RSG, a Method for Pareto Front Approximation and Reference Set Generation

Angel E. Rodriguez-Fernandez *, Hao Wang *, Oliver Schütze

Posted Date: 3 April 2025

doi: 10.20944/preprints202504.0294.v1

Keywords: multi-objective optimization; Pareto front approximation; performance indicators; benchmarking

 doi:10.20944/preprints202504.0294.v1

Article

RSG, a Method for Pareto Front Approximation and Reference Set Generation

Angel E. Rodriguez-Fernandez [1,*], Hao Wang [2] and Oliver Schütze [1]

[1] Centro de Investigación y de Estudios Avanzados del IPN, Departmento de Computación, Mexico City, Mexico
[2] Leiden Institute of Advanced Computer Science and Applied Quantum Algorithms, Leiden University, Leiden, The Netherlands
* Correspondence: angel.rodriguez@,schuetze@cs

Abstract: In this paper, we address the problem of obtaining bias-free and complete finite size approximations of the solution sets (Pareto fronts) of multi-objective optimization problems (MOPs). Such approximations are, in particular, required for the fair usage of distance-based performance indicators, which are frequently used in evolutionary multi-objective optimization (EMO). If the Pareto front approximations are biased or incomplete, the use of these performance indicators can lead to misleading or false information. To address this issue, we propose the Reference Set Generator (RSG), which can, in principle, be applied to Pareto fronts of any shape and dimension. We finally demonstrate the strength of the novel approach on several benchmark problems.

Keywords: multi-objective optimization; Pareto front approximation; performance indicators; benchmarking

1. Introduction

Multi-objective optimization has become an integral part of the decision-making for many real-world problems. In a multi-objective optimization problem (MOP), one is faced with the issue of concurrently optimizing k individual objectives. The set of optimal solutions is called the Pareto set. The image of the Pareto set is called the Pareto front. The latter set is, in many cases, most important for the decision maker (DM) since it provides him/her with an overview of the optimal performances for his/her project. What makes MOPs hard to deal with is that one can expect that both Pareto set and front form – at least locally and under certain assumptions on the model – objects of dimension $k-1$ ([1]). For the numerical treatment of MOPs, specialized evolutionary algorithms, called multi-objective evolutionary algorithms (MOEAs), have caught the interest of many researchers and practitioners during the last three decades ([2]). MOEAs are population-based and hence allow for the obtaining of a finite approximation of the entire solution set in one run of the algorithm. For the performance assessment of the outcome sets, several different indicators have been proposed so far (e.g., [3–8]). Some of these performance indicators are distance-based and require a "suitable" finite-size representation of the Pareto front. While until now, a vast variety of different MOEAs has been proposed and analyzed, it is fair to say that the generation of suitable reference sets has played a rather minor role in the evolutionary multi-objective optimization (EMO) community. It is evident that such reference sets should be complete. Further, as we show in this work, a biased representation can lead to misleading or even false information.

To fill this gap, we propose in this work the Reference Set Generator (RSG). The main steps of RSG are as follows: (i) a first approximation A_y of the Pareto front is either taken or generated. This set can be, in principle, of arbitrary size, and the elements can be non-uniformly distributed along the Pareto front (i.e., biased). However, all of these elements have to be "close enough" to the set of interest. In order to obtain a bias-free approximation, (ii) component detection and (iii) a filling step is applied to A_y. Finally, (iv) a reduction step is applied. The RSG is applicable to MOPs with Pareto fronts of, in principle, any shape and dimension. Further, the resulting reference set is of any desired magnitude.

 doi:10.20944/preprints202504.0294.v1

We will show the strength of the novel method on several benchmark problems.

The remainder of this paper is organized as follows: in Section 2, we briefly recall the background for the understanding of this work. We further discuss the related work and the performance indicators that benefit from our approach. In Section 3, we first motivate the need for bias-free complete finite-size Pareto front approximations, and propose the Reference Set Generator (RSG). In Section 4, we present some numerical results on selected benchmark problems and compare the RSG to related algorithms. Finally, we draw our conclusions in Section 5 and give possible paths for future research.

2. Background and Related Work

We consider multi-objective optimization problems (MOPs) that can be mathematically expressed via

$$\min_{x \in Q} \; F(x). \tag{MOP}$$

Hereby, the map F is defined as

$$F : Q \to \mathbb{R}^k, \qquad F(x) = (f_1(x), \ldots, f_k(x))^T, \tag{1}$$

where we assume each of the individual objectives $f_i : Q \to \mathbb{R}$, $i = 1 \ldots, k$, to be continuous. We stress, however, that the method we propose in the sequel, RSG, can, in principle, also be applied to discrete problems. Q is the domain of the objective functions that is typically expressed by equality and inequality constraints.

In order to define the optimality of an MOP, one can use the concept of *dominance* [9].

Definition 1. *(a)* *Let $v, w \in \mathbb{R}^k$. Then the vector v is* less than *w ($v <_p w$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$. The relation $\leq_p$ is defined analogously.*

(b) *$y \in Q$ is* dominated *by a point $x \in Q$ ($x \prec y$) with respect to (MOP) if $F(x) \leq_p F(y)$ and $F(x) \neq F(y)$.*

(c) *$x \in Q$ is called a* Pareto point *or* Pareto optimal *if there is no $y \in Q$ that dominates x.*

(d) *The set P_Q of Pareto optimal solutions*

$$P_Q := \{x \in Q \; : \; \nexists y \in Q \; s.t. \; y \prec x\} \tag{2}$$

is called the Pareto set.

(e) *The image $F(P_Q)$ of the Pareto set is called the* Pareto front.

One can expect that both the Pareto set and the Pareto front form under certain conditions, at least locally objects of dimension $k - 1$. For details, we refer to [1]. Due to this "curse of dimensionality" It is, hence, not possible for an evolutionary multi-objective optimization algorithm (MOEA) to keep all promising candidate solutions (e.g., all non-dominated ones) during the algorithm run. It is, hence inevitable – at least for continuous problems – to select which of the promising solutions should be kept in order to obtain a "suitable" approximation of the solution set (in most cases, the Pareto front of the given MOP). Within MOEAs, this process is termed "selection". Another term, which can be used synonymously, is "archiving". The latter is typically used when the MOEA is equipped with an external archive.

Most of the existing MOEAs can be divided into three main classes: (a) MOEAs that are based on the concept of dominance (e.g., [10–13]), (b) MOEAs that are based on decompositions (e.g., [14–18]), and (c) MOEAs that make use of an indicator function (e.g., [19–23]). The selection strategies of the first generation of MOEAs of class (a) are based on a combination of non-dominated sorting and niching (e.g., [24–26]). Later, elite preservation was included, leading to an increased overall performance (and, as a consequence, better Pareto front approximations). This holds, e.g., for SPEA ([27]), PAES ([13]), SPEA-II ([11]), and NSGA-II ([10]). Theoretical studies on selection mechanisms have been done by

 doi:10.20944/preprints202504.0294.v1

the groups of Rudolph ([28–31]) and Hanne ([32–35]). All of these selection mechanisms deal with the abilities of the populations to reach the Pareto set/front. On the other hand, the distributions of the individuals along the Pareto sets/fronts have not been considered. The selection mechanisms of the MOEAs within classes (b) and (c) follow directly from the construction of the algorithms: selection in a MOEA of class (b) is done by considering the values of the chosen scalarization functions. Analogously, the selection in a MOEA of class (c) is done by considering the indicator contributions.

Existing (external) archiving strategies can also be divided into three classes: (a) unbounded archivers, (b) implicitly bounded archivers, and (c) bounded archivers. Unbounded archivers store all promising solutions during the algorithm run. The magnitudes of such archives can exceed any given threshold if the algorithm is run long enough. Unbounded archivers have, e.g., been used and analyzed in [36–42]. ϵ-dominance ([43]) can be viewed as a weaker concept of dominance. This relation allows single solutions to "cove" entire parts of the Pareto front of a given MOP, which is the basis for most implicitly bounded archivers. Such strategies have first been considered in the context of evolutionary multi-objective optimization (EMO) by Laumanns et al. ([44]). Later works proposed and analyzed different approximations (such as gap-free approximations of the Pareto front) and deal with different sets of interest (e.g., the consideration of all nearly optimal or all ϵ-locally optimal solutions), ([40,41,45–50]). Finally, bounded archivers have, e.g., been proposed in [51,52] where adaptive grid selections have been utilized. Bounded archivers tailored to the use of the Hypervolume indicator have been suggested in [53,54]). Laumanns and Zenklusen have proposed two bounded archivers that aim for ϵ-approximations of the Pareto front ([55]).

All of the selection/archiving strategies mentioned above have in common that they aim for a "best approximation" of the set of interest out of the given (finite) data. A related but slightly different problem is to generate a "suitable" (in particular complete and bias-free) finite size approximation of the set of interest S for the sake of comparisons, even if S is known approximately or even analytically but not "trivia" (e.g., a line segment). Tian et al. [56] describe the procedure of PlatEMO [57], which uses uniform sampling on the simplex and then maps these points to the particular Pareto front of each problem. However, an analytical expression or characterization of the Pareto front is required, and in some cases, the obtained set is not completely uniform. A method related to RSG can be found in [58] which has the aim of guiding the iterates of a particular Newton method toward the Pareto front. In this work, we extend this idea for the purpose of generating complete and bias-free Pareto front approximations of relatively large magnitudes (in particular compared to population sizes used in EMO).

Finally, reference sets as the ones generated by RSG are helpful for the evaluation of the performance qualities of candidate sets (populations) in EMO. More precisely, such sets are required for all distance-based indicators. The earliest such indicators are the Generational Distance (GD, [3]) and the Inverted Generational Distance. (IGD, [5]). Later, the indicator Δ_p ([6,59]) has been proposed that is a combination of slight variants of GD and IGD, and that can be viewed as an averaged version of the Hausdorff distance d_H. So far, there exist several extensions of these performance indicators. For instance, the consideration of continuous sets – either only the Pareto front or also the candidate solution set – has been done in [8,60] leading to modifications of IGD and Δ_p. The indicators IGD+ ([7]) and DOA ([61]) are modifications of IGD that are Pareto compliant.

3. Reference Set Generator (RSG)

In this section, we first motivate the need for complete and bias-free finite size Pareto front approximations, and then propose the Reference Set Generator (RSG) that targets for such sets.

3.1. Motivation

Distance-based indicators require a "suitable" finite-size approximation of the Pareto front in order to give a "correct" value for the approximation quality of the considered candidate solution set. This holds in particular for the above-mentioned indicators GD, IGD, and Δ_p, together with their variants. Such representations are ideally uniformly spread along the entire Pareto front ([60]). This,

 doi:10.20944/preprints202504.0294.v1

however, is a non-trivial task unless the Pareto front is given analytically and has a relatively simple form (e.g., linear or perfectly spherical). Regrettably, this is the case for only a few test problems (e.g., DTLZ1 and DTLZ2). On the other hand, there exist quite a few benchmark MOPs where the shape of the Pareto *set* is relatively simple. For such problems, it is tempting to choose uniform samples from the Pareto set (i.e., $X = \{s_1, \ldots, s_m\}$, where all $s_i \in P_Q$), and to use the respective image $Y = F(X)$ to represent of the Pareto front. The following discussion shows, however, that this approach has to be handled with care since it can induce unwanted biases in the approximations that, in turn, may result in misleading indicator values.

As the first example, consider the one-dimensional bi-objective problem

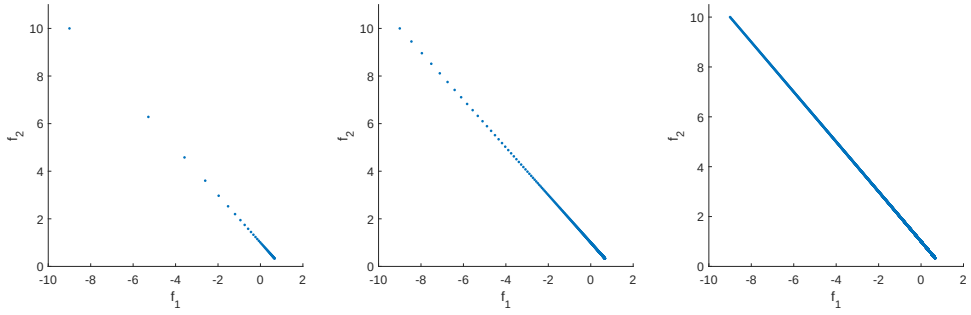
$$F(x) = \begin{pmatrix} 1 - \frac{1}{x} \\ \frac{1}{x} \end{pmatrix}. \tag{3}$$

Let the domain be given by $Q = [0.1, 3]$, then the Pareto set is identical to Q, and the Pareto front is the line segment that connects the points $a = (-9, 10)^\top$ and $b = (2/3, 1/3)^\top$. Figure 1 shows the result when using N equally spaced points along the Pareto set. As it can be seen for $N = 50$ and $N = 500$, there is a clear bias of the images toward the right lower end of the Pareto front. For $N = 10,000$, the Pareto front approximation is "complete" (at least from the practical point of view) and appears to be perfect. However, it possesses the same bias. To see the impact of the reference set on the performance indicators consider the two hypothetical outcomes (e.g., possible results from different MOP solvers):

$$\begin{aligned} A &:= \left\{ a + \frac{2i-1}{10}(b-a) \mid i \in \{1, \ldots, 5\} \right\}, \text{ and} \\ B &:= \left\{ a + \frac{i}{10}(b-a) \mid i \in \{6, \ldots, 10\} \right\}. \end{aligned} \tag{4}$$

Figure 2 shows the two sets together with the Pareto front. Note that A is the perfect 5-element approximation of the Pareto front: the elements are equally distributed along the Pareto front, and the extreme points are shifted “half-way in” ([60]). The set B is certainly not perfect as it, e.g., misses to "cover" more than half of the front. Table 1 shows values $I(O, R)$ for different distance-based indicators, the outcomes $O \in \{A, B\}$, and different representations R of the Pareto front. For $R = R^x_{10,000}$ (the one shown in Fig. 1 (c)), all indicators – except $d_H = \Delta_\infty$ – yield lower values for B than for A, indicating (erroneously) that B is better than A. This is not the case for d_H since the Hausdorff distance is determined by the maximum of the considered distances and not by an average of those (however, d_H has other disadvantages in the context of EMO, most prominently that it punishes single outliers [6]). The situation changes when selecting $R = R^y_{10,000}$ as a representation of the front. This representation also contains $N = 10,000$ elements, but these are chosen uniformly along the Pareto *front*. Now, there is a tie for the two GD variants, and for all other indicators, A leads to better values than B. These values are indeed very close to the ‘correct’ values: all exact GD values are equal to zero since A and B are contained in the Pareto front. In order to compute the exact IGD values, a particular integral has to be solved [60]. When using R^y_{10000} as representation, the computation of the IGD values can be interpreted as a Riemann sum with $N = 10,000$ equally sized sub-intervals leading to perfect solutions, at least from the practical point of view.

 doi:10.20944/preprints202504.0294.v1



(**a**) N=50 (**b**) N=500 (**c**) N=10,000

Figure 1. Pareto front representations of MOP (3) when using N equally distributed samples along the Pareto set.

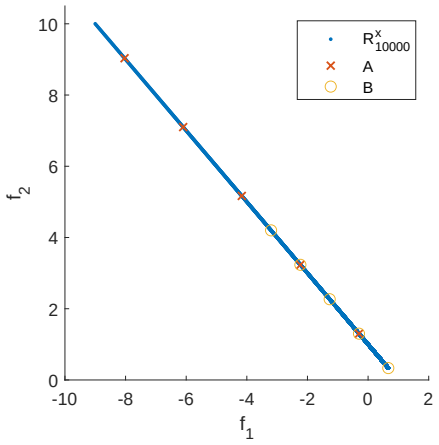


Figure 2. Representation R^x_{10000} of the Pareto front of MOP (3) together with the two hypothetical outcomes A and B.

Table 1. Indicator values $I(O, R)$ for different indicators, the outcomes $O \in \{A, B\}$, and the representations $R \in \{R^x_{100}, R^y_{100}, R^x_{10,000}, R^y_{10,000}\}$ of the Pareto front.

	GD_1	GD_2	IGD_1	IGD_2	$IGD+$	Δ_1	Δ_2	Δ_∞
$I(A, R^x_{100})$	0.5118	0.7384	0.9084	**0.9873**	0.6423	0.9084	**0.9873**	**1.3671**
$I(B, R^x_{100})$	**0.0698**	**0.1002**	**0.4522**	1.0744	**0.3198**	**0.4522**	1.0744	8.2024
$I(A, R^y_{100})$	**0.0684**	**0.0684**	**0.6835**	**0.7883**	**0.4833**	**0.6835**	**0.7883**	**1.2987**
$I(B, R^y_{100})$	**0.0684**	**0.0684**	2.5974	3.6765	1.8367	2.5974	3.6765	8.1341
$I(A, R^x_{10,000})$	0.0028	0.0032	0.8968	0.9776	0.6341	0.8968	0.9776	**1.3671**
$I(B, R^x_{10,000})$	**0.0008**	**0.0010**	**0.4117**	**0.8792**	**0.2911**	**0.4117**	**0.8792**	8.2024
$I(A, R^y_{10,000})$	**0.0007**	**0.0007**	**0.6835**	**0.7893**	**0.4833**	**0.6835**	**0.7893**	**1.3664**
$I(B, R^y_{10,000})$	**0.0007**	**0.0007**	2.5974	3.6767	1.8367	2.5974	3.6767	8.2018

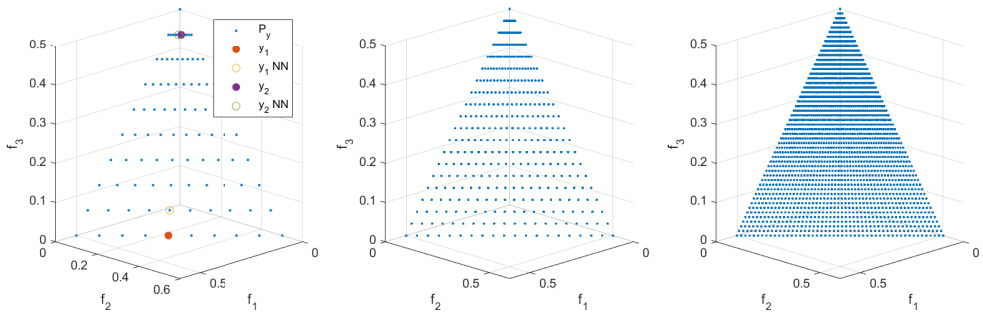
We repeat the process, but now using only $N = 100$ elements for the representation (see Table 1). We can see the same trend, i.e., that B appears to be better for R^x_{100} while A appears to be better when using R^y_{100}. Further, we see that the indicator values for R^y_{100} are already quite close to the exact values (i.e., when using R^y_{10000}). While the proper choice of N may not be an issue for bi-objective problems ($k = 2$), this may get important for a larger number of objectives due to the "curse of dimensionality": at least for continuous problems, one can expect that the Pareto front forms under certain (mild) assumptions a manifold of dimension $k - 1$ ([1]).

We next consider further test problems. Figures 4 to 6 show the Pareto front approximations for the commonly used test problems DTLZ1, ZDT1, and ZDT3, respectively, where the representation has been obtained via uniform sampling in decision variable space along the Pareto set. In all

 doi:10.20944/preprints202504.0294.v1

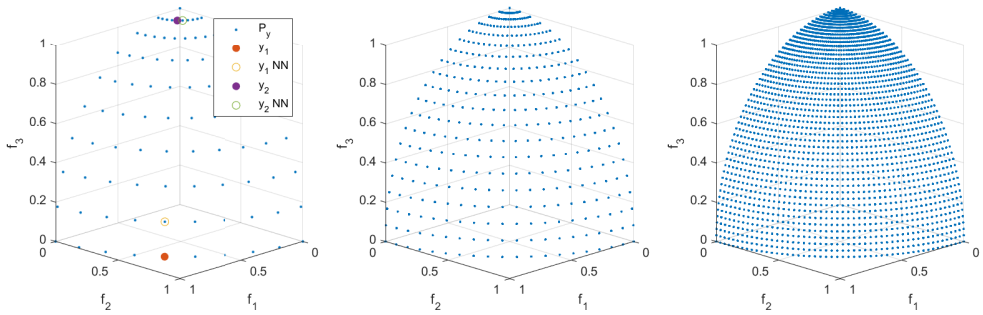
cases, the Pareto front representations appear to be perfect for large enough values of N. However, certain biases can be observed for lower values of N. For all test problems, we selected two points out of the representation and showed their nearest neighbors. These values differ by one order of magnitude which confirms that the solutions are not uniformly distributed along the fronts. Using such representations, the same issues can arise as discussed above.

To conclude, the suitable representation of the Pareto front of a given MOP is crucial when considering distance-based performance indicators that use an average of the distances considered. Such representations are ideally equally distributed over the front. If the representation contains a bias, this may result in misleading indicator values, leading, in turn, to a wrong evaluation of the obtained results. In particular, the approach to performing the sampling along the Pareto set is, though tempting, not appropriate for such indicator-based indicators. In the sequel, we will propose a method that aims to achieve a uniform Pareto front representations.



(**a**) N=100 (**b**) N=400 (**c**) N=2500

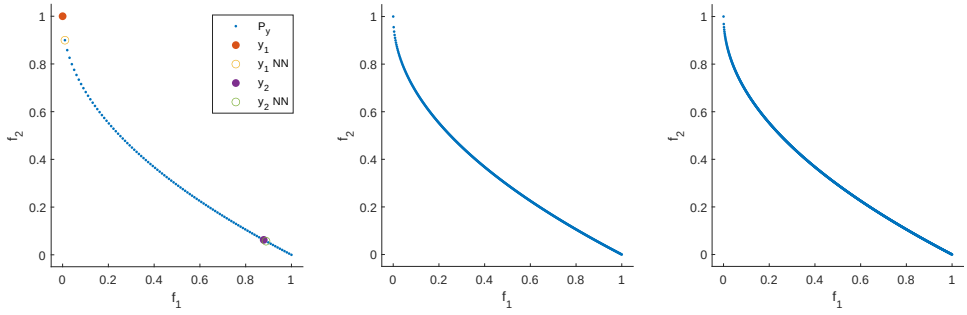
Figure 3. Distribution resulting from uniformly sampling the Pareto set of *DTLZ1* with N points. In (a) $d(y_1, NN_{y_1}) = 0.068181240846835$, $d(y_2, NN_{y_2}) = 0.008729713347982$.



(**a**) N=100 (**b**) N=400 (**c**) N=2500

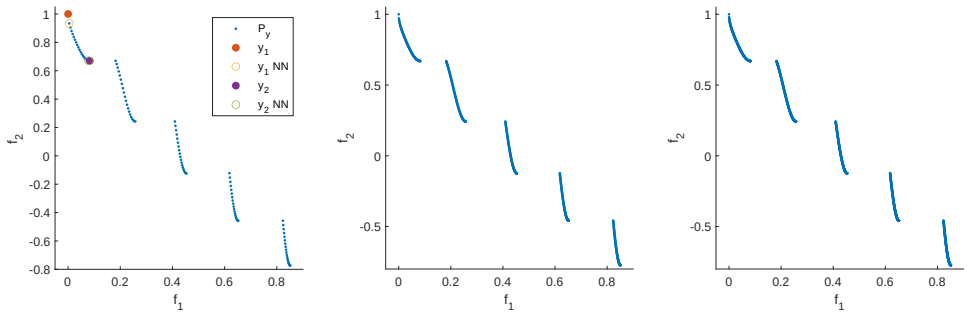
Figure 4. Distribution resulting from uniformly sampling the Pareto set of *DTLZ2* with N points. In (a) $d(y_1, NN_{y_1}) = 0.174311485495316$, $d(y_2, NN_{y_2}) = 0.030268871802677$.

 doi:10.20944/preprints202504.0294.v1



(**a**) N=50 (**b**) N=500 (**c**) N=1000

Figure 5. Distribution resulting from uniformly sampling the Pareto set of *ZDT1* with N points. In (a) $d(y_1, NN_{y_1}) = 0.101010101010101$, $d(y_2, NN_{y_2}) = 0.011440746020984$.



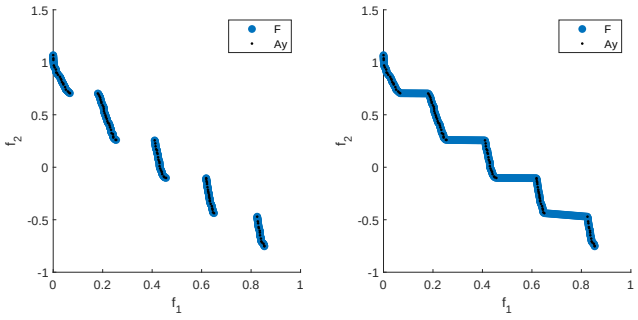
(**a**) N=50 (**b**) N=500 (**c**) N=1000

Figure 6. Distribution resulting from uniformly sampling the Pareto set of *ZDT3* with N points. In (a) $d(y_1, NN_{y_1}) = 0.066834581606972$, $d(y_2, NN_{y_2}) = 0.004486990395919$.

3.2. RSG

In the following, we assume we are interested in a Pareto front approximation of size N for a given MOP. Further, we assume that we are given a set A_y of (in principle) arbitrary size ℓ of non-dominated, possibly non-uniformly distributed points that are "close enough" to the PF. Below, we will discuss different strategies to obtain A_y. Given this data, the Reference Set Generation (RSG) process consists of three main steps: *component detection*, *filling*, and *reduction*. The idea is to *fill* the gaps between the points within each connected point of A_y. This leads to a more complete set F with a higher cardinality than A_y, which can then be *reduced* to obtain a uniform reference set of size N. In general, PFs can be disconnected, and if we simply fill the gaps in A_y, we may introduce points that do not belong to the PF (Figure 7(**b**)). Therefore, *component detection* must be performed before applying the filling process to each detected component (Figure 7(**a**)).

 doi:10.20944/preprints202504.0294.v1



(**a**) (**b**)

Figure 7. Filling with (a) and without (b) component detection for ZDT3. Note that points not in the PF are included if the component detection step is omitted (b).

The general procedure of RSG is presented in Algorithm 1, and each of the main steps will be explained in the following subsections. Figures 8, 12, 9, and 13 illustrate all the steps for the problems ZDT3, CONV3-4, DTLZ7 and CONV4-2F. It is important to note that the cases for $k = 2$ and $k \geq 3$ objectives exhibit slight variations in the *component detection* step and follow completely different procedures for the *filling* step.

Algorithm 1 Reference Set Generation (RSG)

Input: Starting set A_y, filling size N_f, output size N.
Output: PF reference Z.

$C = \{C^1, \ldots, C^{nc}\} \leftarrow$ ComponentDetection(A_y)
for $i = 1, \ldots, nc$ **do** ▷ for each component
    $F^i \leftarrow$ Filling($C^i, \lceil \frac{N_f \cdot |C^i|}{|A_y|} \rceil$)
end for
$F \leftarrow \bigcup_i F^i$
$Z \leftarrow$ reduction(F, N)
return Z

3.3. Component Detection

Since the PF might be disconnected, we first apply a component detection on A_y. We use DBSCAN [62] in the objective space for this purpose for three main reasons: (i) the number of components does not need to be known a priori, (ii) the method detects outliers, and (iii) we have observed that a density-based approach works better than a distance-based one (e.g., k-means) for component detection. DBSCAN has two parameters: $minpts$ and r. To make the component detection process "parameter-free", we compute a small grid-search to find the optimal values of $minpts$ and r based on the *weakest link* function defined in [63]. The default parameter values are $minpts \in \{2, 3\}$ and $r \in \{0.10\bar{d}, 0.11\bar{d}, \ldots, 0.15\bar{d}\}$ for bi-objective problems, and $minpts \in \{3, 4\}$ and $r \in \{0.19\bar{d}, 0.20\bar{d}, \ldots, 0.23\bar{d}\}$ otherwise, where $\bar{d}$ is the average pairwise distance between all points. However, these parameters can be adjusted when calling RSG. A summary of the component detection process is presented in Algorithm 2. In the following, we describe the remaining steps for a *single* connected component. If multiple components exist, the procedures must be repeated analogously for each component C^i identified by Algorithm 2.

3.4. Filling

Even if we know the PS a priori, a uniform sampling of the PS will not result in a uniform sampling of the PF. We assume that we have a set of points A_y that is not uniformly distributed. However, if we fill the gaps and select N points from the filled set, we can obtain a more uniform set, leading to better IGD approximations when selecting points from these filled sets. The idea behind the filling step is to create a set that is as uniform as possible, so that the reduction step (in particular,

 doi:10.20944/preprints202504.0294.v1

Algorithm 2 Component Detection

Input: starting set $A_y = \{p_1, \ldots, p_\ell\}$, number of objectives k
Output: Number of clusters nc, clusters $C = \{C^1, \ldots, C^{nc}\}$.

Set $\bar{d} \leftarrow \frac{2}{\ell(\ell-1)} \sum_{p_i \neq p_j} |p_i - p_j|$
if $k = 2$ **then**
  Set $r\_range \leftarrow \{0.1\bar{d}, 0.11\bar{d}, \ldots, 0.16\bar{d}\}$
  Set $minpts\_range \leftarrow \{2, 3\}$
else
  Set $r\_range \leftarrow \{0.19\bar{d}, 0.20\bar{d}, \ldots, 0.23\bar{d}\}$
  Set $minpts\_range \leftarrow \{3, 4\}$
end if
Set $wlmin \leftarrow \infty$
for $minpts$ **in** $minpts\_range$ **do**
  for r **in** r_range **do**
    $C_t, nc_t \leftarrow \text{DBSCAN}(A_y, r, minpts)$
    $wl \leftarrow \text{WeakestLink}(C_t)$
    if $wl \leq wlmin$ **then**
      $C \leftarrow C_t$
      $nc \leftarrow nc_t$
      $wlmin \leftarrow wl$
    end if
  end for
end for
return $C = \{C^1, \ldots, C^{nc}\}, nc$

k-means) does not get stuck in non-uniform local optima, which would lead to non-uniform final sets. The next task is, therefore, to compute N_f solutions that are ideally uniformly distributed along A_y.

This process is performed differently for $k = 2$ and $k \geq 3$ objectives:

- For $k = 2$, we sort the points of $A_y = \{p_1, \ldots, p_\ell\}$ in increasing order of f_1, i.e., the first objective. Then, we consider the piecewise linear curve formed by the segments between p_1 and p_2, p_2 and p_3, and so on. The total length of this curve is given by $|L| = \sum_{i=1}^{\ell-1} |L_i|$, where $|L_i| = \|p_i - p_{i+1}\|_2$. To perform the filling, we arrange the N_f desired points along the curve L such that the first point is p_1 and the subsequent points are distributed equidistantly along L. This is achieved by placing each point at a distance of $\delta_\ell = |L|/(N_f - 1)$ from the previous one along L. See Algorithm 3 for details.
- The filling process for $k \geq 3$ consists of several intermediate steps that must be described first; see Algorithm 4 for a general outline of the procedure. The procedure is as follows: First, to better represent A_y (particularly for the filling step), we triangulate this set in $k-1$ dimensional space. This is done because the PF for continuous MOPs forms a set whose dimension is at most $k-1$. To achieve this, we compute a "normal vector" η to A_y using equation 8, and then we project it onto the $k-1$ hyperplane normal to η, obtaining the projected set P_{k-1}. After this, we compute the Delaunay triangulation [64] of P_{k-1}, which provides a triangulation T' that can be used in the original k-dimensional space. For concave PFs, the triangulation may include triangles (or simplex for $k > 3$) that extend beyond A_y (Figure 12(d)), so a removal strategy is applied to eliminate these triangles and obtain the final triangulation T. Finally, each triangle $t_i \in T$ is uniformly filled at random with a number of points proportional to its area (or volume for $k > 3$), resulting in the filled set F of size N_f.

  We will now describe each step in more detail in the following:

 doi:10.20944/preprints202504.0294.v1

- *Computing "normal vector"* (η). Since the front is not known, we compute the normal direction η orthogonal to the convex hull defined by the minimal elements of A_y. More precisely, we compute η as follows: if $A_y = \{p_1, \ldots, p_\ell\}$, choose

$$y^m_{(i)} \in \arg\min_{j=1,\ldots,N'} p_{j,i}, \quad i = 1, \ldots, k, \tag{5}$$

where $p_{j,i}$ denotes the i-th element of p_j, and set

$$M := (y^m_{(2)} - y^m_{(1)}, y^m_{(3)} - y^m_{(1)}, \ldots, y^m_{(k)} - y^m_{(1)}) \in \mathbb{R}^{k\times(k-1)}. \tag{6}$$

Next, compute a QR-factorization of M, i.e.,

$$M = QR = (q_1, \ldots, q_k)R, \tag{7}$$

where $Q \in \mathbb{R}^{k\times k}$ is an orthogonal matrix with column vectors q_i, and $R \in \mathbb{R}^{k\times(k-1)}$ is a right upper triangular matrix. Then, the vector

$$\eta = -\mathrm{sgn}(q_{k,1})\frac{q_k}{\|q_k\|_2} \tag{8}$$

is the desired shifting direction. Since Q is orthogonal, the vectors $v_1 := q_1, \ldots, v_{k-1} := q_{k-1}$ form an orthonormal basis of the hyperplane that is orthogonal to η. That is, these vectors can be used for the construction of P_{k-1}.

- $k-1$ *Projection* (P_{k-1}). We use η as the first axis of a new coordinate system $(\eta, v_1, \ldots, v_{k-1})$, where the vectors v_i are defined as above. In this coordinate system, the orthonormal vectors $v_1, \ldots, v_{k-1}$ form the basis of a hyperplane orthogonal to η. The projection of A_y onto this hyperplane (P_{k-1}) is achieved by first expressing A_y in this new coordinate system as $P = \beta\eta + \beta_1 v_1 + \cdots + \beta_{k-1}v_{k-1}$, and then removing the first coordinate, yielding $P_{k-1} = \beta_1 v_1 + \cdots + \beta_{k-1}v_{k-1}$.
- *Delaunay Triangulation* (T'). Compute the Delaunay triangulation of P_{k-1}. This returns T', a list of size δ' containing the indices of P_{k-1} that form the triangles (or simplices for $k > 3$). The list T' serves as the triangulation for the k-dimensional set A_y, which is possible because T' consists of indices, making it independent of the dimension. We use δ' to denote the number of triangles obtained, $\{T'(i,1), \ldots, T'(i,k)\}$ the indices of the vertices forming triangle i and $\{p_{T'(i,1)}, \ldots, p_{T'(i,k)}\}$ to denote the corresponding vertices of triangle i.
- *Triangle Cleaning* (T). We identify three types of unwanted triangles: those with large sides, those with large areas, and those where the matrix containing the coordinates of the vertices has a large condition number. The type of cleaning applied depends on the problem; however, the procedure remains the same for any problematic triangle case and is outlined in Algorithm 5. First, the property ρ_i (area, largest side, or condition number) is computed for all the triangles $i = 1, \ldots, \delta$. Next, triangles i with $\rho_i > \tau$ are removed. The parameter τ is also problem-dependent, and the specific values used for each problem will be detailed in the results section.
- *Triangle Filling* (F). For each triangle $t_i \in T$ with area a_i, we generate $\lceil \frac{a_i}{A} N_f \rceil$ points uniformly at random inside triangle t_i, following the procedure described in [65]. That is, the number of points is proportional to the area (or volume) of each triangle (or simplex). Here, $A = \sum_{i=1}^{\delta} a_i$ is the total area of the triangulation.

3.5. Reduction

Once we have computed the filled set F, we need to select N points that are ideally evenly distributed along F. To this end, we use k-means clustering with N clusters, as there is a strong relationship between k-means and the optimal IGD subset selection ([66,67]). The resulting N cluster

 doi:10.20944/preprints202504.0294.v1

Algorithm 3 Filling ($k = 2$ Objectives)

Input: starting set $A_y = \{p_1, \ldots, p_\ell\}$, filling size N_f
Output: Filled set $F = \{y_1, \ldots, y_{N_f}\}$

$X = \{x_1, \ldots, x_\ell\} \leftarrow$ sort A_y according to its first objective f_1
$L_i \leftarrow \|x_i - x_{i+1}\| \quad \forall i = 1, \ldots, \ell - 1$
$L \leftarrow \sum_{i=1}^{\ell-1} L_i$
$\delta_\ell \leftarrow L/(N_f - 1)$
$dist\_left \leftarrow (0, 0, \ldots, 0) \in \mathbb{R}^\ell$
for $i = 1 : \ell - 1$ **do** ▷ compute number of points per segment
    $ratio \leftarrow (|L_i| + dist\_left(i))/\delta_\ell$
    $points\_per\_segment(i) \leftarrow \lfloor ratio \rfloor$
    $dist\_left(i+1) \leftarrow (ratio - \lfloor ratio \rfloor) \cdot \delta_\ell$
end for
$count = 1$
for $i = 1 : \ell - 1$ **do** ▷ for each line segment
    if $points\_per\_segment(i) > 0$ **then** ▷ check if a point lands in segment L_i
        $v_i := (x_{i+1} - x_i)/L_i$
        $y_{count} = x_i + (\delta_\ell - dist\_left(i)) \cdot v_i$
        $count \leftarrow count + 1$
        for $j = 2 : points\_per\_segment(i)$ **do** ▷ if L_i has more than one point
            $y_{count} \leftarrow y_{count-1} + \delta_\ell \cdot v_i$
            $count \leftarrow count + 1$
        end for
    end if
end for
return $F = \{y_1, \ldots, y_{N_f}\}$

Algorithm 4 Filling ($k \geq 3$ Objectives)

Input: starting set $A_y = \{p_1, \ldots, p_\ell\}$, filling size N_f
Output: Filled set $F = \{y_1, \ldots, y_{N_f}\}$

$\eta \leftarrow$ normal_vector(A_y)
$P_{k-1} \leftarrow$ projection(A_y, η)
$T' \leftarrow$ DelaunayTriangulation(P_{k-1})
$T \leftarrow$ TriangleCleaning(T')
$F \leftarrow$ TriangleFilling(T, A_y)
return $F = \{y_1, \ldots, y_{N_f}\}$

 doi:10.20944/preprints202504.0294.v1

Algorithm 5 Triangle Cleaning

Input: starting set $A_y = \{p_1, \ldots, p_l\}$, triangulation $T' \in \mathbb{R}^{k \times \delta'}$, number of triangles δ', parameter threshold τ

Output: cleaned triangulation T of size δ

if chosen property is area **then**

$$\rho_i \leftarrow \tfrac{1}{k!} \det \left[\begin{pmatrix} p^T_{T'(i,2)} - p^T_{T'(i,1)} \\ p^T_{T'(i,3)} - p^T_{T'(i,1)} \\ \vdots \\ p^T_{T'(i,k)} - p^T_{T'(i,1)} \end{pmatrix} \begin{pmatrix} p^T_{T'(i,2)} - p^T_{T'(i,1)} \\ p^T_{T'(i,3)} - p^T_{T'(i,1)} \\ \vdots \\ p^T_{T'(i,k)} - p^T_{T'(i,1)} \end{pmatrix}^T \right]^{1/2} \qquad \forall i = 1, \ldots, \delta'$$

else if chosen property is largest side **then**

    $\rho_i \leftarrow$ largest side of simplex with vertices $\{p_{T'(i,1)}, \ldots, p_{T'(i,k)}\}$

else if chosen property is condition number **then**

$$\rho_i \leftarrow \kappa \begin{pmatrix} p_{T'(i,1)} \\ p_{T'(i,2)} \\ \vdots \\ p_{T'(i,k)} \end{pmatrix}$$

▷ $\kappa(A)$ is the condition number of matrix A

end if

$\rho \leftarrow \sum_{i=1}^{\delta'} \rho_i$

$T \leftarrow T'$

if $\rho_i > \tau \cdot \rho$ **then**

    remove triangle i from T

end if

return T, number of triangles δ

centroids form the PF reference set Z. Note that this reduction method can be modified if needed to obtain an ideal reference according to other types of indicators (i.e., those that are not distance-based).

3.6. Obtaining A_y

RSG requires an initial approximation A_y of the Pareto front. Note that by construction of the algorithm, this set can have small imperfections (which can be removed by the filling step) and can also have biases in the approximation (reduction step). However, it is desired that A_y "captures" the shape of the entire Pareto front. The computation of such an approximation is certainly problem-dependent. For our computations, we have used the following three main procedures to get A_y:

sampling: For some benchmark problems, either the Pareto set or the Pareto front is given in analytic form. If a sampling can be performed in objective space (e.g., for linear fronts, the remaining steps of the RSG may not be needed to further improve the quality of the solution set. If the sampling is performed in decision variable space, the elements of the resulting image A_y may not be uniformly distributed along the Pareto front as discussed above. However, in that case, the filling and reduction step will help to remove biases.

archiving: The result of an MOEA or any other MOP solver can, of course, be taken. This could be either the final archive of the population, via merging several populations of the same or several runs ([58]), or via using external (unbounded) archives ([50]). Note that this includes taking a reference set from a given repository. We have used archiving, e.g., for the test problems WFG3-9, DTLZ1-4, DTLZ7, ZDT1-6, CONV3, CONV3-4, and CONV4-2F.

continuation: An alternative to the above mentioned techniques is to make use of multi-objective continuation methods, probably in combination with the use of several different starting points. In particular, we have used the Pareto Tracer (PT, [68–70]), a state-of-the-art continuation method that is able to treat problems of in principle any dimensions (both n and k), can handle general constraints and that can even detect local degeneration of the solution set. We have used PT, e.g., for the test problem WFG1, WFG2, DTLZ5, and DTLZ6.

 doi:10.20944/preprints202504.0294.v1

3.7. Complexity Analysis

The overall complexity is $\mathcal{O}(\gamma\ell^2 + \tau_2 NkN_f)$ for $k = 2$ (regardless of the number of components), and $\mathcal{O}(\gamma\ell^2 + \delta k^2 + \tau_2 NkN_f)$ for $k \geq 3$, where ℓ is the size of the initial approximation A_y, k is the number of objectives, δ is the number of triangles in the Delaunay triangulation, τ_2 is the number of iterations of k-means (bounded to 500 in this work), N is the desired size of the reference set Z, and N_f is the size of the filling. This assumes $\ell \gg k$ and that the triangle-cleaning method used is based on the longest side (which was the method applied to all the references presented in this work). Typically, obtaining a decent approximation requires a large value of N_f, making the clustering step the dominant one and thus reducing the overall complexity to $\mathcal{O}(\tau_2 NkN_f)$ for any k. We now present the complexity analysis in detail for each step separately, considering a single component. We now present the complexity analysis in detail for each step separately, considering a single component.

- Component Detection. The time complexity is $\mathcal{O}((\ell)^2 + \gamma(\ell \log(\ell) + (\ell)^2))$ which accounts for the computation of the average distance, plus the size of the grid search (γ) multiplied by the sum of the complexities of DBSCAN and the WeakestLink computation. Here, ℓ is the size of A_y, and γ represents the number of parameter combinations of the grid search, with $\gamma = 14$ for $k = 2$ and $\gamma = 10$ for $k \geq 3$ using the default values. If it is previously known that the Pareto front is connected, then the parameters of DBSCAN can be correctly adjusted, and γ can be set to 1.
- Filling. The time complexity depends on the number of objectives:
  - For $k = 2$ the time complexity is $\mathcal{O}(\ell + kN_f)$, which accounts for sorting and placing the N_f points along the line segments.
  - For $k \geq 3$ the time complexity is $\mathcal{O}([k\ell + k(k-1)^2] + \ell k^2 + \ell \log \ell + [(k-1)\delta + kN_f])$ due to the computations involved in determining the normal vector η, changing coordinates and projecting, performing the Delaunay triangulation, and filling the triangles. Here, δ represents the size of the cleaned Delaunay triangulation, i.e., the number of triangles. Additionally, triangle cleaning must be considered, though its complexity depends on the method used. It is given by $\mathcal{O}(\delta k^3)$ when cleaning is based on area or the condition number (due to determinant computation), or $\mathcal{O}(\delta k^2)$ when cleaning based on the longest side.
- Select Reference Set T. The time complexity is $\mathcal{O}(\tau_2 NkN_f)$ due to the k-means clustering algorithm. Here, τ_2 is the number of iterations of k-means.

4. Results

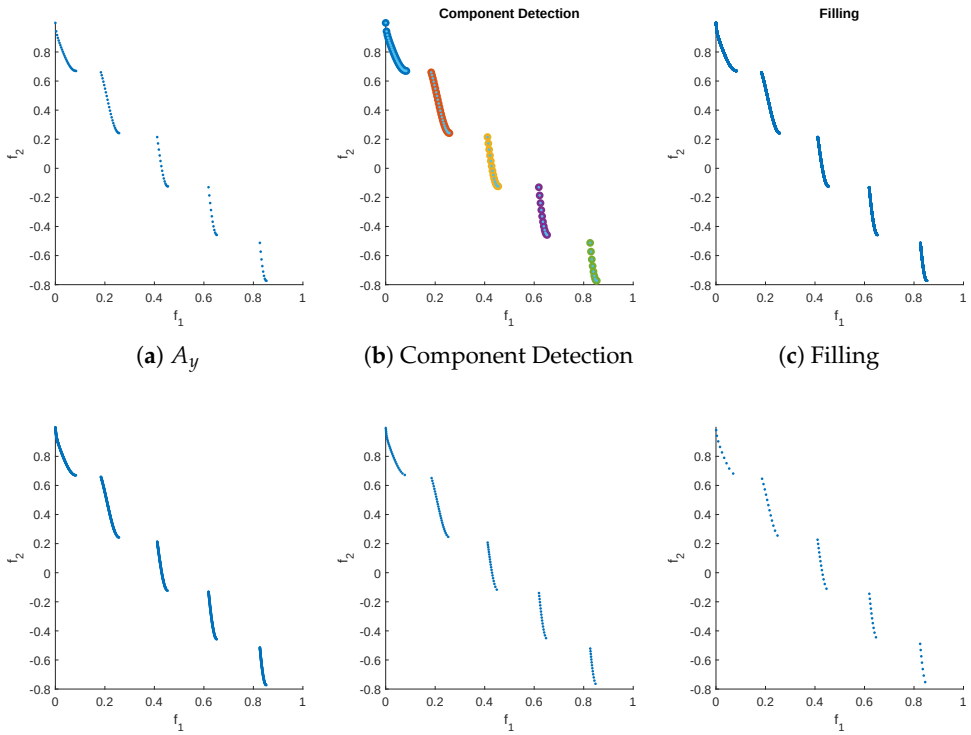
In this section, we show the strength of the novel approach to selected test problems. We further show – as far as possible – comparisons to related methods.
First, we show all the steps of the RSG on five representative problems: Figure 8 shows these steps for ZDT3 ($k = 2$ and disconnected), Figure 9 for DTLZ7 ($k = 3$ and disconnected), Figure 11 for CONV3 (defined in Appendix A, $k = 3$), Figure 12 for CONV3-4 (defined in Appendix A, $k = 3$) and Figure 13 for CONV4-2F (defined in Appendix A, $k = 4$ and disconnected).

In Figures 8 and 9, we have used starting sets A_y that have a slight bias. The last subfigures show the results of RSG for different values of N. For Figures 11-13, we demonstrate the universality of RSG, as it does not need any analytical information of the PF since it utilizes the result of a MOEA and an archiver to generate a reference of any desired size.

Additionally, a comparison between RSG, PlatEMO [57], Pymoo, and the filling step is presented in Figures 14–15 for the WFG2, ZDT1, ZDT3, DTLZ2, Convex DTLZ2 (CDTLZ2), C2-DTLZ2, and DTLZ7 test problems. Although the reference sets provided by PlatEMO and Pymoo are of high quality, they still exhibit some bias in certain bi-objective problems (such as ZDT1 and ZDT3), and especially in three-objective problems such as WFG2, and CDTLZ2. Furthermore, for problems like WFG2, and DTLZ7, the number of points in the reference sets of PlatEMO and Pymoo is limited to a fixed set, in contrast to RSG, which can generate any desired number of points.

 doi:10.20944/preprints202504.0294.v1

Upon acceptance of this paper, we will provide reference sets on our website http://neo.cinvestav.mx:3005/ for the problems considered in this work as well as other problems (in the future, we intend to provide data for all commonly used benchmark test functions). The website will allow users to request reference sets of any desired size for any of the aforementioned problems. Further, the codes of RSG will be found on our GitHub page https://github.com/aerfangel/RSG.

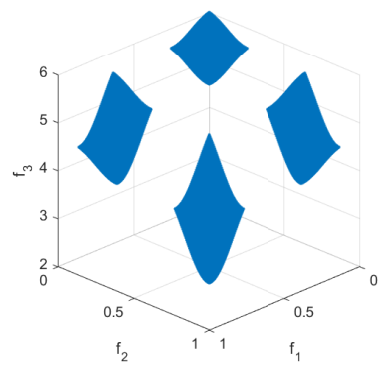


(**a**) A_y (**b**) Component Detection (**c**) Filling

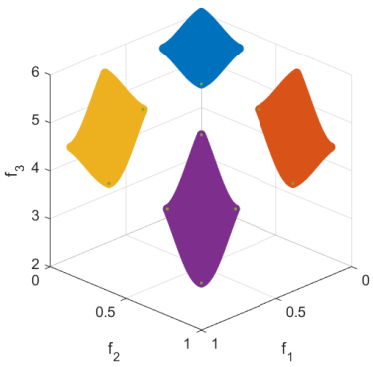
(**d**) Selection $N = 1,000$ (**e**) Selection $N = 100$ (**f**) Selection $N = 50$

Figure 8. RSG process for ZDT3. The starting set A_y is taken from PlatEMO. We set $N_f = 10,000$.

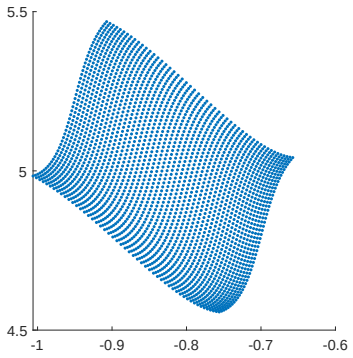
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2025 doi:10.20944/preprints202504.0294.v1



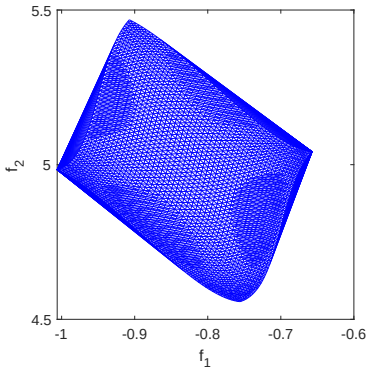
(**a**) A_y



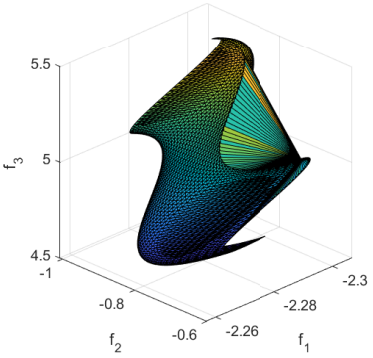
(**b**) Component Detection



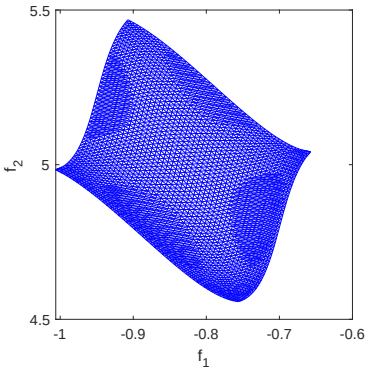
(**c**) 2D Projection (one component)



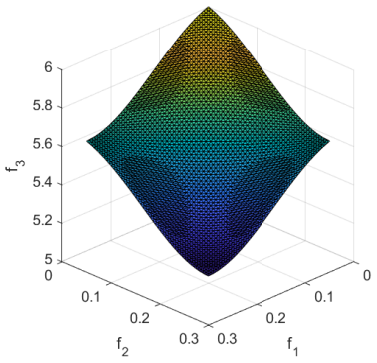
(**d**) 2D Triangulation



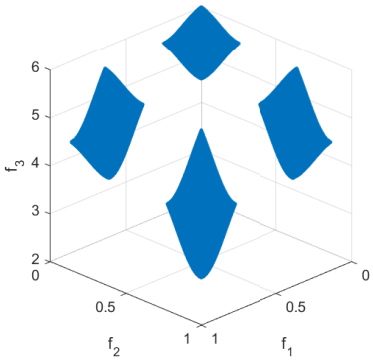
(**e**) 3D Triangulation



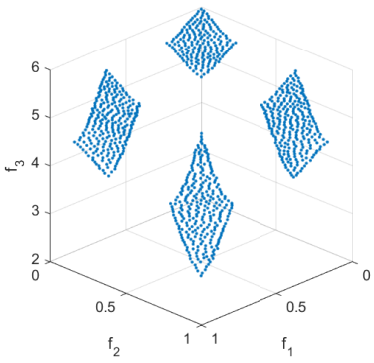
(**f**) 2D Triangulation cleaned



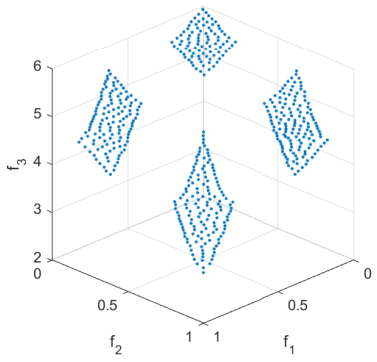
(**g**) 3D Triangulation cleaned



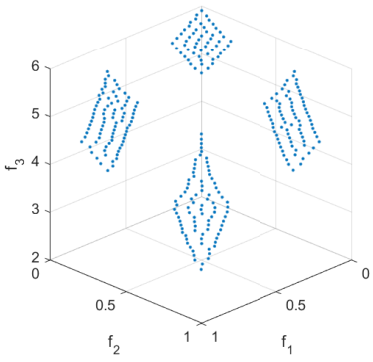
(**h**) Filling



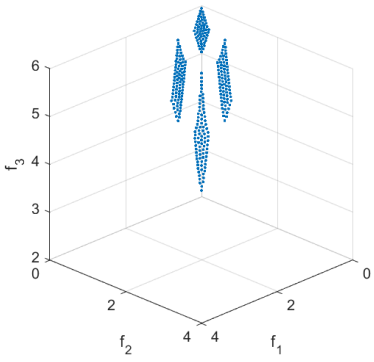
(**i**) Selection $N = 1,000$



(**j**) Selection $N = 500$



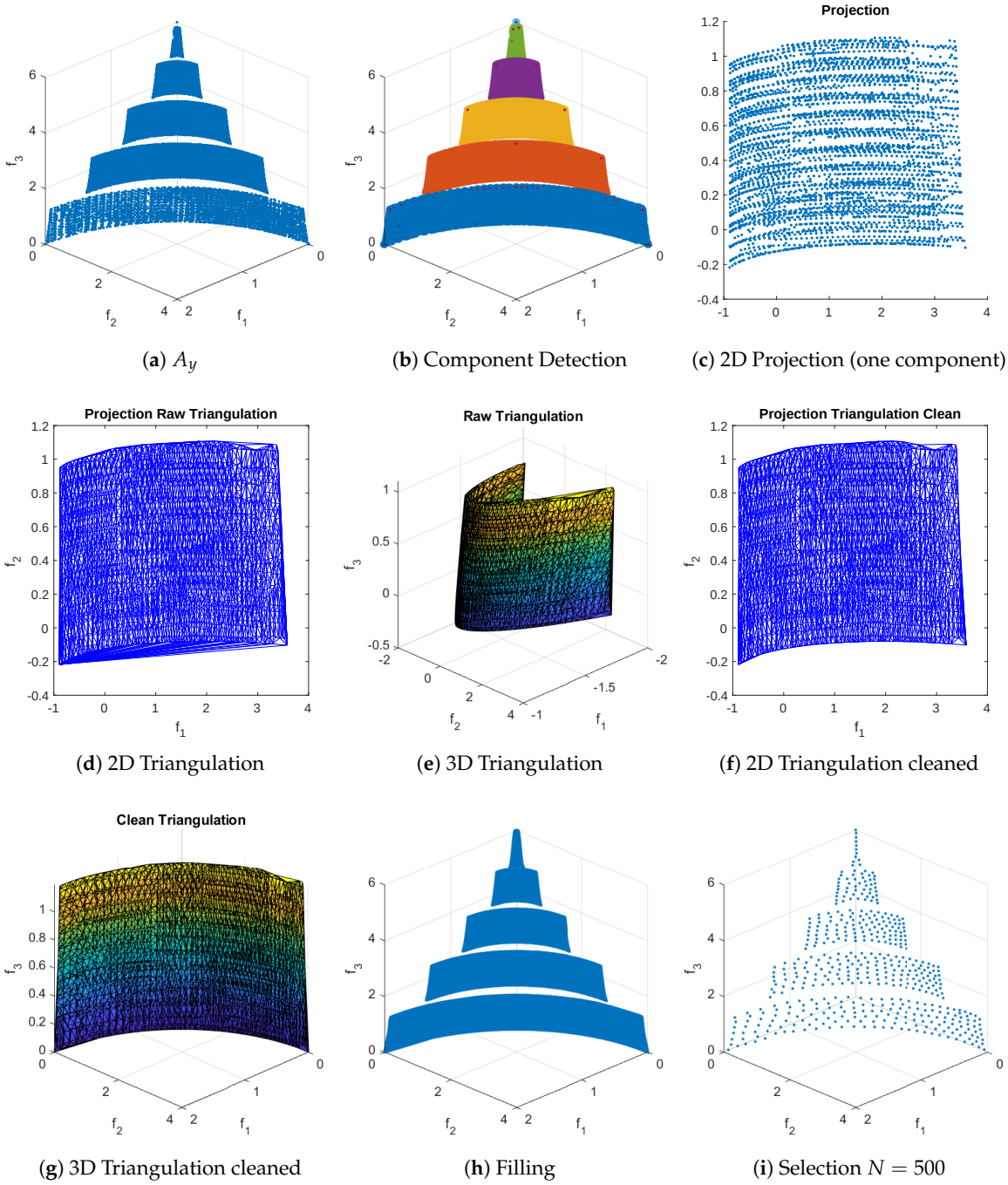
(**k**) Selection $N = 300$



(**l**) Selection $N = 300$

Figure 9. RSG process for DTLZ7. The starting set A_y is taken from Pymoo. We set $N_f = 100,000$. Figure (l) shows the result of RSG with $N = 300$ over the same range for all variables, displaying uniformity in the solution.

 doi:10.20944/preprints202504.0294.v1



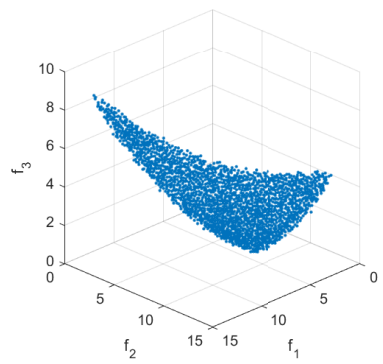
(**a**) A_y (**b**) Component Detection (**c**) 2D Projection (one component)

(**d**) 2D Triangulation (**e**) 3D Triangulation (**f**) 2D Triangulation cleaned

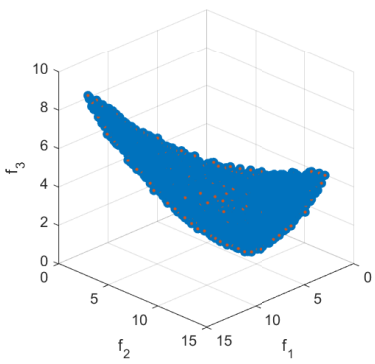
(**g**) 3D Triangulation cleaned (**h**) Filling (**i**) Selection $N = 500$

Figure 10. RSG process for WFG2. The starting set A_y is obtained using the PT and a non-dominance test. We set $N_f = 3,500,000$.

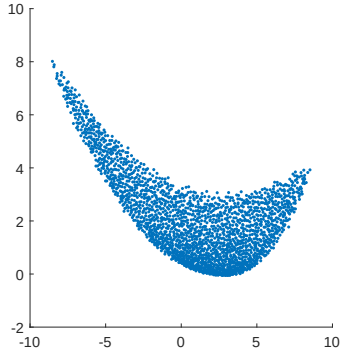
 doi:10.20944/preprints202504.0294.v1



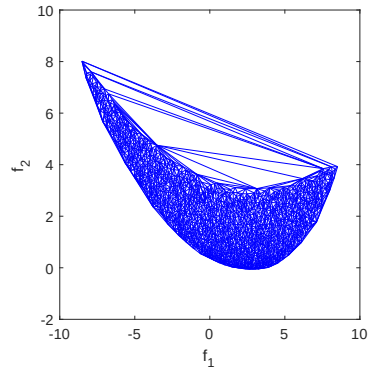
(**a**) A_y



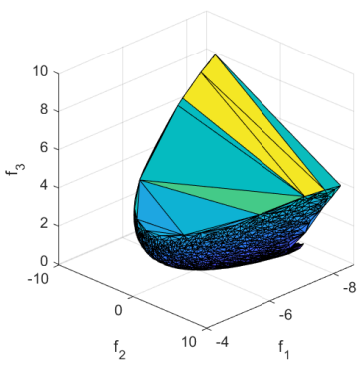
(**b**) Component Detection



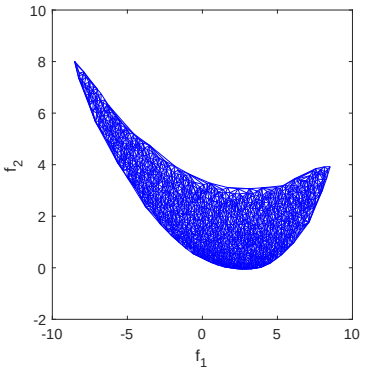
(**c**) 2D Projection (one component)



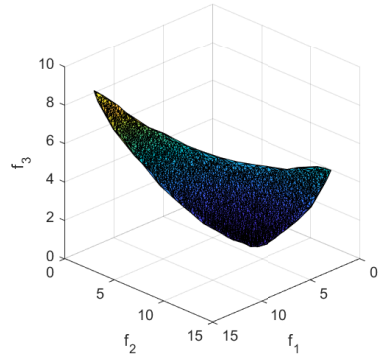
(**d**) 2D Triangulation



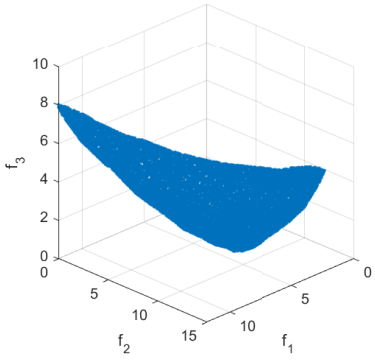
(**e**) 3D Triangulation



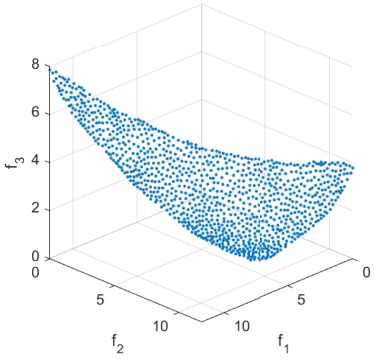
(**f**) 2D Triangulation cleaned



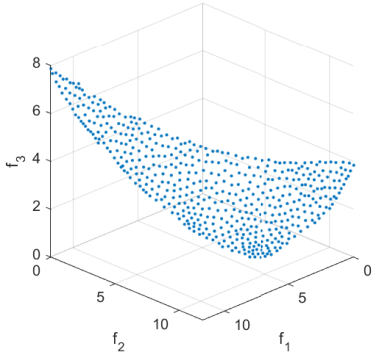
(**g**) 3D Triangulation cleaned



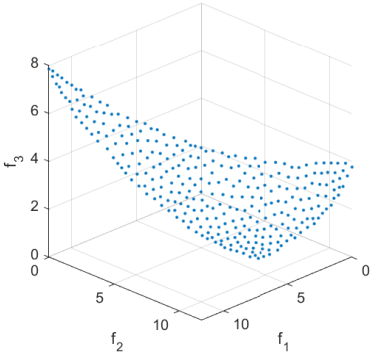
(**h**) Filling



(**i**) Selection $N = 1,000$



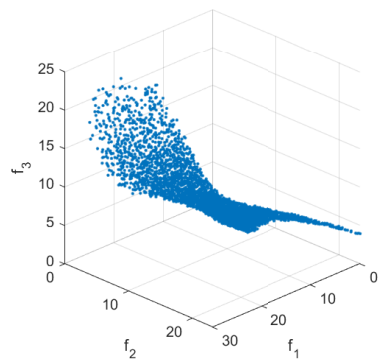
(**j**) Selection $N = 500$



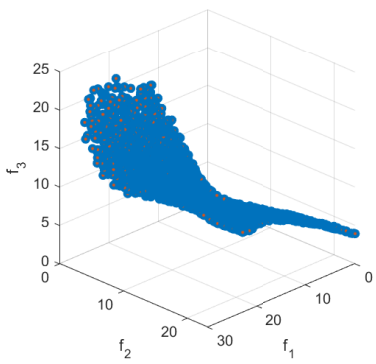
(**k**) Selection $N = 300$

Figure 11. RSG process for CONV3. The starting set A_y is obtained by sampling 100,000 points uniformly at random near the PS and then applying a nondominance test. We set $N_f = 10,000$.

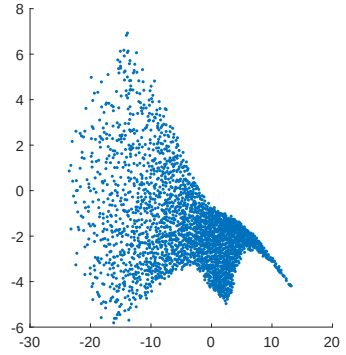
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2025 doi:10.20944/preprints202504.0294.v1



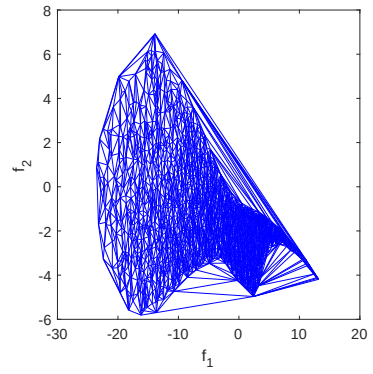
(**a**) A_y



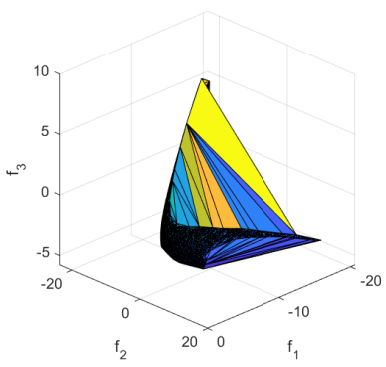
(**b**) Component Detection



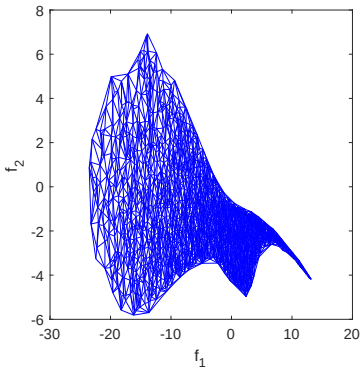
(**c**) 2D Projection (one component)



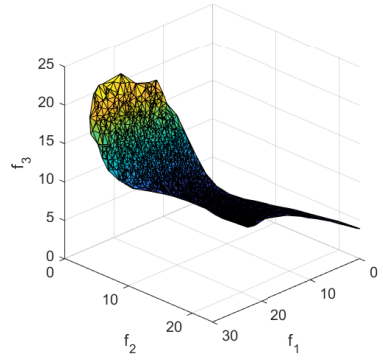
(**d**) 2D Triangulation



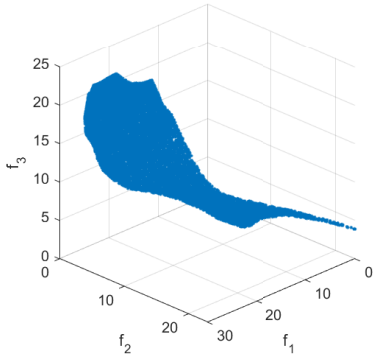
(**e**) 3D Triangulation



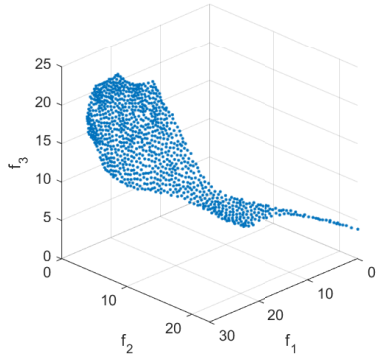
(**f**) 2D Triangulation cleaned



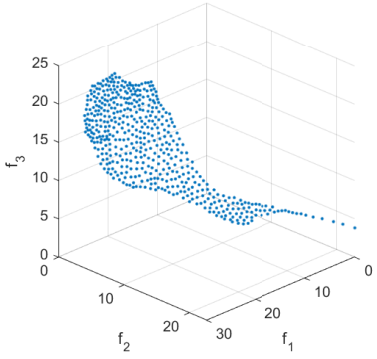
(**g**) 3D Triangulation cleaned



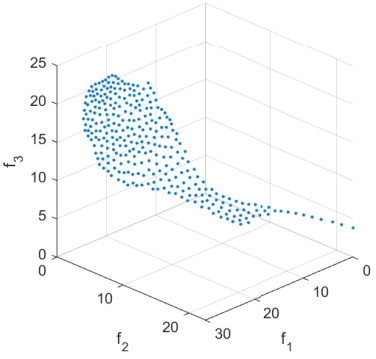
(**h**) Filling



(**i**) Selection $N = 1,000$



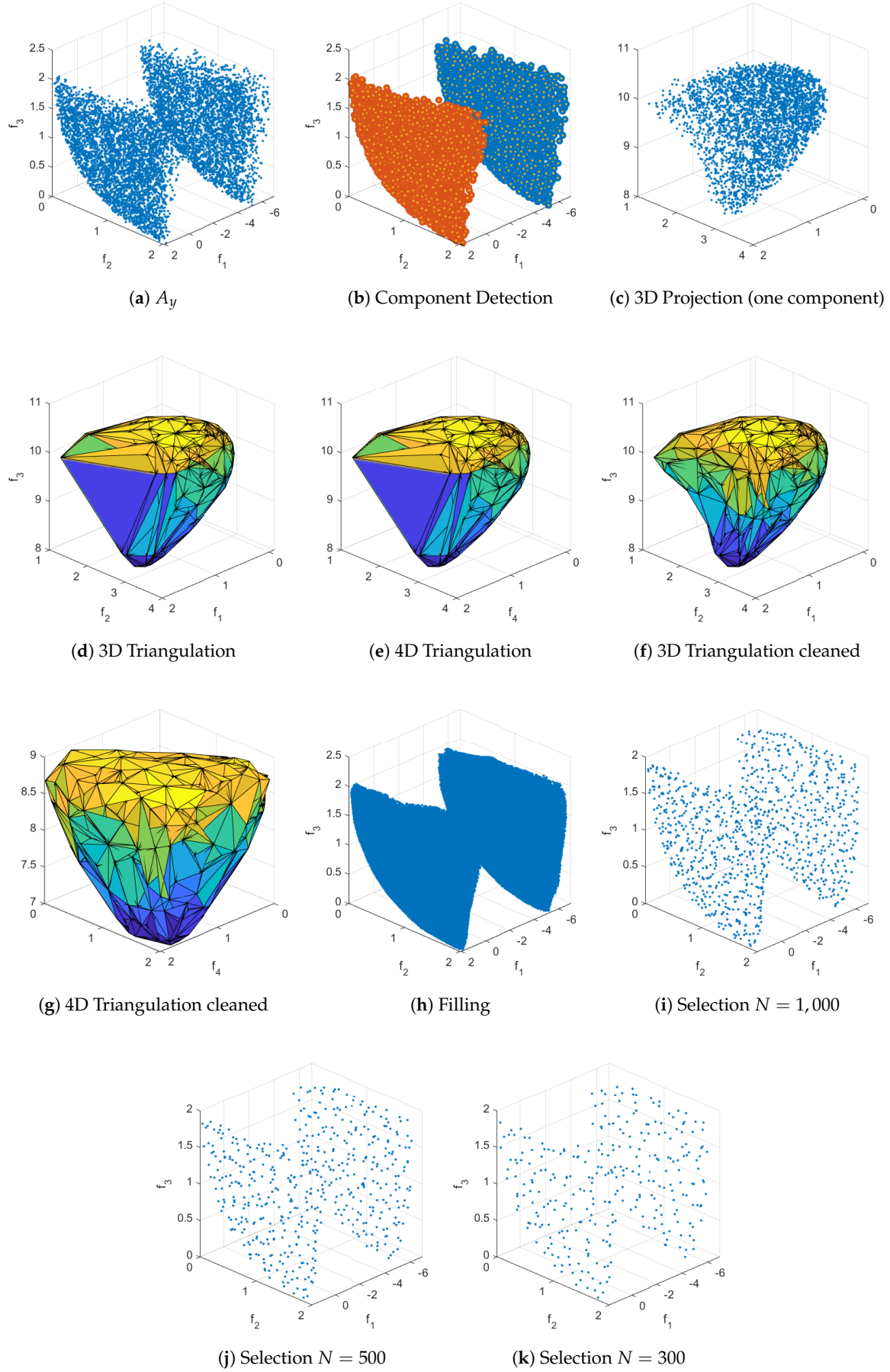
(**j**) Selection $N = 500$



(**k**) Selection $N = 300$

Figure 12. RSG process for CONV3-4. The starting set A_y is obtained by sampling 100,000 points uniformly at random near the PS and then applying a non-dominance test. We set $N_f = 10,000$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2025 doi:10.20944/preprints202504.0294.v1



(**a**) A_y (**b**) Component Detection (**c**) 3D Projection (one component)

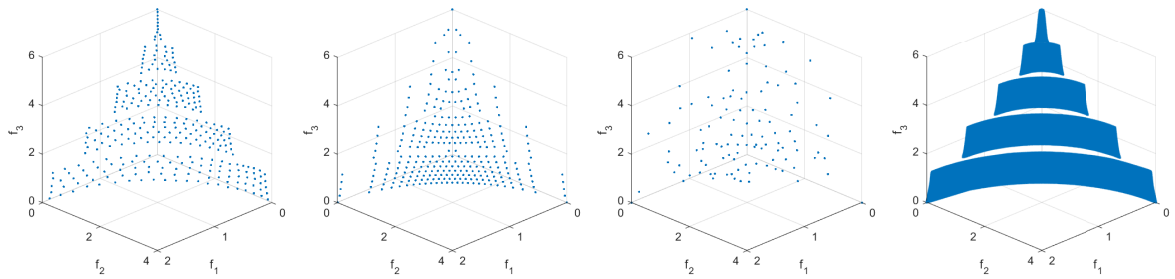
(**d**) 3D Triangulation (**e**) 4D Triangulation (**f**) 3D Triangulation cleaned

(**g**) 4D Triangulation cleaned (**h**) Filling (**i**) Selection $N = 1,000$

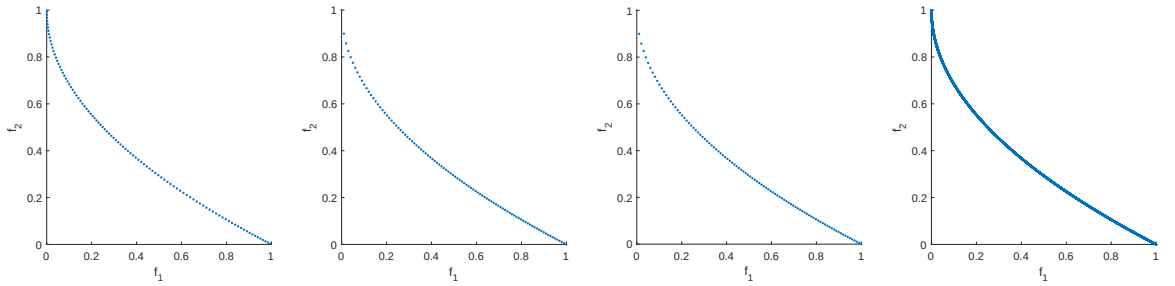
(**j**) Selection $N = 500$ (**k**) Selection $N = 300$

Figure 13. RSG process for CONV4-2F. The starting set A_y is obtained by merging the final populations from 30 independent runs of NSGA-III, each run consisting of 400 generations with a population size of 500, and then applying a non-dominance test to the merged set. We set $N_f = 100,000$.

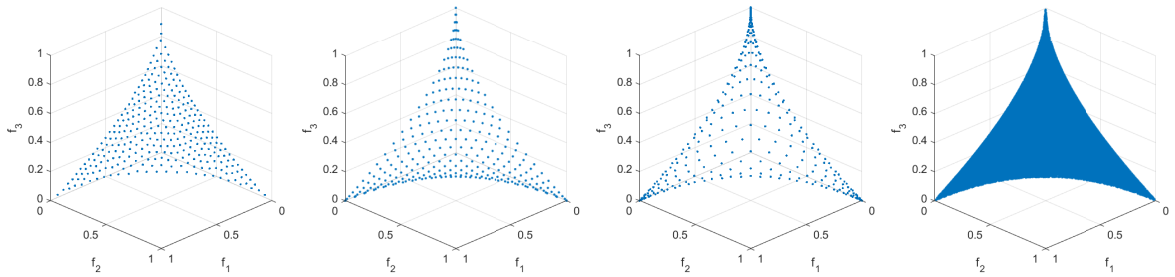
 doi:10.20944/preprints202504.0294.v1



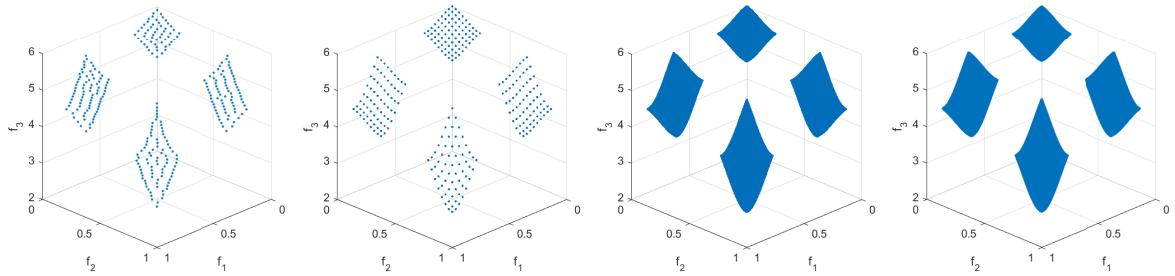
(**a**) RSG WFG2 ($N = 300$) (**b**) PlatEMO WFG2 ($N = 293$) (**c**) Pymoo WFG2 ($N = 136$) (**d**) F WFG2 ($N_f = 3,600,000$)



(**e**) RSG ZDT1 ($N = 100$) (**f**) PlatEMO ZDT1 ($N = 100$) (**g**) Pymoo ZDT1 ($N = 100$) (**h**) F ZDT1 ($N_f = 10,000$)



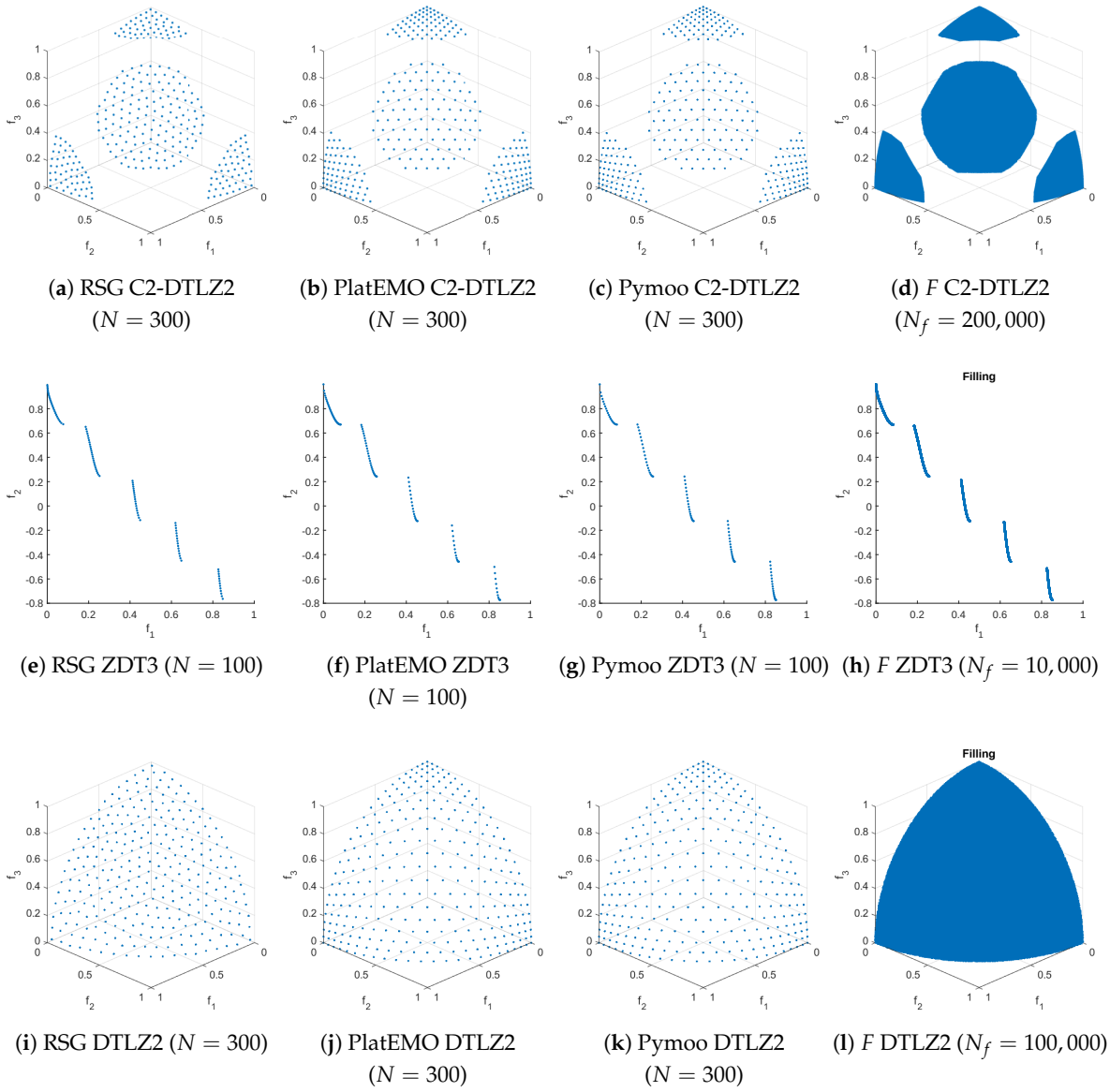
(**i**) RSG CDTLZ2 ($N = 300$) (**j**) PlatEMO CDTLZ2 ($N = 300$) (**k**) Pymoo CDTLZ2 ($N = 300$) (**l**) F CDTLZ2 ($N_f = 200,000$)



(**m**) RSG DTLZ7 ($N = 300$) (**n**) PlatEMO DTLZ7 ($N = 289$) (**o**) Pymoo DTLZ7 ($N = 9409$) (**p**) F DTLZ7 ($N_f = 2,800,000$)

Figure 14. The first part of the comparisons between RSG (first column), PlatEMO (second column), Pymoo (third column), and the filling step F of RSG (fourth column) are shown. A reference set of size 100 was used for bi-objective problems and 300 for three-objective problems. For bi-objective problems, all methods produced exactly 100 points. However, for three-objective problems, PlatEMO and Pymoo do not always yield exactly 300 points—for example, PlatEMO in WFG1, WFG2, and DTLZ7, and Pymoo in WFG1, WFG2, DTLZ7, CDTLZ2, and C2-DTLZ2, the last two being unobtainable (as far as the authors could determine) in Pymoo.

 doi:10.20944/preprints202504.0294.v1



(**a**) RSG C2-DTLZ2 ($N = 300$) (**b**) PlatEMO C2-DTLZ2 ($N = 300$) (**c**) Pymoo C2-DTLZ2 ($N = 300$) (**d**) F C2-DTLZ2 ($N_f = 200,000$)

(**e**) RSG ZDT3 ($N = 100$) (**f**) PlatEMO ZDT3 ($N = 100$) (**g**) Pymoo ZDT3 ($N = 100$) (**h**) F ZDT3 ($N_f = 10,000$)

(**i**) RSG DTLZ2 ($N = 300$) (**j**) PlatEMO DTLZ2 ($N = 300$) (**k**) Pymoo DTLZ2 ($N = 300$) (**l**) F DTLZ2 ($N_f = 100,000$)

Figure 15. The second part of the comparisons between RSG (first column), PlatEMO (second column), Pymoo (third column), and the filling step F of RSG (fourth column) are shown. A reference set of size 100 was used for bi-objective problems and 300 for three-objective problems. For bi-objective problems, all methods produced exactly 100 points. However, for three-objective problems, PlatEMO and Pymoo do not always yield exactly 300 points—for example, PlatEMO in WFG1, WFG2, and DTLZ7, and Pymoo in WFG1, WFG2, DTLZ7, CDTLZ2, and C2-DTLZ2, the last two being unobtainable (as far as the authors could determine) in Pymoo.

5. Conclusions and Future Work

In this paper, we have addressed the problem of obtaining bias-free and complete finite-size approximations of the solution sets (Pareto fronts) of multi-objective optimization problems (MOPs). Such approximations are, in particular, required for the fair usage of distance-based performance indicators, which are frequently used in evolutionary multi-objective optimization (EMO). If the Pareto front approximations are biased or incomplete, the use of these performance indicators can lead to misleading or false information. To address this issue, we have proposed the Reference Set Generator (RSG). This method starts with an initial (probably biased) approximation of the Pareto front. An unbiased approximation is then computed via component detection, filling, and a reduction to the desired size. The RSG can be applied to Pareto fronts of any shape and dimension. We have finally demonstrated the strength of the novel approach on several benchmark problems.

 doi:10.20944/preprints202504.0294.v1

In the future, we intend to use the RSG on the Pareto fronts of all commonly used continuous test problems. Special attention has to be paid to problems with degenerated fronts (i.e., problems where the Pareto front does locally not form an object of dimension $k-1$). In the current approach, we handle degeneracy using the PT to obtain a filled set and, from there, use the reduction step. For future work, we will explore if the projection can be modified to fill such sets, which may lead to a more general approach to handling degeneration.

Author Contributions: All authors contributed equally to this work.

Funding: O. Schütze acknowledges support from CONAHCYT project CBF2023-2024-1463.

Data Availability Statement: **Data Availability Statement:** The code and data presented in the study will be openly available upon acceptance in GitHub at https://github.com/aerfangel/RSG and http://neo.cinvestav.mx:3005/ respectively.

Acknowledgments: Angel E. Rodriguez-Fernandez acknowledges support from the SECIHTI to pursue his postdoc fellowship at the CINVESTAV-IPN.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A_y	Current PF approximation
ℓ	Size of A_y, the starting PF approximation
N	Desired size of approximation
Z	RSG result: Reference set of size N
F	Filled set
N_f	Size of Filling
C^i	i-th detected component
C	Set of all detected components
L	Total length of 2D curve
T'	Delaunay triangulation
δ'	Number of triangles in T'
T	Cleaned Triangulation
δ	Number of triangles in T
η	Normal vector
P_{k-1}	Projected A_y
ρ	Selected cleaning property (area/volume, largest side, or condition number)
ρ_i	Value of property ρ for triangle i
τ	Threshold for removing triangles
a_i	Area/volume of triangle/simplex i
A	Total area/volume of the triangulation
r	Radius of DBSCAN

Appendix A Function Definitions

1. CONV3

$$
\begin{aligned}
F &: [-3,3]^3 \to \mathbb{R}^3 \\
F(x) &= (f_1(x), f_2(x), f_3(x))^T \text{ where :} \\
f_i(x) &= \|x - a^i\|^2 \\
a^1 &= (-1,-1,-1), \quad a^2 = (1,1,1), \quad a^3 = (-1,1,-1)
\end{aligned}
$$

 doi:10.20944/preprints202504.0294.v1

2. **CONV3-4**

$$
\begin{aligned}
F &: [-3,3]^3 \to \mathbb{R}^3 \\
F(x) &= (f_1(x), f_2(x), f_3(x))^T \text{ where :} \\
f_1(x) &= (x_1 - a_1^1)^4 + (x_2 - a_2^1)^2 + (x_3 - a_3^1)^2 \\
f_2(x) &= (x_1 - a_1^2)^2 + (x_2 - a_2^2)^4 + (x_3 - a_3^2)^2 \\
f_3(x) &= (x_1 - a_1^3)^2 + (x_2 - a_2^3)^2 + (x_3 - a_3^3)^4 \\
a^1 &= (-1,-1,-1), \quad a^2 = (1,1,1), \quad a^3 = (-1,1,-1)
\end{aligned}
$$

3. **CONV4-2F**

$$
\begin{aligned}
F &: [-3,3]^4 \to \mathbb{R}^4 \\
F(x) &= (f_1(x), f_2(x), f_3(x), f_4(x))^T \text{ where :} \\
f_i(x) &= \begin{cases} \|x + \mathbb{1} - a^i\|^2 - 3.5\sigma & \text{if } x < (0,0,0,0) \\ \|x - a^i\|^2 & \text{otherwise} \end{cases} \\
\phi_1 &= (0, \|a^1 - a^2\|^2, \|a^1 - a^3\|^2, \|a^1 - a^4\|^2) \\
\phi_4 &= (\|a^4 - a^1\|^2, \|a^4 - a^2\|^2, \|a^4 - a^3\|^2, 0) \\
\sigma &= \phi_4 - \phi_1 \\
a^1 &= (1,0,0,0), \quad a^2 = (0,1,0,0) \quad a^3 = (0,0,1,0), \\
a^4 &= (0,0,0,1), \quad \mathbb{1} = (1,1,1,1),
\end{aligned}
$$

References

1. Hillermeier, C. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*; Vol. 135, Springer Science & Business Media, 2001.
2. Coello Coello, C.A.; Goodman, E.; Miettinen, K.; Saxena, D.; Schütze, O.; Thiele, L. Interview: Kalyanmoy Deb Talks about Formation, Development and Challenges of the EMO Community, Important Positions in His Career, and Issues Faced Getting His Works Published. *Mathematical and Computational Applications* **2023**, *28*. https://doi.org/10.3390/mca28020034.
3. Veldhuizen, D.A.V. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, Air Force Institute of Technology, 1999.
4. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.M.; Grunert, V.D.F. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* **2003**, *7*, 117–132.
5. Coello, C.A.C.; Cruz, N.C. Solving Multiobjective Optimization Problems Using an Artificial Immune System. *Genetic Programming and Evolvable Machines* **2005**, *6*, 163–190.
6. Schütze, O.; Esquivel, X.; Lara, A.; Coello Coello, C.A. Using the averaged Hausdorff distance as a performance measure in evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation* **2012**, *16*, 504–522.
7. Ishibuchi, H.; Masuda, H.; Nojima, Y. A Study on Performance Evaluation Ability of a Modified Inverted Generational Distance Indicator, New York, NY, USA, 2015; p. 695–702. https://doi.org/10.1145/2739480.2754792.
8. Bogoya, J.M.; Vargas, A.; Cuate, O.; Schütze, O. A (p,q)-Averaged Hausdorff Distance for Arbitrary Measurable Sets. *Mathematical and Computational Applications* **2018**, *23*.
9. Deb, K.; Ehrgott, M. On Generalized Dominance Structures for Multi-Objective Optimization. *Mathematical and Computational Applications* **2023**, *28*.
10. Deb, K.; Pratap, A.; Sameer, S.A.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on* **2002**, *6*, 182–197.
11. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In Proceedings of the Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001); Giannakoglou, K.; et al., Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

 doi:10.20944/preprints202504.0294.v1

12. Fonseca, C.M.; Fleming, P.J. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation* **1995**, *3*, 1 – 16.
13. Knowles, J.D.; Corne, D.W. Approximating the nondominated front using the Pareto Archived Evolution Strategy. *Evolutionary Computation* **2000**, *8*, 149 – 172.
14. Zhang, Q.; Li, H. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* **2007**, *11*, 712–731.
15. Deb, K.; Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *Transactions on Evolutionary Computation* **2014**, *18*, 577–601.
16. Jain, H.; Deb, K. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation* **2014**, *18*, 602–622.
17. Zuiani, F.; Vasile, M. Multi Agent Collaborative Search based on Tchebycheff decomposition. *Computational Optimization and Applications* **2013**, *56*, 189–208.
18. Moubayed, N.A.; Petrovski, A.; McCall, J. (DMOPSO)-M-2: MOPSO Based on Decomposition and Dominance with Archiving Using Crowding Distance in Objective and Solution Spaces. *Evolutionary Computation* **2014**, *22*. https://doi.org/{10.1162/EVCO_a_00104}.
19. Beume, N.; Naujoks, B.; Emmerich, M.T.M. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **2007**, *181*, 1653–1669.
20. Zitzler, E.; Thiele, L.; Bader, J. SPAM: Set Preference Algorithm for multiobjective optimization. In Proceedings of the Parallel Problem Solving From Nature PPSN X, 2008, pp. 847–858.
21. Wagner, T.; Trautmann, H. Integration of Preferences in Hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation* **2010**, *14*, 688–701.
22. Schütze, O.; Domínguez-Medina, C.; Cruz-Cortés, N.; de la Fraga, L.G.; Sun, J.Q.; Toscano, G.; Landa, R. A scalar optimization approach for averaged Hausdorff approximations of the Pareto front. *Engineering Optimization* **2016**, *48*, 1593–1617.
23. Sosa-Hernández, V.A.; Schütze, O.; Wang, H.; Deutz, A.; Emmerich, M. The Set-Based Hypervolume Newton Method for Bi-Objective Optimization. *IEEE Transactions on Cybernetics* **2020**, *50*, 2186 – 2196.
24. Fonseca, C.M.; Fleming, P.J., Genetic algorithms for multiobjective optimization: formulation, discussion, and generalization. In *Proceedings of the 5-th International Conference on Genetic Algorithms*; 1993; pp. 416 – 423.
25. Srinivas, N.; Deb, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* **1994**, *2*, 221 – 248.
26. Horn, J.; Nafpliotis, N.; Goldberg, D.E. A niched Pareto genetic algorithm for multiobjective optimization. In Proceedings of the Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation. IEEE Press, 1994, pp. 82 – 87.
27. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* **1999**, *3*, 257 – 271.
28. Rudolph, G. Finite Markov Chain results in evolutionary computation: A Tour d'Horizon. *Fundamenta Informaticae* **1998**, *35*, 67 – 89.
29. Rudolph, G. On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC 1998). IEEE Press, 1998, pp. 511 – 516.
30. Rudolph, G.; Agapie, A. Convergence Properties of Some Multi-Objective Evolutionary Algorithms. In Proceedings of the Evolutionary Computation (CEC), 2011 IEEE Congress on. IEEE Press, 2000.
31. Rudolph, G. Evolutionary Search under Partially Ordered Fitness Sets. In Proceedings of the Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001). ICSC Academic Press, Sliedrecht, The Netherlands, 2001, pp. 818 – 822.
32. Hanne, T. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research* **1999**, *117*, 553 – 564.
33. Hanne, T. Global multiobjective optimization with evolutionary algorithms: Selection mechanisms and mutation control. In Proceedings of the Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland. Springer Berlin, 2001, pp. 197 – 212.
34. Hanne, T. A multiobjective evolutionary algorithm for approximating the efficient set. *European Journal of Operational Research* **2007**, *176*, 1723 – 1734.

 doi:10.20944/preprints202504.0294.v1

35. Hanne, T. A Primal-Dual Multiobjective Evolutionary Algorithm for Approximating the Efficient Set. In Proceedings of the Evolutionary Computation (CEC), 2007 IEEE Congress on. IEEE Press, 2007, pp. 3127 – 3134.
36. Brockhoff, D.; Tran, T.D.; Hansen, N. Benchmarking numerical multiobjective optimizers revisited. In Proceedings of the Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, pp. 639–646.
37. Wang, R.; Zhou, Z.; Ishibuchi, H.; Liao, T.; Zhang, T. Localized weighted sum method for many-objective optimization. *IEEE Transactions on Evolutionary Computation* **2016**, *22*, 3–18.
38. Pang, L.M.; Ishibuchi, H.; Shang, K. Algorithm Configurations of MOEA/D with an Unbounded External Archive. *arXiv preprint arXiv:2007.13352* **2020**.
39. Nan, Y.; Shu, T.; Ishibuchi, H. Effects of External Archives on the Performance of Multi-Objective Evolutionary Algorithms on Real-World Problems. In Proceedings of the 2023 IEEE Congress on Evolutionary Computation (CEC), 2023, pp. 1–8. https://doi.org/10.1109/CEC53210.2023.10253980.
40. Rodriguez-Fernandez, A.E.; Schäpermeier, L.; Hernández, C.; Kerschke, P.; Trautmann, H.; Schütze, O. Finding ϵ-Locally Optimal Solutions for Multi-Objective Multimodal Optimization. *IEEE Transactions on Evolutionary Computation* **2024**, pp. 1–1. https://doi.org/10.1109/TEVC.2024.3458855.
41. Schütze, O.; Rodriguez-Fernandez, A.E.; Segura, C.; Hernández, C. Finding the Set of Nearly Optimal Solutions of a Multi-Objective Optimization Problem. *IEEE Transactions on Evolutionary Computation* **2024**.
42. Nan, Y.; Ishibuchi, H.; Pang, L.M. Small Population Size is Enough in Many Cases with External Archives. In Proceedings of the Evolutionary Multi-Criterion Optimization; et al., H.S., Ed. Springer Nature Singapore, 2025, pp. 99–113.
43. Author, P. Placehodler Title. *Journal of Placeholder* **1984**, *42*, 4200.
44. Laumanns, M.; Thiele, L.; Deb, K.; Zitzler, E. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation* **2002**, *10*, 263 – 282.
45. Schütze, O.; Laumanns, M.; Coello Coello, C.A.; Dellnitz, M.; Talbi, E.G. Convergence of Stochastic Search Algorithms to Finite Size Pareto Set Approximations. *Journal of Global Optimization* **2008**, *41*, 559 – 577.
46. Schütze, O.; Laumanns, M.; Tantar, E.; Coello Coello, C.A.; Talbi, E.G. Computing gap free Pareto front approximations with stochastic search algorithms. *Evolutionary Computation* **2010**, *18*, 65–96.
47. Schütze, O.; Lara, A.; Coello, C.A.C.; Vasile, M. On the Detection of Nearly Optimal Solutions in the Context of Single-Objective Space Mission Design Problems. *Journal of Aerospace Engineering* **2011**, *225*, 1229 – 1242.
48. Schütze, O.; Vasile, M.; Coello, C.A.C. Computing the Set of Epsilon-Efficient Solutions in Multiobjective Space Mission Design. *Journal of Aerospace Computing, Information, and Communication* **2011**, *8*, 53 – 70.
49. Schütze, O.; Hernández, C.; Talbi, E.G.; Sun, J.Q.; Naranjani, Y.; Xiong, F.R. Archivers for the Representation of the Set of Approximate Solutions for MOPs. *Journal of Heuristics* **2019**, *5*, 71 – 105.
50. Schütze, O.; Hernández, C. *Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms*; Springer, 2021.
51. Knowles, J.D.; Corne, D.W. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation* **2003**, *7*, 100 – 116.
52. Knowles, J.D.; Corne, D.W. Bounded Pareto archiving: Theory and practice. In Proceedings of the Metaheuristics for Multiobjective Optimisation. Springer, 2004, pp. 39 – 64.
53. Knowles, J.D.; Corne, D.W.; Fleischer, M. Bounded archiving using the Lebesgue measure. In Proceedings of the Proceedings of the IEEE Congress on Evolutionary Computation. IEEE Press, 2003, pp. 2490 – 2497.
54. López-Ibáñez, M.; Knowles, J.D.; Laumanns, M. On Sequential Online Archiving of Objective Vectors. In Proceedings of the Evolutionary Multi-Criterion Optimization (EMO 2011). Springer, Berlin, Heidelberg, 2011, pp. 46 – 60.
55. Laumanns, M.; Zenklusen, R. Stochastic convergence of random search methods to fixed size Pareto front approximations. *European Journal of Operational Research* **2011**, *213*, 414 – 421.
56. Tian, Y.; Xiang, X.; Zhang, X.; Cheng, R.; Jin, Y. Sampling Reference Points on the Pareto Fronts of Benchmark Multi-Objective Optimization Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–6. https://doi.org/10.1109/CEC.2018.8477730.
57. Tian, Y.; Cheng, R.; Zhang, X.; Jin, Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine* **2017**, *12*, 73–87.
58. Wang, H.; Rodriguez-Fernandez, A.E.; Uribe, L.; Deutz, A.; na, O.C.P.; Schütze, O. A Newton Method for Hausdorff Approximations of the Pareto Front Within Multi-objective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* **2024**, pp. 1–1. https://doi.org/10.1109/TEVC.2024.3469373.

 doi:10.20944/preprints202504.0294.v1

59. Bogoya, J.M.; Vargas, A.; Schütze, O. The Averaged Hausdorff Distances in Multi-Objective Optimization: A Review. *Mathematics* **2019**, *7*.
60. Rudolph, G.; Schütze, O.; Grimme, C.; Domínguez-Medina, C.; Trautmann, H. Optimal averaged Hausdorff archives for bi-objective problems: Theoretical and numerical results. *Comput Optim Appl* **2016**, *64*, 589 – 618.
61. Dilettoso, E.; Rizzo, S.A.; Salerno, N. A Weakly Pareto Compliant Quality Indicator. *Mathematical and Computational Applications* **2017**, *22*.
62. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the KDD; et al., E.S., Ed. AAAI Press, 1996, pp. 226–231.
63. Ben-David, S.; Ackerman, M. Measures of Clustering Quality: A Working Set of Axioms for Clustering. In Proceedings of the Advances in Neural Information Processing Systems; et al., D.K., Ed. Curran Associates, Inc., 2008, Vol. 21.
64. Delaunay, B. Sur la sphère vide. *Bulletin de l'Acadeémie des Sciences de l'URSS. Classe des sciences mathématiques et na* **1934**, *1934*, 793–800.
65. Smith, N.A.; Tromble, R.W. Sampling uniformly from the unit simplex **2004**.
66. Uribe, L.; Bogoya, J.M.; Vargas, A.; Lara, A.; Rudolph, G.; Schütze, O. A Set Based Newton Method for the Averaged Hausdorff Distance for Multi-Objective Reference Set Problems. *Mathematics* **2020**, *8*. https://doi.org/10.3390/math8101822.
67. Chen, W.; Ishibuchi, H.; Shang, K. Clustering-Based Subset Selection in Evolutionary Multiobjective Optimization. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2021, pp. 468–475. https://doi.org/10.1109/SMC52423.2021.9658582.
68. Martín, A.; Schütze, O. Pareto Tracer: A predictor-corrector method for multi-objective optimization problems. *Engineering Optimization* **2018**, *50*, 516–536.
69. Beltrán, F.; Cuate, O.; Schütze, O. The Pareto Tracer for General Inequality Constrained Multi-Objective Optimization Problems. *Mathematical and Computational Applications* **2020**, *25*. https://doi.org/10.3390/mca25040080.
70. Schütze, O.; Cuate, O. The Pareto Tracer for the treatment of degenerated multi-objective optimization problems. *Engineering Optimization* **2024**, pp. 1–26.