Experiments on greedy and local search heuristics for d–dimensional hypervolume subset selection

Matthieu Basseur, Bilel Derbel, Adrien Goeffon, Arnaud Liefooghe

▶ **To cite this version:**

Matthieu Basseur, Bilel Derbel, Adrien Goeffon, Arnaud Liefooghe. Experiments on greedy and local search heuristics for d–dimensional hypervolume subset selection. Genetic and Evolutionary Computation Conference (GECCO 2016), 2016, Denver, United States. pp.541-548, 10.1145/2908812.2908949 . hal-01302283

HAL Id: hal-01302283
https://hal.science/hal-01302283

Submitted on 13 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Experiments on Greedy and Local Search Heuristics for d–dimensional Hypervolume Subset Selection

Matthieu Basseur
LERIA, Université d'Angers, France
matthieu.basseur@univ-angers.fr

Bilel Derbel
Univ. Lille, CNRS, UMR 9189 – CRIStAL, France
Dolphin, Inria Lille-Nord Europe, France
bilel.derbel@univ-lille1.fr

Adrien Goëffon
LERIA, Université d'Angers, France
adrien.goeffon@univ-angers.fr

Arnaud Liefooghe
Univ. Lille, CNRS, UMR 9189 – CRIStAL, France
Dolphin, Inria Lille-Nord Europe, France
arnaud.liefooghe@univ-lille1.fr

ABSTRACT

Subset selection constitutes an important stage of any evolutionary multiobjective optimization algorithm when truncating the current approximation set for the next iteration. This appears to be particularly challenging when the number of solutions to be removed is large, and when the approximation set contains many mutually non-dominating solutions. In particular, indicator-based strategies have been intensively used in recent years for that purpose. However, most solutions for the indicator-based subset selection problem are based on a very simple greedy backward elimination strategy. In this paper, we experiment additional heuristics that include a greedy forward selection and a greedy sequential insertion policies, a first-improvement hill-climbing local search, as well as combinations of those. We evaluate the effectiveness and the efficiency of such heuristics in order to maximize the enclosed hypervolume indicator of candidate subsets during a hypothetical evolutionary process, or as a post-processing phase. Our experimental analysis, conducted on randomly generated as well as structured two-, three- and four-objective mutually non-dominated sets, allows us to appreciate the benefit of these approaches in terms of quality, and to highlight some practical limitations and open challenges in terms of computational resources.

1. INTRODUCTION

The design and analysis of evolutionary multiobjective optimization (EMO) algorithms and other randomized search heuristics for multiobjective optimization is nowadays a well-established research area. These approaches can usually be viewed as an iterative process evolving a set of solutions, that constitutes the current Pareto set approximation, and selecting a representative subset that will take part in the next iteration. A main issue when designing such algorithms is how to decide on the solutions to be deleted and the ones to be kept at each iteration in order to maximize the overall approximation set quality at the selection or archiving stages of EMO algorithms. When the goal of the search process is defined in terms of a quality indicator, which assigns a scalar value that reflects the quality of any approximation set [20], a key issue of multiobjective optimizers is to iteratively maximize this indicator-value when extracting a limited number of solutions within the current approximation. It is well understood that computing such a subset is a challenging problem to be solved in practice. In addition, the corresponding indicator-based subset selection problem relevantly arises when a post-processing phase has to be designed, after a multiobjective optimizer had gathered a whole set of solutions from which one has to choose only a subset. In such a scenario, the goal is to select the best representative subset from all solutions recorded during the search process [9].

Among quality indicators, the hypervolume is more and more preferred when designing indicator-based EMO algorithms, but also when assessing the quality of the approximation set obtained at termination. This indicator has in fact some interesting properties such as the strict monotonicity property [22]. However, the computation of the hypervolume covered by a set is known to be $\#P$-complete with respect to the number of objective functions [7]. Generally speaking, a conventional approach consists in restricting the evaluation of the hypervolume for sets which only differ by one solution, consequently optimizing computations and reducing in practice the hypervolume computing cost by typically evaluating the hypervolume contribution of every single solution [4]. Nevertheless, this computation remains $\#P$-complete in the general case, even if it can be efficiently computed for problems with a small number of objectives [8].

As a consequence, the subset selection problem is a bi-partitioning problem in which the cost function is itself $\#P$-complete. When dealing with the hypervolume indicator, a common approach in EMO consists in applying a *Greedy Backward Elimination* (GBE) procedure, that removes the worst solution with respect to the hypervolume contribution, and iterates until the considered set shrinks to the target size; see e.g. [2, 21]. It is not difficult to see that the subset returned by such a heuristic procedure is generally

not optimal with respect to the problem under consideration. Alternatively, one can find other heuristics [5, 6], as well as exact methods [1, 8, 10, 14, 16] and approximation algorithms with performance guarantee [12, 13].

In this paper, we investigate the design of alternative heuristic procedures and consider to study empirically their relative performance under several configurations. The proposed heuristics can be classified into two categories. In the first one, we present two simple yet efficient greedy heuristics: *Greedy Forward Selection* (GFS) [6, 13] and *Greedy Sequential Insertion* (GSI). In the second category, we consider the application of a basic *local search* procedure (LS) similar to [6], a first-improvement hill-climbing, as a second stage that operates after an initial greedy or random stage. In order to study the performance of these heuristics, we propose to compete them in some mutually non-dominated sets in which various proportions of solutions have to be selected. More specifically, we use some known Pareto fronts from the literature as well as artificial empirical sets generated at random. Besides the quality of the subset approximation, our experimental analysis includes a study of the sensitivity of the proposed methods to several features such as the number of objectives, the number of solutions, and the size of the subset. Through an extensive experimental analysis, we are in particular able to highlight some improvements over the standard GBE heuristic, both in terms of computational cost and approximation quality. More importantly, our study provides new insights into the design of alternative subset selection heuristics, thus suggesting that more efficient and effective EMO algorithms could be designed in the future.

The remainder of the paper is organized as follows. In Section 2, we recall some definitions related to multiobjective optimization and we propose a formulation of the problem under consideration, before introducing greedy heuristics as well as a local search to solve it. In Sections 3 to 5, we provide a experimental analysis of the proposed methods on various sets of mutually non-dominated solutions. Finally, we provide some conclusions in Section 6.

2. INDICATOR-BASED SUBSET SELECTION

2.1 Problem Formulation

Let $Z \subseteq \mathbb{R}^d$ be a space of d-dimensional vectors, and let $\mathcal{I} : 2^Z \to \mathbb{R}_+$ be a unary quality indicator which assigns to any set of d-dimensional vectors a scalar value that reflects its quality. Without loss of generality, we assume this indicator to be maximized. Given $R \subseteq Z$ a reference set of cardinality n, and $k \leqslant n$ a positive subset size, the *indicator-based subset selection problem* (ISSP) consists in determining the set $S \subseteq R$ of maximal quality:

$$\underset{\substack{S \subseteq R \\ |S|=k}}{\arg\max}\, \mathcal{I}(S) \tag{1}$$

Let us remark that a binary quality indicator $\mathcal{I} : 2^Z \times 2^Z \to \mathbb{R}_+$ could be considered as well, simply by replacing $\mathcal{I}(S)$ by $\mathcal{I}(S, R)$. It is obvious to see that the solution space size associated with ISSP is $\binom{n}{k}$. Apart from the inherent complexity of computing indicator-values, this makes from ISSP a challenging problem from combinatorial optimization.

In the following, we focus on ISSP in the context of multiobjective optimization. Let us then recall some general background relative to multiobjective optimization in order to better illustrate the problem description, as well as our experimental setup.

2.2 ISSP in Multiobjective Optimization

A multiobjective optimization problem (MOP) is defined by a decision space X, and an objective function vector $f := (f_i)_{i \in [\![1,d]\!]}$ such that each objective $f_i : X \to \mathbb{R}$ is to be minimized. It shall be clear for the reader that we differentiate between the set of candidate subsets for ISSP, denoted as solution space, and the space of candidate solutions for the underlying MOP to be solved, denoted as decision space. $f[X] \subseteq \mathbb{R}^d$ is called the *objective space*. A d-dimensional vector $z := \{z_i\} \in \mathbb{R}^d$ *dominates* a vector $z' := \{z'_i\} \in \mathbb{R}^d$ ($z \succ z'$) iff $\forall i \in [\![1, d]\!], z_i \leqslant z'_i$ and $\exists i \in [\![1, d]\!], z_i < z'_i$. Similarly, $z \in \mathbb{R}^d$ *weakly dominates* $z' \in \mathbb{R}^d$ ($z \succeq z'$) iff $z \succ z'$ or $z = z'$. z and z' are *mutually non-dominated* iff $z \not\succ z'$ and $z' \not\succ z$. This dominance relation can be extended to solutions from the decision space: given two solutions $x, x' \in X$, $x \succ_{(\succeq)} x'$ iff $f(x) \succ_{(\succeq)} f(x')$. In the same way, a set S ($\in X$ or $\mathbb{R}^d$) strictly dominates a set S' iff $\forall x'_j \in S', \forall x_i \in S, x_i \succeq x'_j$ and $\exists x'_j \in S', \exists x_i \in S, x_i \succ x'_j$. Solving a multiobjective optimization problem (X, f) is to determine a *Pareto set* $X^\star = \{x \in X, \forall x' \in X, x' \not\succ x\}$. $X^\star$ is *minimal* if $\forall x, x' \in X^\star, f(x) \neq f(x')$; $Z^\star := f[X^\star]$ is called the *Pareto front*.

For difficult and intractable MOPs, heuristic approaches like EMO algorithms seek a good Pareto set approximation. To assess the performance of such approximation sets, a large number of (unary and binary) multiobjective set quality indicators, that assign a scalar value reflecting a given aspect of each approximation set quality, have been proposed in the literature [20, 22]. Importantly, a *Pareto-compliant* quality indicator $\mathcal{I}$ is strictly monotonic with respect to the Pareto dominance relation iff, $\forall S, S' \in 2^X, S \succ S' \Rightarrow \mathcal{I}(f[S]) > \mathcal{I}(f[S'])$. Such an indicator does not disagree with the (partial) order induced by the dominance relation [20]. We give below some examples of ISSP applications that find their root in multiobjective optimization and EMO techniques.

Example 1: Solving a combinatorial MOP (X, f), where X is a discrete set, can itself be formulated as the following variant of subset selection in the decision space, relatively to a Pareto-compliant quality indicator $\mathcal{I}$:

$$\underset{S \in \arg\max_{X^\star \subseteq X} \mathcal{I}(f[X^\star])}{\arg\min} |S| \tag{2}$$

Example 2: Finding the best Pareto front approximation of a predefined size k within an objective space Z is actually a (large-scale) subset selection problem (see Eq. (1)), with $R = Z$ and $\mathcal{I}$ Pareto-compliant.

Example 3: In a more practical way, indicator-based subset selection might occur during the resolution process of a MOP, with R being explicitly described and containing a reasonable number of mutually non-dominated points. In particular, EMO algorithms handle populations which represent sets containing individual solutions. Evolving a population by means of variation (crossover, mutation) or neighborhood operators implies to fix or restrict its size. Depending of the evolutionary process, a few or many new individuals are considered to integrate the population at each iteration, and selecting the subset of individuals defining the

next-step population is generally considered as a challenging issue. Given a population $P := \{x_i\}_{i \in [\![1,n]\!]} \subseteq X$, i.e. a set of individual solutions, let us denote $R := \{f(x_i)\}_{x_i \in P}$ its associated set of d-dimensional vectors in the objective space. Then, choosing k individuals among P with respect to a quality function $\mathcal{I}$ amounts to solve Problem (1). For instance, this ISSP application is investigated in [2].

Example 4: Similarly, an unbounded archive can easily be set up in order to record all solutions evaluated during any EMO search process, and is a common practice in the multi-objective literature. Obviously, the obtained archive cardinality can be large, and a possible post-processing procedure can actually compute a subset of a limited size maximizing an indicator-value, which corresponds to an ISSP. Its has been shown in [9] that such a post-processing phase allows for a significant improvement compared against the final population obtained by the corresponding EMO algorithm, without any overhead in terms of function evaluations.

In our experiments, we consider artificial scenarios mimicking the process of indicator-based subset selection that arises in the last couple of examples.

2.3 Hypervolume Quality Indicator

The hypervolume (hv) [22] is a unary quality indicator which gives the multidimensional volume of the portion of $\mathbb{R}^d$ which is weakly dominated by a set $S \subseteq Z$:

$$\mathsf{hv}(S) := \int_{z^{min}}^{z^{\max}} \alpha_S(z) dz$$

such that:

$$\alpha_S(z) := \begin{cases} 1 & \text{if } \exists s \in S \text{ such that } z \prec s \\ 0 & \text{otherwise} \end{cases}$$

In practice, for minimization problems, only the upper-bound $z^{\max} \in \mathbb{R}^d$ is required to compute the hypervolume. This parameter is called the *reference point*. A larger hypervolume-value implies a set of better quality. Note that generally, the hypervolume computation is considered over *approximation sets*, i.e. sets of mutually non-dominated elements of $\mathbb{R}^d$.

The hypervolume is the only known Pareto-compliant quality indicator [20]. However the hypervolume is parameter-dependent, since it is based on a reference point that must be specified by the practitioner. More importantly, the computational resources required to compute an indicator-value constitute and important feature of the indicator characteristics. The computational complexity of the hypervolume is exponential in the number of objectives; see e.g. [3, 11, 17].

2.4 Heuristics for ISSP

ISSP frequently occurs in multiobjective optimization, and is usually tackled using simple greedy techniques. Indeed, in many cases, a greedy backward elimination (GBE) procedure is used; see e.g. [2, 21]. In the following, we focus on solving ISSP as described previously and considering score values provided in terms of hypervolume. Four algorithms are considered in this paper and are described below. Computationally speaking, a subset solution to ISSP is represented as a binary string $x = (x_1, \ldots, x_i, \ldots, x_n)$ of size n, with $x_i = 1$ if the corresponding vector is included in the subset of selected elements, and $x_i = 0$ otherwise. Within all algorithms, ties are broken at random.

Algorithm 1 Greedy Backward Elimination (GBE)

$S \leftarrow R$
repeat
  $z^\star \leftarrow \arg\max_{z \in S} \ \mathsf{hv}(S \setminus \{z\})$
  $S \leftarrow S \setminus \{z^\star\}$
until $|S| = k$

Algorithm 2 Greedy Forward Selection (GFS)

$S \leftarrow \emptyset$
repeat
  $z^\star \leftarrow \arg\max_{z \in \{R \setminus S\}} \ \mathsf{hv}(S \cup \{z\})$
  $S \leftarrow S \cup \{z^\star\}$
until $|S| = k$

Algorithm 3 Greedy Sequential Insertion (GSI)

shuffle R such that $R = \{z_i\}_{i \in [\![1,n]\!]}$
$S \leftarrow \{z_i\}_{i \in [\![1,k]\!]}$
for $z \in \{z_i\}_{i \in [\![k+1,n]\!]}$ **do**
  $S \leftarrow S \cup \{z\}$
  $z^\star \leftarrow \arg\max_{z' \in S} \ \mathsf{hv}(S \setminus \{z'\})$
  $S \leftarrow S \setminus \{z^\star\}$
end for

Algorithm 4 First-improvement Local Search (LS)

start with a subset $S \subseteq R$ s. t. $|S| = k$
while $\exists z \in S,\ z' \in R \setminus S$ s.t. $\mathsf{hv}(S \setminus \{z\} \cup \{z'\}) > \mathsf{hv}(S)$ **do**
  $S \leftarrow S \setminus \{z\} \cup \{z'\}$
end while

GBE: The aforementioned greedy backward elimination GBE (Algorithm 1) starts with the complete initial set R, and the iterative process always removes the worst element in terms of hypervolume contribution, until k elements remain in the set [2, 21]. The *hypervolume contribution* of an element $s \in S$ corresponds to the difference $\mathsf{hv}(S) - \mathsf{hv}(S \setminus \{s\})$.

GFS: The greedy forward selection (GFS, Algorithm 2), originally proposed in [6, 13], starts with an empty set, and the iterative process always adds the best element in terms of hypervolume contribution, until k elements are selected. This approach is known to provide a $(1-1/e)-$approximation to the optimal subset [13].

GSI: The greedy sequential insertion (GSI, Algorithm 3) starts with a set S of k elements randomly selected from R. At each iteration, a remaining element from R is selected at random and added to S, and the element with the worst hypervolume contribution is deleted. Notice that the element that is to be removed can actually map to the most recently inserted one. This process iterates until all elements from R have been considered exactly once for integrating S.

LS: The first-improvement hill-climbing local search (LS, Algorithm 4) starts with a set of k elements from R, provided by some initialization process. At each step, LS seeks a pair of elements to be swapped, one being selected while the other being not, in order to improve the hypervolume value of the obtained subset. Hence, at each iteration, the number of possible swaps is $k \cdot (n-k)$. The local search continues until no swap can bring any hypervolume improvement. Notice

that a similar approach has been investigated in [5, 6].

3. EXPERIMENTAL SETUP

3.1 Competing Algorithms

The competing algorithms consist of all greedy heuristics presented in the previous section (GBE, GSI and GFS), from which we add the construction of a *random* subset RND, that we use as a baseline approach. Moreover, we measure the performance of the local search procedure LS depending on the starting solution (either random RND, or constructed by means of a greedy heuristic GBE, GSI, or GFS). The corresponding two-stage approaches are denoted as RND+LS, GBE+LS, GSI+LS, and GFS+LS, respectively. In the following, 8 variants of hypervolume subset selection heuristics are then investigated. Notice that there exists polynomial exact algorithms for the two-objective case [10, 14], which are not used for comparison since we focus on heuristics that can be applied to higher problem dimensions.

3.2 Problem Instances

In order to experiment those approaches, we consider two types of indicator-based subset selection problem instances.

Random instances: The first set of instances consists of reference sets containing randomly-generated mutually non-dominated vectors in $[0,1]^d$. Their structure can then potentially take any form within this hyper-box. The procedure is as follows. We first initialize an empty reference set $R := \emptyset$. Then, we iteratively generate a random objective vector z in $[0,1]^d$. If z does not dominate any point in R, and there does not exist any point in R that dominates z, then we add z to R ($R := R \cup \{z\}$). The procedure stops when the reference set reaches the expected cardinality, i.e. $|R| = n$. This set of instances share similarities with the ones from [5].

Structured instances: The second set of instances consist of reference sets from the CEC 2009 special session and competition on the performance assessment of multi-objective optimization algorithms [19]. The corresponding benchmark continuous functions have been specifically designed to resemble complicated real-life optimization problems, and their Pareto front presents different properties in terms of dimension and shape. We consider all the unconstrained functions UF01–10, with the exception of UF05 which is not relevant in our context, because it contains a very small number of points in the Pareto front. The objective space dimension is $d = 2$ for UF01–06 and $d = 3$ for UF07–10. The Pareto front from UF01, UF02 and UF03 is convex, the one from UF04, UF08 and UF10 is concave, and the one from UF06, UF07 and UF09 is a line or plane. In addition, there are gaps in the Pareto front of UF06 and UF09. Based on the code provided by the organizers at the following URL: http://dces.essex.ac.uk/staff/qzhang/moeacompetition09.htm, we generate a set of uniformly distributed points along the Pareto front in order to construct a reference set R for each function. Notice that for all problems, the reference set lies in $[0,1]^d$.

3.3 Parameter Setting and Implementation

For all the instances, we consider minimizing objectives, and we define the hypervolume indicator as the selection criterion. The hypervolume reference point is set to $z_i^{\max} = 1.1$, $i \in [\![1, d]\!]$. For random instances, we consider the following parameters: an objective space dimension $d \in \{2, 3, 4\}$, a reference set cardinality $n \in \{200, 500, 1000\}$, and a target subset size of $k \in \{0.1, 0.2, 0.5\} \cdot n$. A set of 10 instances is independently generated for each parameter combination $\langle d, n, k \rangle$. For structured instances, the objective space dimension is $d = 2$ for UF01–06, $d = 3$ for UF07–10, the reference set cardinality is $n = 1\,000$ for two-objective problems, $n = 2\,025$ for three-objective problems, and we investigate a target subset size of $k \in \{0.05, 0.1, 0.5\} \cdot n$ as well. As pointed out in [8], k is typically much smaller than n. Notice that a $k-$value of $k = 0.5 \cdot n$ allows to simulate an indicator-based subset selection problem arising at each iteration of a generational EMO search process, while lower $k-$values rather mimic a post-processing scenario, which would be based on the archive of all non-dominated solutions evaluated during an EMO search process. Overall, this leads to a total of 297 subset selection problem instances.

For each algorithm and each instance, 30 independent executions are performed. All algorithms have been executed under comparable conditions and share the same base components for a fair comparison. They have been implemented in C++, and compiled with g++ 4.8.4 using the -O3 compilation option. The hypervolume computation is based on [3], for which the worst-case complexity is $O(\ell^{d-2} \log \ell)$, where ℓ is the number of points in the set. The corresponding implementation is available at the following URL: http://lopez-ibanez.eu/hypervolume/. The experimental analysis has been conducted in R [15], using the ggplot2 [18] package.

4. RESULTS ON RANDOM INSTANCES

Our experimental analysis on random instances is two-fold. We first investigate the performance of competing algorithms in terms of approximation quality. To this end, we compute the relative deviation of the output of each algorithm $\mathsf{hv}(A)$ with respect to the best-found hypervolume value $\mathsf{hv}^\star$ for the instance under consideration as follows: $(\mathsf{hv}^\star - \mathsf{hv}(A))/\mathsf{hv}^\star$. As a consequence, a lower value is better. Next, we focus on the algorithms running time, measured both in terms of CPU time, and in terms of the number of calls performed on the hypervolume calculation function, which can be considered as a bottleneck of all hypervolume-based subset selection approaches.

4.1 Approximation Quality

Figure 1 reports, for each heuristic, the relative deviation to the best-found hypervolume for the instance under consideration, with respect to the problem dimension d, the reference set size n, and the target subset size k. Let us start with the performance of stand-alone greedy heuristics *without* any local search performed (in red color on the figure). First of all, selecting a subset of cardinality k at random, as done with RND, obviously lead to a poor strategy compared against any other competing heuristic. Second, the conventional greedy heuristic based on a backward selection strategy GBE, which is used in many indicator-based EMO algorithms [2, 21], is far from being the most effective approach for all subset selection problem instances. Interestingly, this suggests that other simple heuristics can actually provide an attractive alternative to enhance the performance of indicator-based EMO algorithms for some instances. For example, for two-objective instances, the GSI heuristic, that sequentially inserts solutions in a random order with imme-

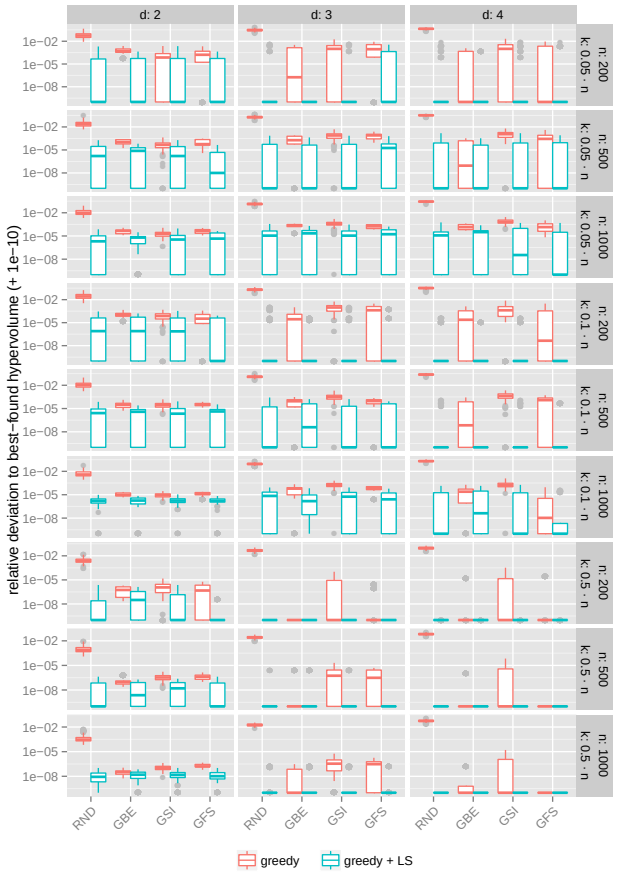

Figure 1: Approximation quality of each heuristic for random instances, given in terms of relative hypervolume deviation (in log-scale, lower is better).

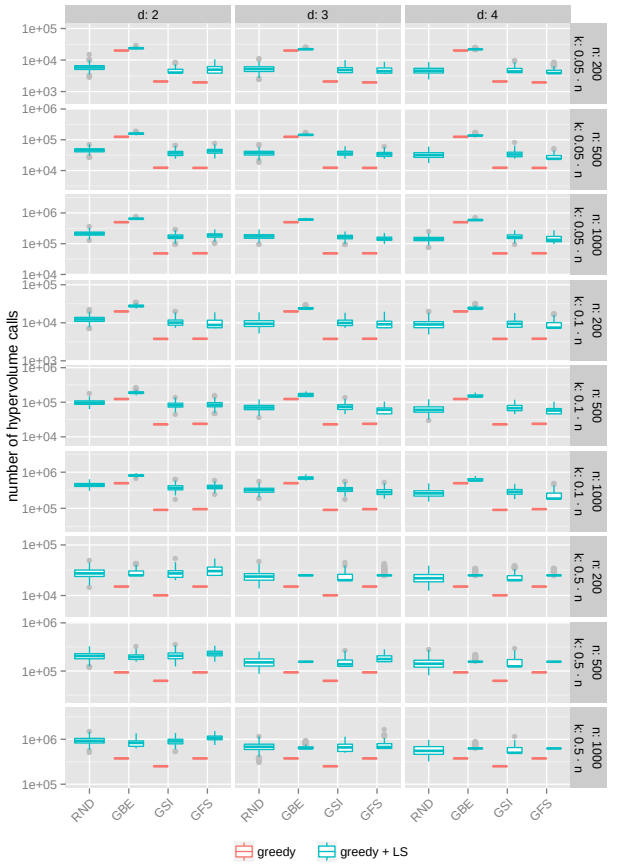

Figure 2: Number of hypervolume calls (in log-scale) performed by each heuristic for random instances.

diate truncation of the worst-contributing solution, is actually better than GBE for a small $k-$value, while being worst for a larger k. However, for $d \in \{3, 4\}$, the performance of GBE and GSI can hardly be distinguished. Next, the approach based on forward selection GFS that iteratively inserts the best-fit solution to the subset is never outperforming the other greedy heuristics for two and three objectives, except for $d = 2$, $n = 200$ and $k = 20$ where it is slightly better. However, for $d = 4$, GFS appears to be especially effective, particularly for $n = 200$ and $k = 20$, as well as for $n = 1000$ and $k \in \{50, 100\}$. To summarize, for greedy heuristics, the overall ranking is as follows: GBE $>$ GFS $>$ GSI for all the random instances we experimented. Notice, however, that this ranking is actually reversed for $d = 2$.

If we now focus on the local search heuristic (in green color on the figure), our experiments clearly show that, independently of the starting solution, LS is significantly better than any standalone greedy heuristic for almost all random instances, and is actually never outperformed by any of them (the performance is comparable in some cases, with $d \in \{3, 4\}$ and $k = 0.5 \cdot n$). In fact, all LS variants are always very close to the best-found hypervolume, although more fluctuations appear as n and k grow. This means that LS accurately identifies high-quality subsets independently of the problem dimension. However, no matter if a greedy heuristic is used to initialize the starting solution of LS or not, and which greedy heuristic is used, its performance remains nearly unchanged. Indeed, the distribution of hypervolume values for different LS settings overlaps for almost any random instance.

4.2 Computational Cost

Figure 2 reports the number hypervolume computations required by each algorithm to terminate. Indeed, this is an essential aspect to measure the efficiency of subset selection approaches, the hypervolume calculation being particularly heavy computationally speaking. However, the computational complexity is not only exponentially impacted by the problem dimension d, but it is polynomially affected by the number points in the set from which the indicator-value is sought. Hence, a hypervolume call for GFS, which manipulates increasing-size subsets from cardinality 1 to k, is less expensive than a hypervolume call for GBE, which manipulates decreasing-size subsets from cardinality $n-1$ to k. As a consequence, the number of hypervolume calls only partially define the running time of the competing approaches. This is the reason why we also report the CPU time required by each heuristic in Figure 3. For the sake of generality and in order to ease the deployment of our thorough experiments, our implementation is the same for all dimensions d, the running time of all greedy approaches can however be definitely improved for fixed and smaller dimensions, for instance by using the pre-computations from [13].

Overall, the computational cost of all approaches seems to increase exponentially with the objective space dimension d

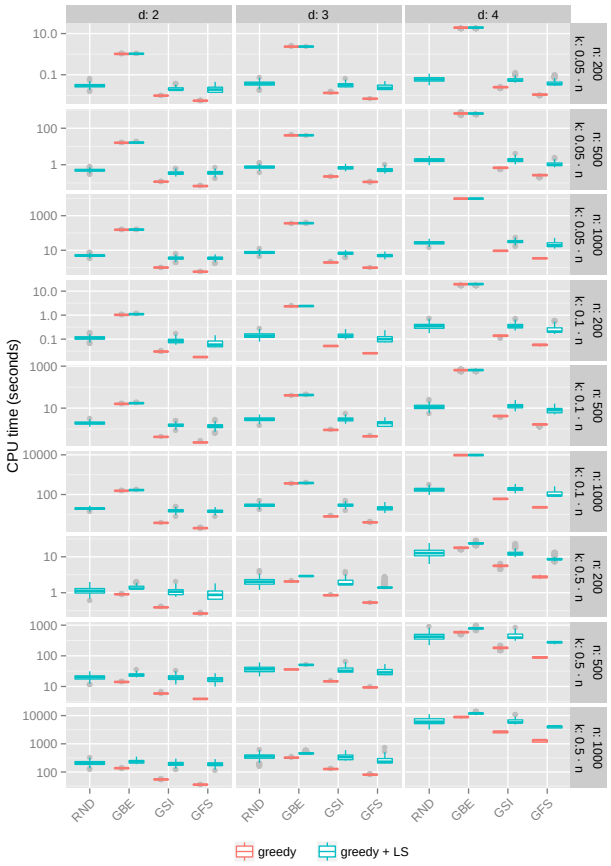

Figure 3: CPU time (in log-scale) required by each heuristic for random instances.

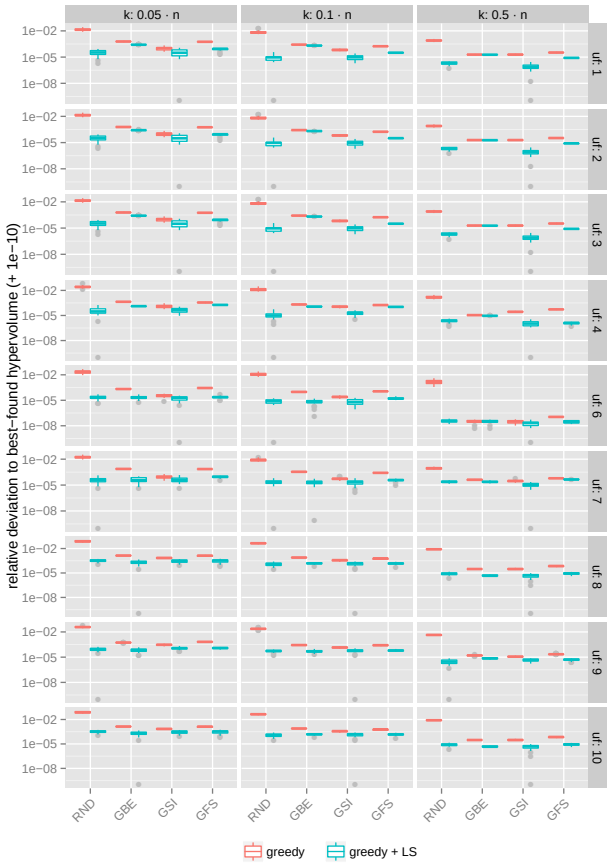

Figure 4: Approximation quality of each heuristic for structured instances, given in terms of relative hypervolume deviation (in log-scale, lower is better).

and the reference set size n, and to increase linearly with the target subset size k. The GSI and GFS greedy heuristics are the less computationally demanding heuristics. GFS requires more hypervolume calls than GSI as n and k grow, but since each hypervolume call is cheaper for GFS than for GSI, the former is actually faster in terms of CPU time. On the contrary, GBE is much more computationally demanding than other stand-alone heuristics, in terms hypervolume calls and even more in terms of CPU time. In fact, GBE is usually more costly than any LS variant, except of course the one that is seeded with GBE. The trend for the efficiency of LS heuristics roughly correspond to the following ranking: GFS+LS > GSI+LS > LS > GBE+LS for all the random instances we experimented. This means that, apart from GBE, initializing the search process of LS with a greedy heuristic (GSI or GFS) actually allows speeding up the identification of a local optima, independently of its obtained quality. More importantly, the small overhead implied by LS when performed after a greedy heuristic comes with a significant improvement in terms of solution quality.

5. RESULTS ON STRUCTURED INSTANCES

In this section, we focus on structured instances, based on the reference sets from the CEC 2009 competition [19].

5.1 Approximation Quality

Figure 4 reports the approximation quality obtained by greedy and local search heuristics for structured instances. First, GSI is clearly outperforming other stand-alone greedy heuristics for this set of instances. Indeed, it is never outperformed by GBE nor GFS, except for UF04 when $k = 0.5 \cdot n$. However, GBE obtains the same performance than GSI for $k = 0.05 \cdot n$, it is slightly worse for $k = 0.1 \cdot n$ (except for three-objective instances UF08–10), and it is better for $k = 0.5 \cdot n$. At last, GFS is never outperforming GBE nor GSI. Overall, for structured instances, the ranking of stand-alone greedy heuristics is: GSI > GBE > GFS. This ranking is different compared to random instances, where GSI was the best-performing greedy heuristic only for $d = 2$, but the worst one for other $d-$values.

With respect to local search heuristics, any LS variant is always outperforming standalone greedy heuristics. Furthermore, running LS on the solution obtained by any greedy heuristic invariably allow for a significant improvement. For UF01–04, GSI+LS is slightly better than RND+LS, and both approaches are better than GBE+LS and GFS+LS. For other instances, there is no clear difference between the different LS strategies, although GBE+LS tends to be better for small k, but not anymore for larger $k-$values. Overall, GFS+LS appears to be the less efficient LS approach. As a consequence, initializing the solution from which the local search is started at random or with the GSI heuristic allows to obtain a sound subset in terms of hypervolume, while using GBE or GFS as a seeding strategy tends to make the local search being trapped in worse-quality local optima.

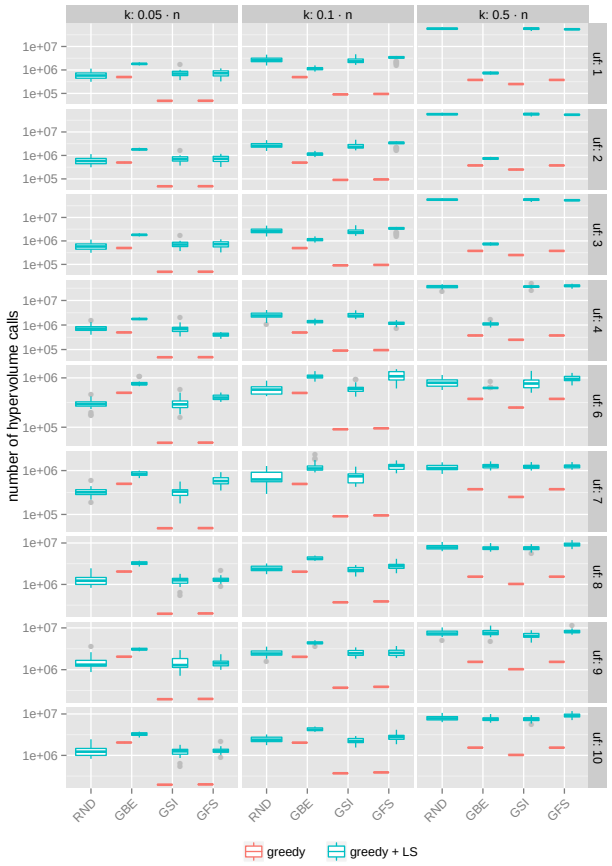

Figure 5: Number of hypervolume calls (in log-scale) performed by each heuristic for structured instances.

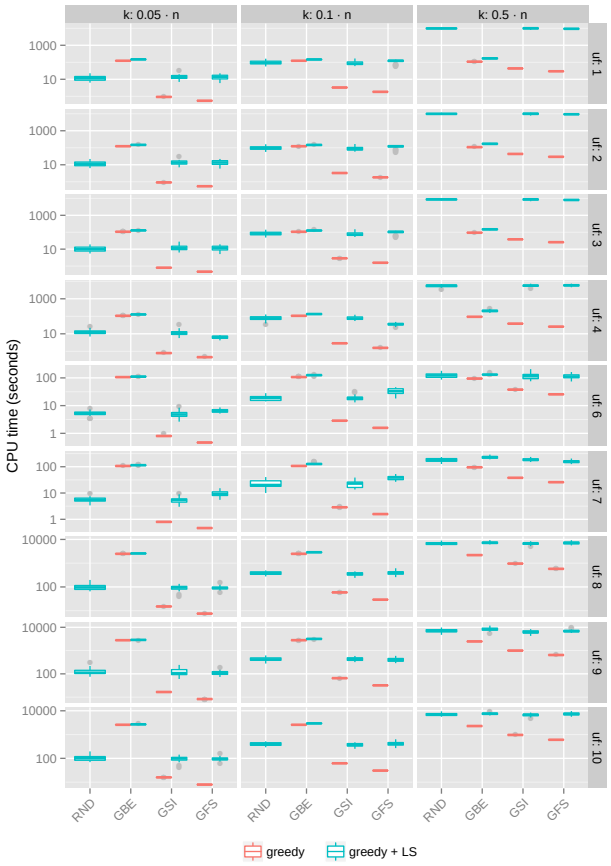

Figure 6: CPU time (in log-scale) required by each heuristic for structured instances.

5.2 Computational Cost

Figures 5 and 6 respectively report the number of hypervolume calls and the CPU time required by each greedy and local search heuristic for structured instances. As for random instances, GBE, which is the suggested approach in many indicator-based EMO algorithms [2, 21], is invariably the most costly stand-alone greedy approach: not only it makes more calls to the hypervolume function, but each hypervolume computation is actually more demanding in terms of computational time. The number of hypervolume calls is roughly the same for GSI and GFS, except for $k = 0.5 \cdot n$, where GFS needs additional ones. However, in terms of CPU time, GFS is clearly faster than any other greedy heuristic.

As for local search, independently of the initialization strategy, the number of hypervolume calls is roughly the same. GBE+LS tends to be more demanding for $k = 0.05 \cdot n$, and less demanding for larger $k-$values and UF01–04 (i.e. reference sets from two-objective Pareto fronts without any gap). This means that, apart from GBE+LS, the number of LS steps remains almost unchanged whatever the starting solution. The same conclusions can be drawn in terms of CPU time, except that GBE+LS is now clearly more demanding for $k < 0.5 \cdot n$. Interestingly, LS is also more effective than GBE alone for many instances. Overall, this makes from RND+LS and GSI+LS the most effective and efficient local search variants for structured instances.

6. CONCLUSIONS

In this paper, we investigated different greedy and local search heuristics for indicator-based subset selection. This problem constitutes an important challenge in multiobjective optimization, not only during the survival selection or archiving phase of indicator-based EMO algorithms, but as a post-processing step as well. More particularly, our empirical study investigates the effectiveness and the efficiency of three greedy heuristics, a local search algorithm, and combinations of those on random and structured instances. Our experimental findings reveal important information on the performance of these different algorithmic solutions to the hypervolume-based subset selection problem. Indeed, the performance of the greedy backward elimination heuristic, which is commonly employed in practice, can be outperformed by local searches, without necessarily requiring more computation resources. As well, this conventional procedure may be improved by a greedy sequential insertion heuristic with immediate truncation, especially on two-objective structured instances. As a consequence, within an hypervolume-based EMO search process, instead of merging the parent and the offspring populations and then iteratively removing the less contributing solution, it might actually be more effective to add offspring solutions one by one while truncating the population to its expected size after each insertion. This might be one of the reasons why steady-state indicator-based EMO algorithms tend to be favored over generational ones [4]. In addition, allowing a

reasonable computational overhead by running a simple local search process, initialized with a solution obtained from a greedy heuristic, constantly leads to an improvement in terms of hypervolume, independently of the actual initialization strategy.

In the future, we plan to measure the hypervolume cumulative improvement within the EMO search process, and to extend our experimental analysis, especially with respect to lower bounds [14], exact or approximation methods [8, 10, 12, 14], and to implementations using more sophisticated data structures [13]. In fact, the computational cost of subset selection shall be explicitly related to the overall computational cost of the EMO search process it operates with. Indeed, one can consider different scenarios where the complexity of subset selection is negligible compared against the cost of the evaluation function from the multiobjective optimization problem to be solved. In such a case, the hypervolume, or any other indicator's computational cost shall not be an issue for designing a subset selection algorithm, i.e. only quality matters. On the contrary, if the problem at hand is not expensive, then the computational cost induced by several hypervolume calls shall be taken into account more carefully when deriving a subset selection approach in the course of an EMO search process, or even as a post-processing procedure.

Acknowledgements

This work is partially supported by the PGMO program from the *Fondation Mathématique Jacques Hadamard*.

7. REFERENCES

[1] J. Bader. *Hypervolume-based search for multiobjective optimization: theory and methods*. PhD thesis, ETH Zurich, Switzerland, 2010.

[2] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[3] N. Beume, C. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

[4] 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.

[5] L. Bradstreet, L. Barone, and L. While. Maximising hypervolume for selection in multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2006)*, pages 1744–1751. IEEE, 2006.

[6] L. Bradstreet, L. While, and L. Barone. Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2007)*, pages 3203–3210. IEEE, 2007.

[7] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry*, 43:601–610, 2010.

[8] K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18(3):383–402, 2010.

[9] K. Bringmann, T. Friedrich, and P. Klitzke. Generic postprocessing via subset selection for hypervolume and epsilon-indicator. In *Parallel Problem Solving from Nature (PPSN XIII)*, volume 8672 of *Lecture Notes in Computer Science*, pages 518–527. Springer, 2014.

[10] K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Genetic and Evolutionary Computation Conference (GECCO 2014)*, pages 589–596. ACM, 2014.

[11] T. Chan. Klee's measure problem made easy. In *Foundations of Computer Science (FOCS 2013)*, pages 410–419. IEEE, 2013.

[12] T. Friedrich and F. Neumann. Maximizing submodular functions under matroid constraints by multi-objective evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN XIII)*, volume 8672 of *Lecture Notes in Computer Science*, pages 922–931. Springer, 2014.

[13] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. Greedy hypervolume subset selection in the three-objective case. In *Conference on Genetic and Evolutionary Computation (GECCO 2015)*, pages 671–678. ACM, 2015.

[14] T. Kuhn, C. M.Fonseca, L. Paquete, S.Ruzika, M. Duarte, and J. R. Figueira. Hypervolume subset selection in two dimensions: formulations and algorithms. *Evolutionary Computation*, 2015. (early access).

[15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[16] D. Vaz, L. Paquete, C. M. Fonseca, K. Klamroth, and M. Stiglmayr. Representation of the non-dominated set in biobjective discrete optimization. *Computers & Operations Research*, 63:172–186, 2015.

[17] L. While. A new analysis of the LebMeasure algorithm for calculating hypervolume. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *Lecture Notes in Computer Science*, pages 326–340. Springer, 2005.

[18] H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer, New York, USA, 2009.

[19] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the CEC 2009 special session and competition. Working Report CES-887, University of Essex, 2008.

[20] E. Zitzler, J. Knowles, and L. Thiele. Quality assessment of Pareto set approximations. In *Multiobjective optimization – interactive and evolutionary approaches*, volume 5252 of *Lecture Notes in Computer Science*, chapter 14, pages 373–404. Springer, 2008.

[21] E. Zitzler, L. Thiele, and J. Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79, 2010.

[22] 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.