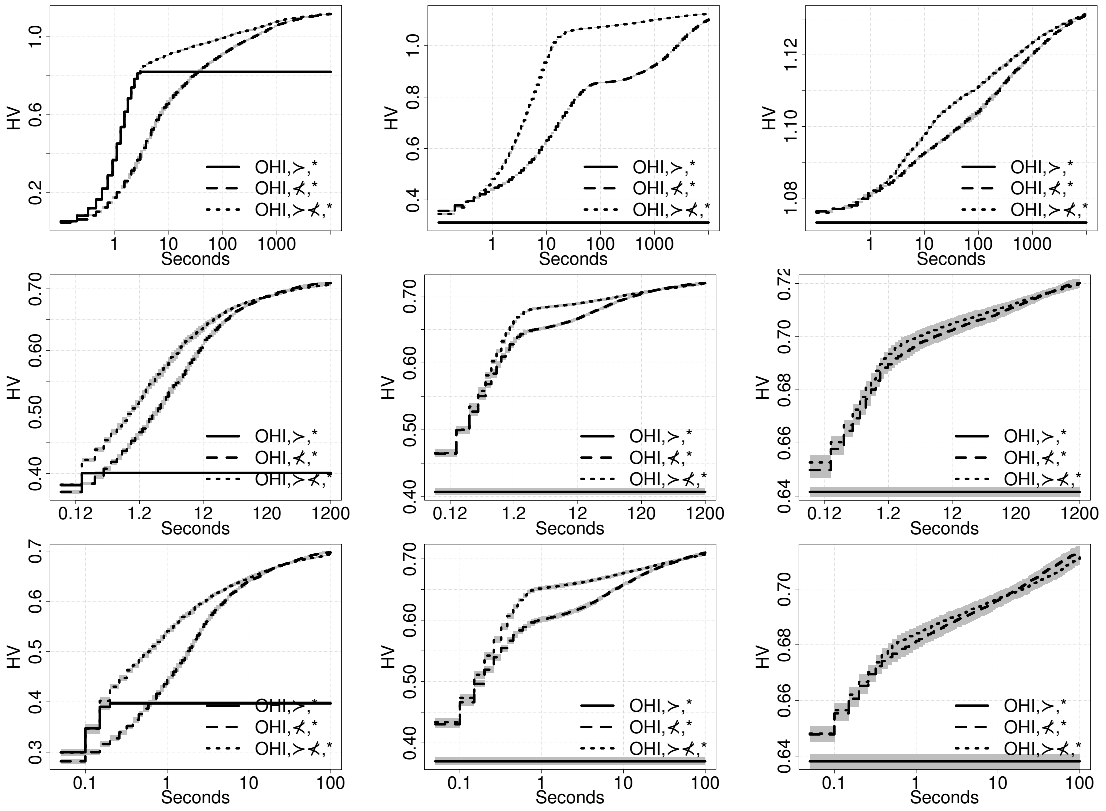

Figure 5: **Acceptance Criterion:** PLS⟨OHI, ≻, ∗⟩ vs PLS⟨OHI, ⊁, ∗⟩ vs PLS⟨OHI, ≻⊁, ∗⟩ for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). Initial conditions are RS (left), TS (middle) and HQS (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

results for a bTSP instance when starting from a random solution. This configuration is the only one where OHI is slightly worse for a short period of time (between 1 and 100 seconds), likely due to the overhead of computation time required to perform the selection using OHI.

Nonetheless, the selection based on OHI is, overall, a clear improvement over the original component. For this reason, we use OHI instead of RND for all the experiments in the rest of this section. Hence, we test whether we can improve the anytime behavior even further by using the alternative variants for the other components.

4.2.2. Acceptance Criterion

Figure 5 presents a comparison of the various alternative acceptance criteria proposed in Sec. 4. The variants PLS⟨OHI, ≻, ∗⟩ and PLS⟨OHI, ≻⊁, ∗⟩ behave exactly the same as long as at least one neighbor dominates the current solution. This clearly happens in the initial phases of the search when starting from a random solution (left column). However, PLS⟨OHI, ≻, ∗⟩ quickly reaches completion since it cannot find any new dominating solutions, whereas PLS⟨OHI, ≻⊁, ∗⟩ switches strategy and matches the final quality of PLS⟨OHI, ⊁, ∗⟩. When the initial set is close to the Pareto front (middle and right columns), no solution can be found that dominates any of the initial ones and, thus, the hypervolume curve is entirely flat for PLS⟨OHI, ≻, ∗⟩. However, as the switch from accepting only dominating neighbors to accepting also nondominated neighbors occurs for each new solution explored, there is still an observable difference between PLS⟨OHI, ≻⊁, ∗⟩ and PLS⟨OHI, ⊁, ∗⟩. In fact, PLS⟨OHI, ≻⊁, ∗⟩ compares very positively

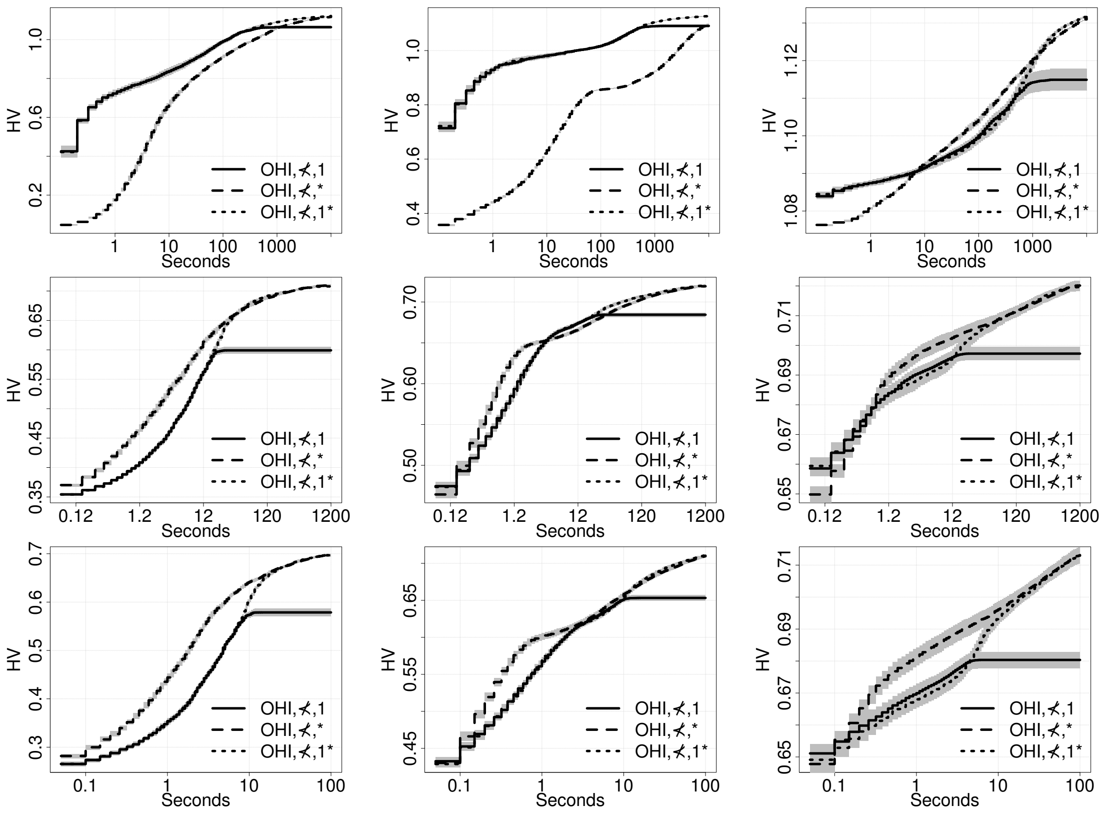

Figure 6: **Neighborhood Exploration:** PLS$\langle$OHI, $\nprec$, 1$\rangle$ vs PLS$\langle$OHI, $\nprec$, $*\rangle$ vs PLS$\langle$OHI, $\nprec$, 1$*\rangle$ for one bTSP instance (top), and two bQAP instances with correlation -0.75 (middle) and -0.5 (bottom). Initial conditions are RS (left), TS (middle) and HQS (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

to PLS$\langle$OHI, $\nprec$, $*\rangle$: in many cases the former shows a large improvement of quality over the latter at any time of the execution, for instance, in the plots for bTSP starting from RS (top-left) or TS (top-middle). In the case of HQS (right column), and specially for the bQAP (middle-right and bottom-right), the differences are smaller or not statistically significant, due to the fact that dominating solutions are very rarely found and the two variants are essentially equivalent. There is no case showing PLS$\langle$OHI, $\nprec$, $*\rangle$ better than PLS$\langle$OHI, $\succ\!\nprec$, $*\rangle$ and, overall, the switching component of PLS$\langle$OHI, $\succ\!\nprec$, $*\rangle$ yields a better anytime behavior than the original one.

4.2.3. Neighborhood Exploration

Figure 6 reports a comparison of the different alternatives for the neighborhood exploration. It shows the anytime behavior of three PLS variants: PLS$\langle$OHI, $\nprec$, 1$\rangle$, which stops the exploration of the neighborhood after accepting one solution, PLS$\langle$OHI, $\nprec$, $*\rangle$, which explores the full neighborhood, and PLS$\langle$OHI, $\nprec$, 1$*\rangle$, which switches from the first to the second rule. The behavior of PLS$\langle$OHI, $\nprec$, 1$*\rangle$ before the switch is exactly the same as PLS$\langle$OHI, $\nprec$, 1$\rangle$, therefore the two curves coincide until the point where PLS$\langle$OHI, $\nprec$, 1$\rangle$ stagnates due to the limited neighborhood exploration. It is clear, therefore, that PLS$\langle$OHI, $\nprec$, 1$*\rangle$ should be preferred over PLS$\langle$OHI, $\nprec$, 1$\rangle$ in every situation.

For the bTSP, when the initial set is either RS or TS, PLS$\langle$OHI, $\nprec$, 1$*\rangle$ shows a much better anytime behavior than the original component used by PLS$\langle$OHI, $\nprec$, $*\rangle$. However, for the bTSP when the initial set is HQS and for the bQAP in all cases, PLS$\langle$OHI, $\nprec$, $*\rangle$ has better anytime behavior than PLS$\langle$OHI, $\nprec$, 1$*\rangle$.

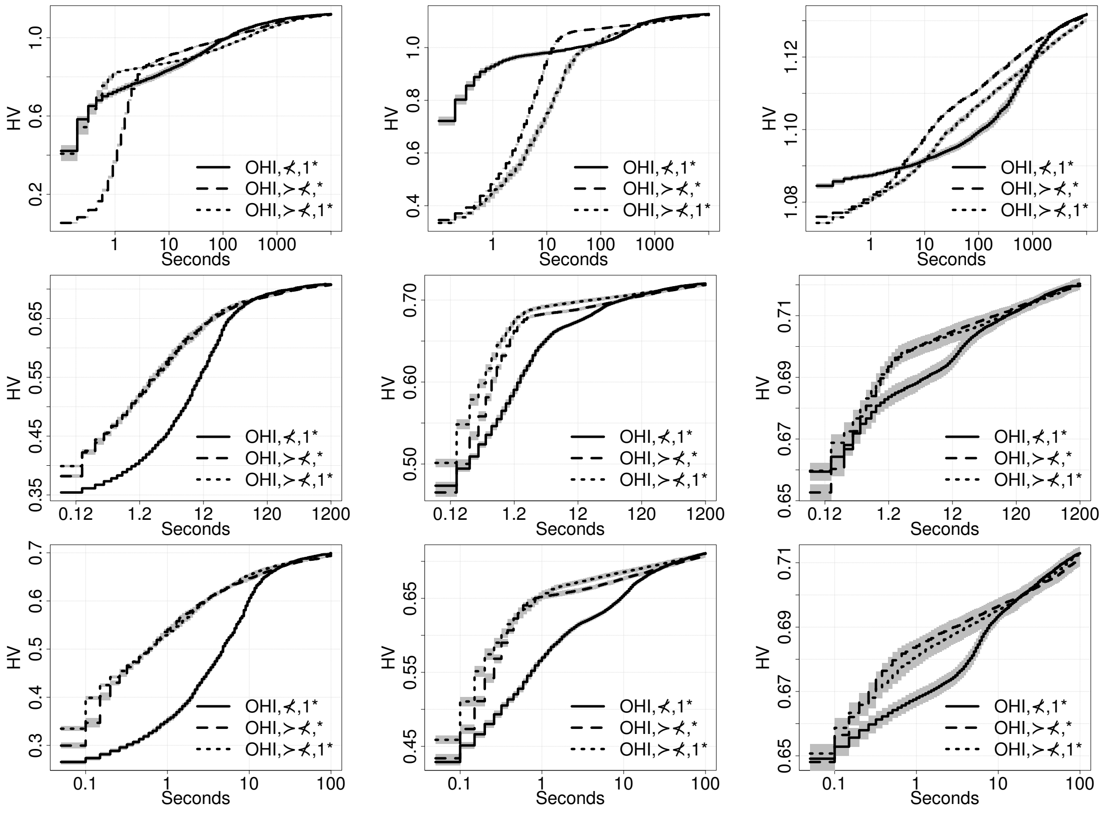

Figure 7: **Combination of alternative components:** PLS$\langle$OHI, $\not\prec$, $1*\rangle$ vs PLS$\langle$OHI, $\succ\!\not\prec$, $*\rangle$ vs PLS$\langle$OHI, $\succ\!\not\prec$, $1*\rangle$ for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). Initial conditions are `RS` (left), `TS` (middle) and `HQS` (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

Therefore, the best alternative for this component is dependent of the situation.

Since no strategy outperforms the other in every case, it could be interesting to test different trade-offs between PLS$\langle$OHI, $\not\prec$, $*\rangle$ and PLS$\langle$OHI, $\not\prec$, $1*\rangle$, for example, by accepting more than one solution and switching to a full neighborhood exploration later. However, it is likely that the best number of solutions to accept is strongly problem-dependent. Hence, we leave further exploration of this direction for future research.

4.2.4. Combination of Components

We have shown that switching the acceptance criterion during execution improves significantly the anytime behavior over the original acceptance criterion in most of the cases. We also have shown that switching from first-improvement to best-improvement can be helpful, but not in all cases. As a last step, we explore whether even better anytime behavior can be obtained by combining these two components in PLS$\langle$OHI, $\succ\!\not\prec$, $1*\rangle$. Fig. 7 compares PLS$\langle$OHI, $\succ\!\not\prec$, $1*\rangle$ with the strategies evaluated in the previous two sections. The plots show that in none of the cases, PLS$\langle$OHI, $\succ\!\not\prec$, $1*\rangle$ improves over its two competitors. Hence, there are interactions between the components that prevent their respective advantages to combine together. We conjecture that this is because PLS$\langle$OHI, $\succ\!\not\prec$, $*\rangle$ and PLS$\langle$OHI, $\not\prec$, $1*\rangle$ restrict in two different ways the number of solutions that are considered for inclusion in the archive and that probably the restriction incurred by their combination is too strong.

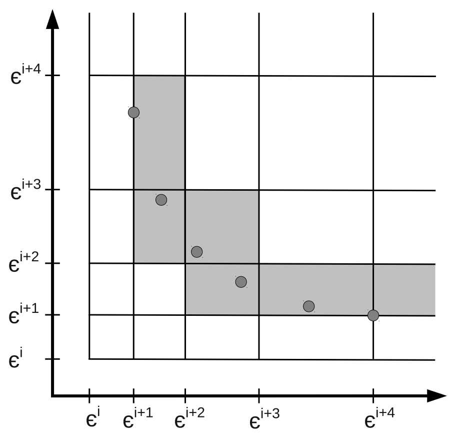

Figure 8: Representation of the normalized objective space with the epsilon-grid mechanism. This mechanism does not allow a new solution s' to enter a box already filled by another solution s (those boxes are shown in gray), unless s' dominates s. (Adapted from [1]).

5. Dynagrid: Dynamic Objective Space Discretization for Anytime PLS

The best performing PLS variants for neighborhood exploration and acceptance criterion have in common that they restrict the number of non-dominated solutions added to the archive. In particular, such a restriction improves the anytime behavior in the early phases of the search by driving PLS towards regions in the objective space close to the Pareto front. This pressure is exerted by avoiding the acceptance of too many non-dominated solutions into the archive.

In this section, we exploit further this observation. We explicitly limit the potential number of non-dominated solutions to enter the archive by incorporating an epsilon-grid mechanism. We first explain the static version of the epsilon-grid mechanism, which was proposed by Angel et al. [1]. Later, we propose a dynamic adaptation of the grid, which we call *Dynagrid*. Finally, we show that *Dynagrid*, and by extension the anytime PLS, can be further improved by taking into account the hypervolume contribution of each point in the grid. The selection criterion based on the OHI was shown to be consistently better than the random version, therefore we use it for all variants designed in this section.

5.1. Static Epsilon-Grid Discretization

The *epsilon-grid* mechanism proposed by Angel et al. [1] discretizes the objective space into a grid of boxes whose size increases with the distance to the axis; in this way, more solutions can be accepted in the center of the Pareto front than in the extremes. The size of the boxes is defined by a parameter ϵ: given $\epsilon > 1$, for each objective d, normalized such the minimum value is 1, the bounds of boxes $i = 0, 1, 2, \ldots$ are computed as

$$B_i^d = \left[\epsilon^i, \epsilon^{i+1}\right[\tag{3}$$

Figure 8 shows graphically this epsilon-grid mechanism. We can determine directly the box corresponding to solution s for objective d by:

$$B^d(s) = B^d_{\lfloor log(norm(f_d(s)))/log(\epsilon) \rfloor} \tag{4}$$

The *norm*() function gives the normalized value for a given objective. In this paper, we use a mapping of the objectives such that the lower bound provided for a given objective (see section 4.1.2) is mapped to 1, the upper bound to 100 000, and any value in between is mapped linearly to the range [1, 100 000]. Using a normalization mechanism whenever a decision must be taken based on objective values is important to not emphasize one objective over another when their ranges differ. It also allows applying the same algorithm to problems with objectives having different scales.

The main drawback of the epsilon-grid mechanism is that the parameter ϵ (Eqs. 3 and 4) must be defined carefully before solving a problem. Therefore, it can not be applied without previous knowledge of the problem (and even instance) to be tackled [36].

Algorithm 2 Dynagrid (Dynamic Epsilon-Grid)

```
1: Input: Initial set of non-dominated solutions: $\mathcal{A}_0$,
2:        Value that defines the initial grid: $\epsilon_0$,
3:        Ratio to decrease the value of $\epsilon$: $r$.
4: $\mathcal{A} := \mathcal{A}_0$
5: $\epsilon := \epsilon_0$
6: while ! termination criterion do
7:     $\mathcal{A} := \mathsf{PLS}_{Grid}(\mathcal{A}, \epsilon)$
8:     $\epsilon := 1 + (\epsilon - 1) \cdot r$
9:     $\mathsf{explored}(s) := \text{FALSE}, \quad \forall s \in \mathcal{A}$
10: end while
11: Output: $\mathcal{A}$
```

5.2. Dynamic Adaptation of Epsilon-Grid: Dynagrid

Here, we propose *Dynagrid*, a new variant of the epsilon-grid mechanism that dynamically adapts the value of ϵ and, hence, the grid size, within the PLS algorithm. The aim of this mechanism is threefold. First, it allows the application of this method without any previous knowledge on the problem or the instance to be tackled. Second, it improves the anytime behavior of PLS, as we show later. Third, it can be applied directly to other MCOPs and, possibly also, to other algorithms for MCOPs.

The dynamic adaptation of ϵ is triggered by the convergence of PLS. Each time PLS using a fixed epsilon-grid terminates, that is, all solutions are marked as explored, the value of ϵ is reduced by a fixed factor. This mechanism is described in pseudo-code in Algorithm 2. A PLS algorithm using the grid defined by ϵ (denoted by PLS_{Grid}, line 7 of the pseudo-code) starts from the current solution archive $\mathcal{A}$. When PLS_{Grid} reaches completion and all solutions in the archive are marked as explored, the value of ϵ is decreased according to a parameter $r < 1$ (line 8), and the solutions are marked again as unexplored. Then PLS_{Grid} is re-started using the grid defined by the updated ϵ value (line 9). This process (lines 6 to 10) is repeated until a given termination criterion is met.

After a change of the value of ϵ, the grid changes and, as a result, there may be boxes that contain more than one solution. We do not remove these extra solutions, since the quality of the archive will decrease and removing these solutions would incur a computational overhead.

The rationale behind *Dynagrid* is to obtain quickly well-distributed solutions in the objective space, thus reaching quickly a good Pareto front approximation in terms of hypervolume, using large initial values for ϵ, and then reducing the value of ϵ to refine the grid and add more solutions only when PLS actually needs them to improve the quality of the results.

In our experimental analysis, we set the initial value of ϵ to 5. This is a rather large value (recall that ϵ has an exponential effect on the boxes size), which makes unlikely even the acceptance of one new solution when starting from the initial set. This is done on purpose to avoid the usage of problem-specific knowledge as much as possible. For the value of parameter r, we tested values in $\{0.9, 0.7, 0.5, 0.3, 0.1\}$ during preliminary experiments. The differences due to different values of the parameter r were minor, resulting in rather similar curves of the development of the hypervolume; we provide the plots for this comparison as supplementary material [9]. For the following experiments, we chose a setting of $r = 0.5$.

Figure 9 compares the *Dynagrid* algorithm against variants that use different static grids, that is, they use different fixed values for ϵ. We tested values of $\epsilon \in \{1.5, 1.05, 1.005\}$, but for the sake of clarity we show here only those for value 1.5 and 1.005; the results for 1.05 are a trade-off between the results obtained by the two extreme values. Clearly, the larger ϵ, the faster is the quality improvement in short computation time, but also the earlier PLS stagnates, as visible by the fact that curves become flat (that is, the hypervolume does not increase anymore), indicating that PLS actually stops.

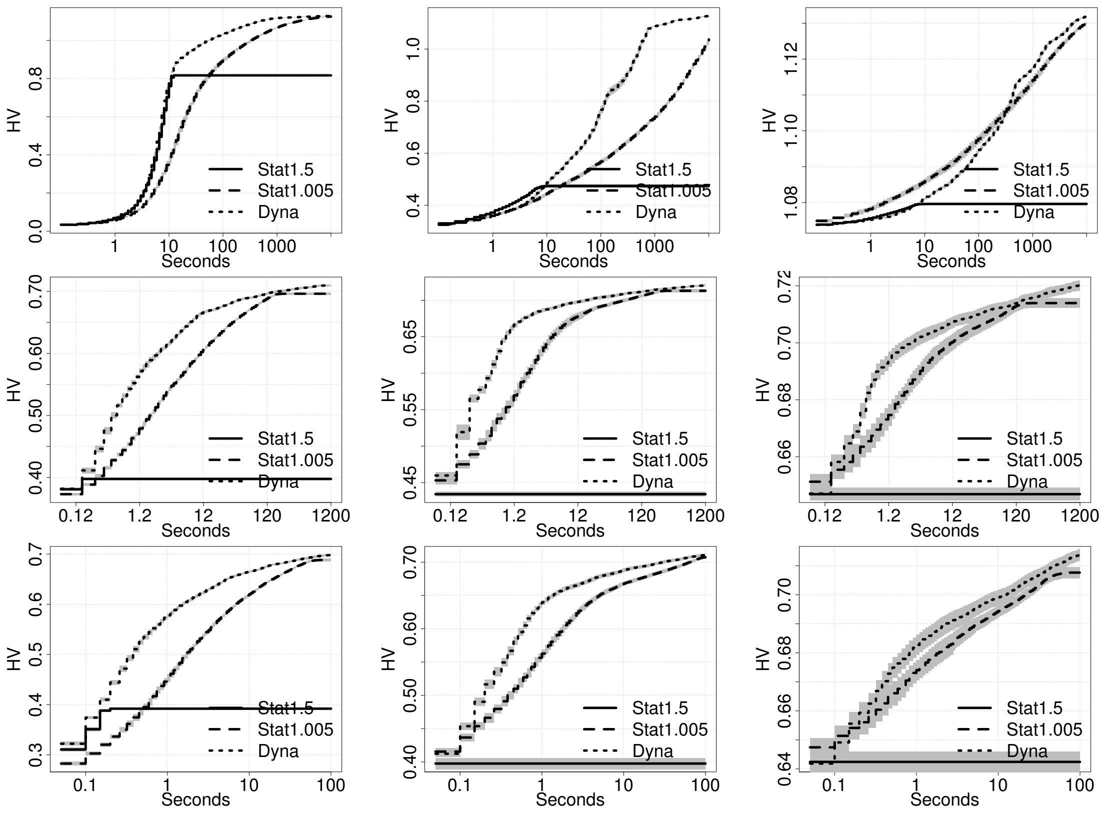

Figure 9: Experimental evaluation of PLS combined with Epsilon-Grid for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). We compare here two variants that use a static grid (with $\epsilon \in \{1.05, 1.005\}$), against the version that uses a dynamic one. Initial conditions are `RS` (left), `TS` (middle) and `HQS` (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

For the bTSP with initial conditions `RS` and `TS`, the adaptive mechanism is able to improve the quality of the archive as quickly as the coarse static grid ($\epsilon = 1.5$) does it. Moreover, it outperforms the fine static grid ($\epsilon = 1.005$) at any time. Starting from `HQS` (top-right plot in Fig. 9), it gives slightly worse solution quality than the fine grid until roughly 200 seconds, but it is better for larger computation time. For the bQAP, the results obtained by the dynamic grid are clearly better than those for fixed values of ϵ. The dynamic grid improves quality quickly and reaches higher quality results than the fine grid at any time for all initial conditions.

Overall, the good performance of the dynamic grid is remarkable because it does not require preliminary knowledge of an appropriate setting of ϵ when being applied.

5.3. Improving Dynagrid By Considering Hypervolume Contribution

In the original epsilon-grid method [1], a new candidate solution s' that falls into the same box as an already existing solution s in the archive only replaces s if s' dominates s. Therefore, solutions that enter the archive earlier are preferred over more recent candidate solutions for the sole reason that they were encountered earlier. However, since our goal is to obtain a set of non-dominated solutions with the highest possible quality at any time, it may be preferable to relax this condition if the new candidate solution improves the quality measure over the existing one. More precisely, since we measure quality in terms of hypervolume (see Section 2.1), we extend our dynamic epsilon-grid (see Section 5.2) to optimize this indicator. In particular, we allow a solution s' to replace a solution s if the hypervolume of the archive increases. More formally, given a new solution s' that falls into the same box as another solution $s \in \mathcal{A}$, where $\mathcal{A}$ is the current archive, s' replaces s if $s' \prec s$ or $\mathsf{HV}(\mathcal{A}/\{s\} \cup \{s'\}) > \mathsf{HV}(\mathcal{A})$. Note that if $s' \prec s$, due to the properties of the hypervolume indicator [46], the hypervolume of the archive resulting from the removal of s and the addition of s' can only increase. Therefore, the dynamic grid considering *hypervolume contribution* aims explicitly at optimizing the hypervolume of the current archive at any time.

If there are several solutions in the box, then solution s above corresponds to the one with the least hypervolume contribution. The computation of the least hypervolume contribution is computationally cheap in two and three objectives: Computing all hypervolume contributions of a set of n points requires $O(n \log n)$ [13]. In our case, we do it in linear time since the set is kept sorted (according to one objective). For a higher number of objectives, it is unclear if the improvement in terms of quality would compensate the computational overhead; we leave this for future research.

In Fig. 10, we compare the performance, in terms of anytime behavior, of Dynagrid when considering hypervolume contributions, called *Dynagrid-HV*, versus the basic *Dynagrid* algorithm, which was shown to outperform the static grid in the previous section. For the bTSP, with initial conditions RS, considering the hypervolume contribution slows down the initial increase of the solution quality between roughly 2 and 20 seconds. This is probably explained by the fact that the grid size is updated later with the hypervolume enhancement than without, as several solutions can successively enter in the box(es), postponing the decrease of epsilon. For longer computation time, however, *Dynagrid-HV* clearly improves over the *Dynagrid* algorithm. For the initial conditions TS and HQS, the improvement is very large: *Dynagrid-HV* reaches the same quality than the *Dynagrid* algorithm in about one order of magnitude shorter computation time. For the bQAP, *Dynagrid-HV* shows, for any given computation time, the same or better average quality than the *Dynagrid* algorithm, although the improvement is typically smaller than in the bTSP case. The improvements are clear for RS and TS initial sets; when starting from HQS, both alternatives bring similar results.

In summary, *Dynagrid-HV* improves further the anytime behavior of PLS. Since in the majority of cases and termination criteria, *Dynagrid-HV* is clearly better than the basic *Dynagrid* algorithm, we would recommend its use when tackling bi-objective problems with similar characteristics as the ones we tackle here.

6. Comparison of Alternative PLS Components versus Dynagrid

In previous sections, we designed variants of PLS with better anytime behavior by exploring two fundamentally different directions. Here, we compare these two approaches. Figure 11 presents a comparison of the best algorithms obtained from each of the two alternatives. Additionally, to highlight the improvement obtained by these new PLS variants over the original PLS algorithm, we include also the original PLS in this comparison. For the bTSP, PLS$\langle\mathsf{OHI}, \nprec, 1{*}\rangle$ dominates completely the original PLS algorithm. It also performs better than *Dynagrid-HV* up to roughly 20 seconds, when the hypervolume-over-time curves of the two algorithms cross. This crossing may be explained by the fact that *Dynagrid-HV* requires some time to adapt the discretization, which penalizes it for very short runtimes. For the bQAP, PLS$\langle\mathsf{OHI}, \succ\nprec, {*}\rangle$ consistently and completely surpasses the original PLS algorithm, but it is itself mostly outperformed by *Dynagrid-HV*.

6.1. Scaling Behavior for Larger Instances

To determine whether our conclusions are consistent across different instance sizes, we analyze how the experimental results scale when tackling larger instances. To do so, we generated larger instances of size 1000 for the bTSP and size 150 for the bQAP, following the same setup used for the other instances (see Section 4.1). Table 2 presents the cut-off times for these instances for each problem.

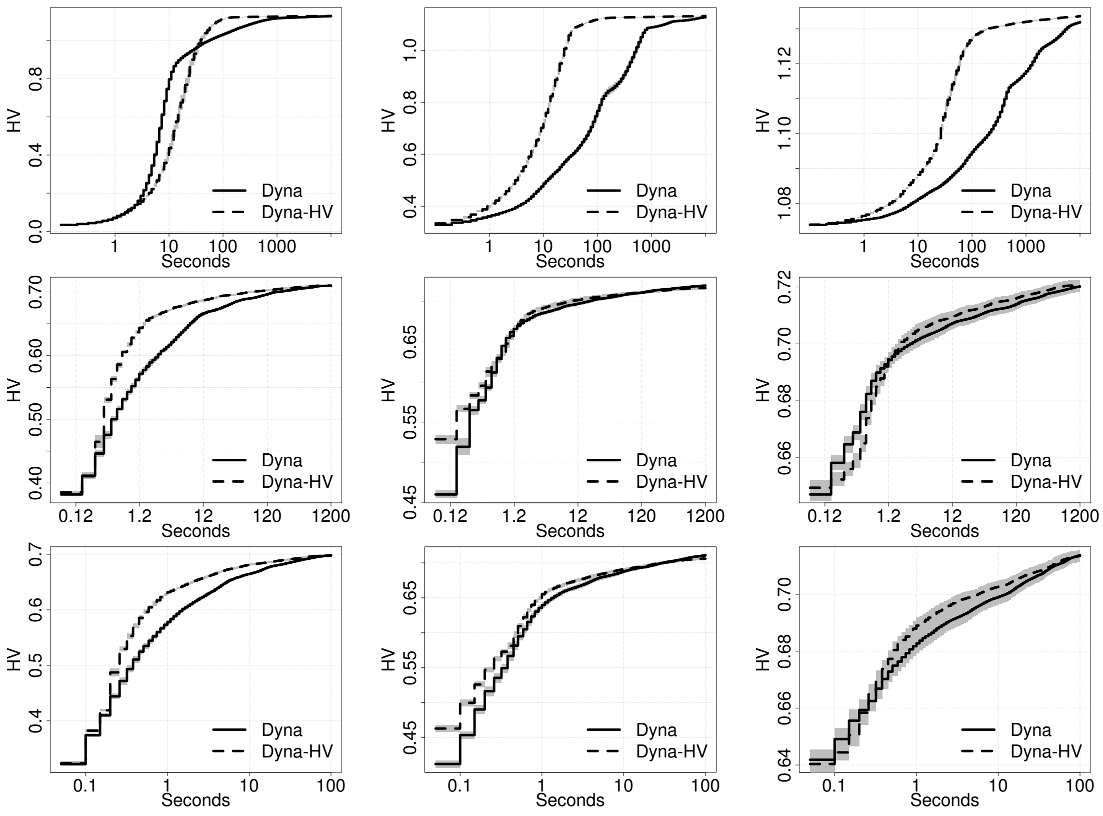

Figure 10: Experimental analysis of PLS combined with Epsilon-Grid for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). The plots present the comparison of the quality obtained by the dynamic epsilon grid, with and without the proposed hypervolume enhancement. Initial conditions are `RS` (left), `TS` (middle) and `HQS` (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

We present a comparison of the same algorithms on these larger instances in Fig. 12. All the trends observed for smaller instances not only remain true, but are strengthened. In fact, the gap between the original PLS algorithm and the two new anytime PLS variants widens, as it is most clearly visible on the bTSP. Hence, the performance improvements introduced by the proposed anytime PLS algorithms over the original PLS are particularly important when tackling large-scale bi-objective problems.

6.2. *Statistical Comparison of the Best Anytime PLS Algorithms*

We perform a statistical comparison of the anytime PLS algorithms that were previously evaluated in this section. For the statistical tests, we focus on specific snapshots in time that are at a 1000^{th}, a 100^{th}, and a 10^{th} of the cut-off time, and the cut-off time itself. Since the time steps that we use increase exponentially (see Section 4.1), they do not necessarily correspond exactly to these selected times: in this case we use the closest ones.

We use the Friedman test to assess the significance of the differences in terms of hypervolume, with a standard confidence level of 0.95 (thus, a p-value < 0.05 indicates that the null hypothesis is to be rejected). In the Friedman test, each instance and run is considered as a different block. Table 3 presents this comparison for both problems. These results confirm (i) the improvement of the proposed anytime PLS variants over the original PLS algorithm, and (ii) that *Dynagrid-HV* is almost always statistically significantly better than other strategies.

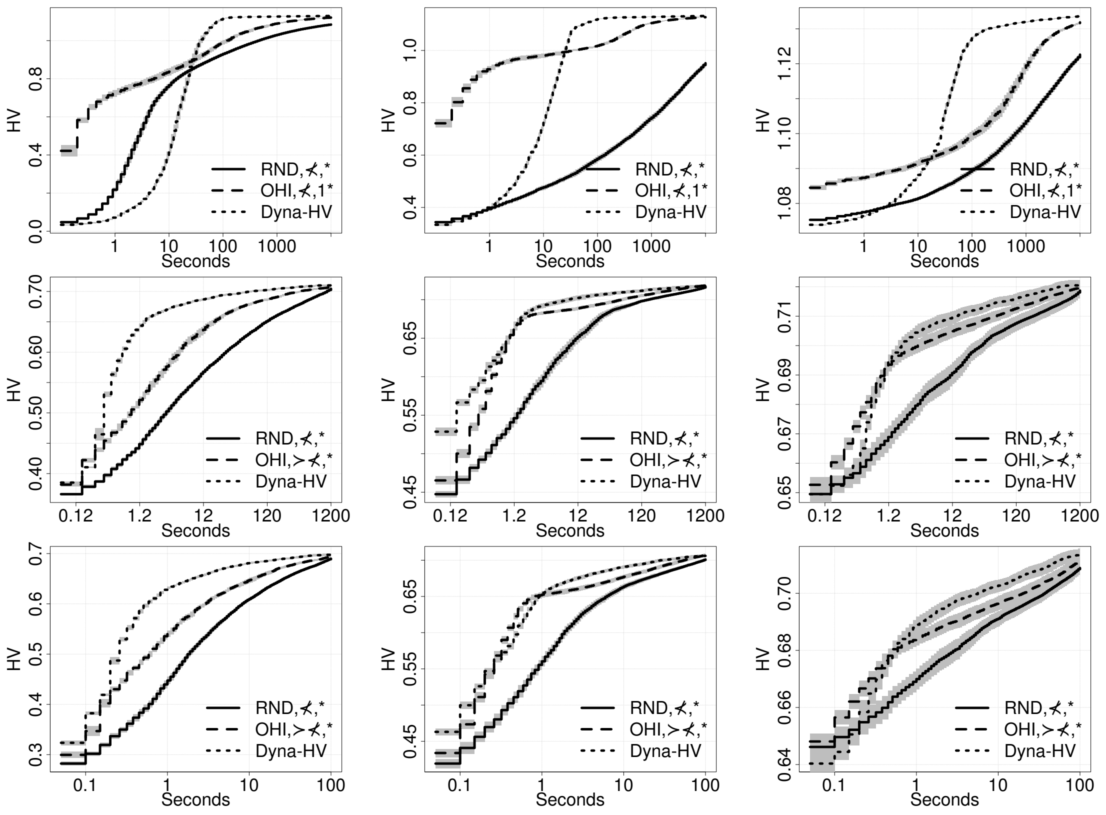

Figure 11: Experimental comparison of the original PLS algorithm, the best anytime PLS obtained in Section 4 (PLS⟨OHI, $\not\prec$, $1*$⟩ for the bTSP and PLS⟨OHI, $\succ\not\prec$, $*$⟩ for the bQAP), and the best anytime PLS obtained in Section 5 (*Dynagrid-HV*), for one bTSP instance (top), and two bQAP instances with correlation −0.75 (middle) and −0.5 (bottom). Initial conditions are `RS` (left), `TS` (middle) and `HQS` (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

7. Literature Survey of Related Work

Many previous works are based on the original PLS algorithm, here we only mention those that are closer to our work. A more complete list of references can be found in a recent overview [10].

Liefooghe et al. [27, 28] also decompose the PLS algorithm into different components and study the performance of some of their variants. Nonetheless, there are significant differences between their study and our proposals here. First, they do not distinguish between acceptance criterion and neighborhood exploration, and they do not consider variants that switch strategies during a single run of PLS, such as the ones studied here (Sec. 4). Thus, we explore a more refined decomposition of PLS and therefore also test more combinations. Second, their goal is to compare the results obtained after a given computation time. More precisely, in their experimental setup, they choose the same, predefined computation time limit for all variants, and variants that would finish before this computation time limit are restarted from scratch. A variant is then judged by the final aggregated non-dominated set found across the multiple restarts. Here, we focus on the anytime behavior of the resulting algorithms, making our study rather different.

PLS [41] and similar algorithms [1] are used as either stand-alone algorithms [38, 41] or as components of hybrid algorithms [7, 33, 34]. When tackling large instances, these algorithms may require a long time to reach completion. One solution is to stop the algorithm before completion [7]. In this case, our proposed anytime PLS variants will produce much better results. An alternative solution is to

Table 2: Cut-off times used for large instances of each problem.

Problem	Instance type	Cut-off time (s)
bTSP	Size = 1000	30 000
bQAP	Size = 150, Correlation = −0.75	2 000
	Size = 150, Correlation = −0.5	1 000

reduce the completion time by using problem-specific speed-up techniques to avoid exploring the whole neighborhood of each solution [33]. Our proposed variants can be combined with these problem-specific techniques in order to also improve the anytime behavior.

Drugan and Thierens [4, 5] focus on PLS-based algorithms that can successfully handle the situation where PLS reaches completion and still additional computation time is available. They show that promising results can be attained when restarting PLS from new solutions on a "path" between two solutions in the Pareto local optimum set (in a manner similar to the *path-relinking* [17] mechanism). A different approach to handle the same situation can be found in Geiger [16], where the author proposes to apply a different neighborhood operator when PLS converges to a Pareto local optimum: this idea can be seen as an adaptation of *variable neighborhood descent* [18] to PLS. Despite the fact that their goal is different from ours, these studies can be seen as complementary to this paper: they focus on time scales that are larger than the time required by PLS to terminate, while we focus here on computation times that are shorter or roughly similar to the time required by the original PLS to reach completion.

The Dynagrid mechanism proposed here is related to the concept of bounded archiving, that is, limiting the size of an archive of non-dominated solutions. In particular, the idea of using a hyper-grid on the objective space to limit the number of solutions in a nondominated archive has been used extensively in the bounded archiving literature [31]. The Pareto archived evolution strategy (PAES) [22] can be understood as a random mutation hill-climber that uses a discrete grid to limit the size of its archive of solutions. One difference between PAES and PLS is that PAES uses a random exploration of neighborhoods while PLS uses a systematic one analogous to iterative improvement algorithms. The adaptive grid mechanism of PAES is quite different from Dynagrid, since the former allows a new solution to enter an already populated box as long as the maximum archive size has not been reached. By contrast, Dynagrid does not take into account a maximum archive size, and new solutions can only enter an already populated box by replacing one of the solutions already in the box. Moreover, the grid mechanism in PAES is adaptive because the grid is recomputed when the ranges in the archive change, but the number of subdivisions is static. By contrast, Dynagrid increases the number of subdivisions over time in order to refine the current approximation to the Pareto front. The ϵ-Pareto archiving algorithm [26] also uses an epsilon-grid in the objective space, but the acceptance criterion is slightly different from the static epsilon-grid mechanism [1] (and by extension from Dynagrid). In ϵ-Pareto archiving, dominance is first tested with respect to the boxes in the grid, whereas epsilon-grid always considers dominance among solutions. The dynamic version of ϵ-Pareto archiving starts from a small ϵ and increases it every time the archive size exceeds a predefined maximum size. Its goal is to maintain a fixed-size archive with minimal ϵ. The multi-level grid archiving algorithm [25] does not actually use an explicit ϵ value but instead considers a succession of coarse-grained boxes when deciding whether to accept each individual solution. The goal is again to maintain a fixed-size archive with a good approximation quality. The dynamic adaptation of ϵ in Dynagrid is fundamentally different. It starts with a large ϵ and decreases it every time PLS converges to a Pareto local optimum w.r.t. the current value of ϵ. It allows PLS to quickly converge to a well-spread approximation set and, once PLS converges, it allows PLS to continue the search by refining the grid.

Although there has been some work on anytime algorithms for single-objective optimization problems (e.g., [29, 32, 45]), the anytime behavior of multi-objective optimization algorithms is rarely considered.

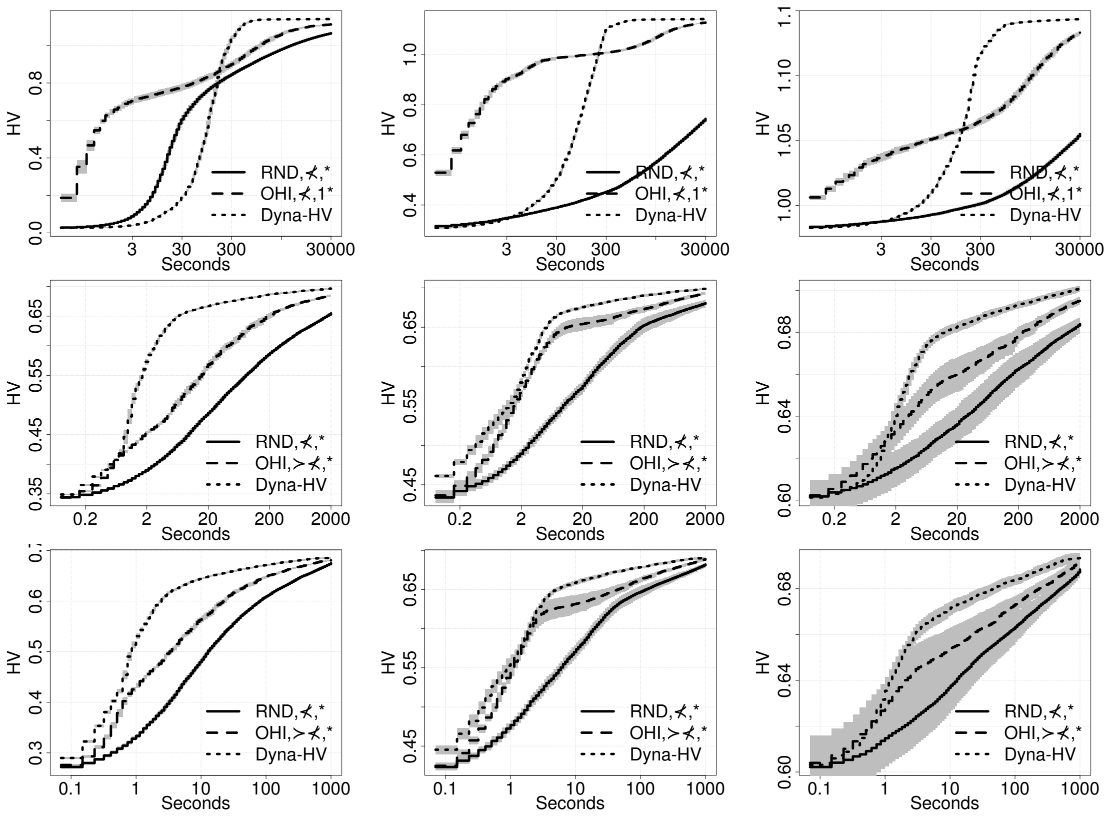

Figure 12: Experimental comparison of the original PLS algorithm, the best anytime PLS obtained in Section 4 (PLS$\langle$OHI, $\not\succ$, $1*\rangle$ for the bTSP and PLS$\langle$OHI, $>\not\succ$, $*\rangle$ for the bQAP), and the best anytime PLS obtained in Section 5 (*Dynagrid-HV*), for one large bTSP instance of size 1000 (top), and two large bQAP instances of size 150 with correlation -0.75 (middle) and -0.5 (bottom). Initial conditions are RS (left), TS (middle) and HQS (right). The gray area corresponding to each curve shows the 95% confidence interval across different runs.

With respect to evolutionary multi-objective algorithms, some preliminary work has shown that the anytime behavior of some popular algorithms is rather poor when using their default settings [43]. A good anytime behavior is specially crucial for algorithms that have a natural stopping criterion but are often stopped earlier in practice, such as PLS. In a previous work, we have studied the anytime behavior of Two-Phase Local Search (TPLS) [6], which is a different local search algorithm for multi-objective combinatorial optimization. Our work here shares a similar goal, that is, to improve the anytime behavior of an existing local search algorithm. Nonetheless, the improvements proposed to achieve such a goal are fundamentally different because TPLS and PLS follow completely different search paradigms: TPLS is based on scalarizations, whereas PLS is based on dominance [39].

Finally, recent studies [29, 43] have applied automatic configuration tools, also called tuners, to automatically select and tune the algorithmic components and parameters of optimization algorithms to obtain a good anytime behavior. This approach can only select among the components provided and not devise new ones as we have done here. Applying automatic configuration to the original PLS in order to improve its anytime behavior would not bring much improvement, since its components do not provide a good anytime behavior. On the other hand, the diversity of behaviors demonstrated by the anytime PLS variants proposed here, where particular variants are better for certain problem characteristics, suggests that automatic configuration can be applied to select the appropriate variant when tackling a new problem.

8. Conclusion

To the best of our knowledge, our present work and our preliminary study [8] are the first to analyze and improve the anytime behavior of PLS, a high-performing multi-objective local search algorithm, and an important component of state-of-the art algorithms for several well-known MCOPs [7, 33, 34, 35].

The experimental analysis presented in this work studied the impact of various alternatives for the algorithmic components of PLS. In particular, we considered different algorithm components for the selection of the solutions to be explored, the type of neighborhood exploration and the acceptance criteria for new solutions. We also analyzed PLS variants that switch from one alternative to a different one during a single run of PLS. Our results show that such "switching" variants of PLS improve significantly the anytime behavior over the original PLS. However, we found that the best variants are different for the bTSP and the bQAP, and, thus, the best variant depends on the problem being tackled.

In addition, we proposed a radically different anytime PLS algorithm based on a dynamic discretization of the objective space, where the objective space is divided into a grid. Our results clearly showed that the dynamic adaptation of the grid size outperforms any static value in terms of anytime behavior. Moreover, we found that results could be further improved by taking into account the hypervolume contribution when deciding whether to add a new solution to an already populated box of the grid. The resulting PLS variant, which we call *Dynagrid-HV*, is consistently better than the other anytime PLS variants examined in this paper and completely outperforms the original PLS for any given termination time.

Since our experimental study considered three different scenarios that include starting PLS from random solutions of typically poor quality and also from nearly-Pareto-optimal solutions, we are certain that our conclusions are useful for applications of PLS as a stand-alone algorithm and also for hybrid algorithms where PLS is a major component. In particular, by replacing the original PLS with the proposed anytime PLS variants, we expect significant improvements in the results obtained by the current state-of-the-art algorithms for the multi-objective traveling salesman problem [34], various bi-objective permutation flowshop problems [7], and the bi-objective multi-dimensional knapsack problem [33].

Moreover, a combination of the variants proposed in this paper with the mechanisms that were proposed to restart PLS [4, 5] once it has reached a Pareto local optimum [42] is also promising. Such a combination will make the resulting algorithm applicable to a very wide range of computation times and, thus, real-world situations.

Our study has considered the standard neighborhood operators for the benchmark problems tackled. However, the choice of neighborhood operators has a strong influence on the number and quality of nondominated solutions found and, thus, it impacts the anytime behavior of PLS. A possible direction of future research is to study the influence of different neighborhoods on the anytime behavior of PLS.

Finally, our work has focused on bi-objective problems. We plan to extend our analysis to three and more objectives. The number of nondominated solutions and the complexity of some algorithmic components increase exponentially with the number of objectives. Thus, such problems pose different computational challenges that may require different algorithmic components to design PLS algorithms with good 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, the COMEX project within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office, and by the MIBISOC network, an Initial Training Network funded by the European Commission, grant PITN–GA–2009–238819. Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a senior research associate, respectively. The authors thank Mohamed S. bin Hussin for providing his Simulated Annealing implementation for single-objective QAP instances.

References

[1] E. Angel, E. Bampis, and L. Gourvés. Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem. *Theoretical Computer Science*, 310(1-3):135–146, 2004.

[2] D. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. Concorde TSP solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`, 2014. Version visited last on 15 April 2014.

[3] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[4] M. M. Drugan and D. Thierens. Path-guided mutation for stochastic Pareto local search algorithms. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 485–495. Springer, Heidelberg, Germany, 2010.

[5] M. M. Drugan and D. Thierens. Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, 18(5):727–766, 2012.

[6] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Improving the anytime behavior of two-phase local search. *Annals of Mathematics and Artificial Intelligence*, 61(2):125–154, 2011.

[7] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers & Operations Research*, 38(8):1219–1236, 2011.

[8] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Pareto local search algorithms for anytime bi-objective optimization. In J.-K. Hao and M. Middendorf, editors, *Proceedings of EvoCOP 2012 – 12th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 206–217. Springer, Heidelberg, Germany, 2012.

[9] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Supplementary material: Anytime pareto local search. `http://iridia.ulb.ac.be/supp/IridiaSupp2013-003`, 2013.

[10] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Combining two search paradigms for multi-objective optimization: Two-Phase and Pareto local search. In E.-G. Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 97–117. Springer Verlag, 2013.

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

[12] M. Ehrgott and X. Gandibleux. Hybrid metaheuristics for multi-objective combinatorial optimization. In C. Blum, M. J. Blesa, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An emergent approach for optimization*, pages 221–259. Springer, Berlin, Germany, 2008.

[13] M. T. M. Emmerich and C. M. Fonseca. Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 121–135. Springer, Heidelberg, Germany, 2011.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman & Co, San Francisco, CA, 1979.

[15] D. Garrett and D. Dasgupta. Multiobjective landscape analysis and the generalized assignment problem. In V. Maniezzo, R. Battiti, and J.-P. Watson, editors, *Learning and Intelligent Optimization, Second International Conference, LION 2*, volume 5313 of *Lecture Notes in Computer Science*, pages 110–124. Springer, Heidelberg, Germany, 2008.

[16] M. J. Geiger. Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology. *Computers and Industrial Engineering*, 61(3):805–812, 2011.

[17] F. Glover. A template for scatter search and path relinking. In J.-K. Hao, E. Lutton, E. M. A. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pages 1–51. Springer, Heidelberg, Germany, 1998.

[18] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[19] H. H. Hoos and T. Stützle. *Stochastic Local Search—Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[20] M. S. Hussin and T. Stützle. Tabu search vs. simulated annealing for solving large quadratic assignment instances. *Computers & Operations Research*, 43:286–291, 2014.

[21] A. Jaszkiewicz, H. Ishibuchi, and Q. Zhang. Multiobjective memetic algorithms. In *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*, pages 201–217. Springer, 2012.

[22] J. D. Knowles and D. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[23] J. D. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization, EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 295–310. Springer, Heidelberg, Germany, 2003.

[24] J. D. Knowles, D. Corne, and M. Fleischer. Bounded archiving using the Lebesgue measure. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2490–2497. IEEE Press, Piscataway, NJ, Dec. 2003.

[25] M. Laumanns and R. Zenklusen. Stochastic convergence of random search methods to fixed size Pareto front approximations. *European Journal of Operational Research*, 213(2):414–421, 2011.

[26] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[27] A. Liefooghe, S. Mesmoudi, J. Humeau, L. Jourdan, and E.-G. Talbi. A study on dominance-based local search approaches for multiobjective combinatorial optimization. In T. Stützle, M. Birattari, and H. H. Hoos, editors, *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. SLS 2009*, volume 5752 of *Lecture Notes in Computer Science*, pages 120–124. Springer, Heidelberg, Germany, 2009.

[28] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E.-G. Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2):317–352, 2011.

[29] M. López-Ibáñez and T. Stützle. Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, 235(3):569–582, 2014.

[30] M. López-Ibáñez, L. Paquete, and T. Stützle. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5 (1):111–137, 2006.

[31] M. López-Ibáñez, J. D. Knowles, and M. Laumanns. On sequential online archiving of objective vectors. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 46–60. Springer, Heidelberg, Germany, 2011.

[32] S. Loudni and P. Boizumault. Combining VNS with constraint programming for solving anytime optimization problems. *European Journal of Operational Research*, 191:705–735, 2008.

[33] T. Lust and J. Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In C. A. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 119–141. Springer, 2010.

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

[35] T. Lust and J. Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.

[36] L. Paquete. *Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimization: Methods and Analysis*. PhD thesis, FB Informatik, TU Darmstadt, Germany, 2005.

[37] L. Paquete and T. Stützle. A two-phase local search for the biobjective traveling salesman problem. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization, EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 479–493. Springer, Heidelberg, Germany, 2003.

[38] L. Paquete and T. Stützle. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research*, 169(3):943–959, 2006.

[39] L. Paquete and T. Stützle. Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, pages 29–1—29–15. Chapman & Hall/CRC, Boca Raton, FL, 2007.

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

[41] L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–200. Springer, Berlin, Germany, 2004.

[42] L. Paquete, T. Schiavinotto, and T. Stützle. On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 156:83–97, 2007.

[43] A. Radulescu, M. López-Ibáñez, and T. Stützle. Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-criterion Optimization, EMO 2013*, volume 7811 of *Lecture Notes in Computer Science*, pages 825–840. Springer, Heidelberg, Germany, 2013. ISBN 978-3-642-37139-4.

[44] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens. On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. *European Journal of Operational Research*, 227(2):331–342, 2013.

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

[46] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Table 3: Statistical analysis of the best anytime variants of PLS and the original PLS at different time steps. The time steps are those from $(\texttt{cutoff\_time} + 1)^{(i/100)} - 1, \quad i \in 1, \ldots, 100$ (see Section 4.1.2) that are closest to $\texttt{cutoff\_time}/(10^k)$, with $k \in \{0, 1, 2, 3\}$. Each run of the algorithms is ranked according to the hypervolume of its output. The numbers in parenthesis are the differences of the sum of ranks relative to the best variant for 30 independent runs. ΔR_α gives the difference of the sum of ranks that is statistically significant, for an α value of 0.05. The best variant, and others that are not significantly different from it are indicated in bold face.

	TSP	
Time	ΔR_α	**Strategies** (ΔR)
		Size 500, Initial conditions RS
9.97	4.77	**PLS⟨OHI, ⊀, 1∗⟩**, PLS⟨RND, ⊀, ∗⟩ (69), *Dynagrid-HV* (147)
99.00	2.79	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (76), PLS⟨RND, ⊀, ∗⟩ (149)
999.07	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
10000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 500, Initial conditions TS
9.97	0	**PLS⟨OHI, ⊀, 1∗⟩**, *Dynagrid-HV* (75), PLS⟨RND, ⊀, ∗⟩ (150)
99.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
999.07	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
10000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 500, Initial conditions HQS
9.97	5.47	**PLS⟨OHI, ⊀, 1∗⟩**, *Dynagrid-HV* (67), PLS⟨RND, ⊀, ∗⟩ (146)
99.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
999.07	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
10000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 1000, Initial conditions RS
29.02	0	**PLS⟨OHI, ⊀, 1∗⟩**, PLS⟨RND, ⊀, ∗⟩ (75), *Dynagrid-HV* (150)
289.02	10.22	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (62), PLS⟨RND, ⊀, ∗⟩ (136)
3104.75	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
30000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 1000, Initial conditions TS
29.02	0	**PLS⟨OHI, ⊀, 1∗⟩**, *Dynagrid-HV* (75), PLS⟨RND, ⊀, ∗⟩ (150)
289.02	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
3104.75	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
30000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 1000, Initial conditions HQS
29.02	2.79	**PLS⟨OHI, ⊀, 1∗⟩**, *Dynagrid-HV* (76), PLS⟨RND, ⊀, ∗⟩ (149)
289.02	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
3104.75	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
30000.00	0	***Dynagrid-HV***, PLS⟨OHI, ⊀, 1∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)

	QAP	
Time	ΔR_α	**Strategies** (ΔR)
		Size 100, Correlation -0.75, Initial conditions RS
1.18	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
11.84	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
123.19	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
1200.00	18.722	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (48), PLS⟨RND, ⊀, ∗⟩ (96)
		Size 100, Correlation -0.75, Initial conditions TS
1.18	12.095	***Dynagrid-HV*****, PLS⟨OHI, ≻⊀, ∗⟩ (9)**, PLS⟨RND, ⊀, ∗⟩ (117)
11.84	3.966	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (74), PLS⟨RND, ⊀, ∗⟩ (148)
123.19	11.559	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (63), PLS⟨RND, ⊀, ∗⟩ (132)
1200.00	23.767	**PLS⟨OHI, ≻⊀, ∗⟩,** ***Dynagrid-HV*** **(15)**, PLS⟨RND, ⊀, ∗⟩ (33)
		Size 100, Correlation -0.75, Initial conditions HQS
1.18	11.937	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (15), PLS⟨RND, ⊀, ∗⟩ (120)
11.84	12.173	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (51), PLS⟨RND, ⊀, ∗⟩ (129)
123.19	16.328	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (63), PLS⟨RND, ⊀, ∗⟩ (111)
1200.00		p-value > alpha (no significant difference)
		Size 100, Correlation -0.5, Initial conditions RS
0.10	6.08	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (65), PLS⟨RND, ⊀, ∗⟩ (145)
1.00	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
10.02	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
100.00	18.65	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (57), PLS⟨RND, ⊀, ∗⟩ (96)
		Size 100, Correlation -0.5, Initial conditions TS
0.10	8.31	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (72), PLS⟨RND, ⊀, ∗⟩ (141)
1.00	11.87	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (17), PLS⟨RND, ⊀, ∗⟩ (121)
10.02	6.82	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (72), PLS⟨RND, ⊀, ∗⟩ (144)
100.00	18.51	**PLS⟨OHI, ≻⊀, ∗⟩,** ***Dynagrid-HV*** **(10)**, PLS⟨RND, ⊀, ∗⟩ (89)
		Size 100, Correlation -0.5, Initial conditions HQS
0.10	16.74	**PLS⟨OHI, ≻⊀, ∗⟩**, PLS⟨RND, ⊀, ∗⟩ (53), *Dynagrid-HV* (109)
1.00	9.82	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (61), PLS⟨RND, ⊀, ∗⟩ (137)
10.02	12.92	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (78), PLS⟨RND, ⊀, ∗⟩ (126)
100.00	21.68	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (53), PLS⟨RND, ⊀, ∗⟩ (64)
		Size 150, Correlation -0.75, Initial conditions RS
1.90	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
19.92	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
203.58	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
2000.00	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
		Size 150, Correlation -0.75, Initial conditions TS
1.90	9.75	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (45), PLS⟨RND, ⊀, ∗⟩ (135)
19.92	2.79	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (73), PLS⟨RND, ⊀, ∗⟩ (149)
203.58	5.56	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (76), PLS⟨RND, ⊀, ∗⟩ (146)
2000.00	7.92	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (57), PLS⟨RND, ⊀, ∗⟩ (141)
		Size 150, Correlation -0.75, Initial conditions HQS
1.90	14.52	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (38), PLS⟨RND, ⊀, ∗⟩ (118)
19.92	7.67	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (80), PLS⟨RND, ⊀, ∗⟩ (142)
203.58	6.73	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (78), PLS⟨RND, ⊀, ∗⟩ (144)
2000.00	6.08	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (65), PLS⟨RND, ⊀, ∗⟩ (145)
		Size 150, Correlation -0.5, Initial conditions RS
1.00	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
10.22	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
101.40	0	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (75), PLS⟨RND, ⊀, ∗⟩ (150)
1000.00	15.62	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (62), PLS⟨RND, ⊀, ∗⟩ (115)
		Size 150, Correlation -0.5, Initial conditions TS
1.00	11.6	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (23), PLS⟨RND, ⊀, ∗⟩ (124)
10.22	2.79	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (76), PLS⟨RND, ⊀, ∗⟩ (149)
101.40	5.47	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (79), PLS⟨RND, ⊀, ∗⟩ (146)
1000.00	17.67	***Dynagrid-HV*****, PLS⟨OHI, ≻⊀, ∗⟩ (15)**, PLS⟨RND, ⊀, ∗⟩ (96)
		Size 150, Correlation -0.5, Initial conditions HQS
1.00	16.06	***Dynagrid-HV*****, PLS⟨OHI, ≻⊀, ∗⟩ (7)**, PLS⟨RND, ⊀, ∗⟩ (101)
10.22	6.18	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (77), PLS⟨RND, ⊀, ∗⟩ (145)
101.40	6.73	***Dynagrid-HV***, PLS⟨OHI, ≻⊀, ∗⟩ (78), PLS⟨RND, ⊀, ∗⟩ (144)
1000.00	22.03	***Dynagrid-HV*****, PLS⟨OHI, ≻⊀, ∗⟩ (1)**, PLS⟨RND, ⊀, ∗⟩ (56)