A Parameterless Performance Metric for Reference-Point Based Multi-Objective Evolutionary Algorithms

Sunith Bandaru
School of Engineering Science
University of Skövde, Sweden
sunith.bandaru@his.se

Henrik Smedberg
School of Engineering Science
University of Skövde, Sweden
henrik.smedberg@his.se

ABSTRACT

Most preference-based multi-objective evolutionary algorithms use reference points to articulate the decision maker's preferences. Since these algorithms typically converge to a sub-region of the Pareto-optimal front, the use of conventional performance measures (such as hypervolume and inverted generational distance) may lead to misleading results. Therefore, experimental studies in preference-based optimization often resort to using graphical methods to compare various algorithms. Though a few ad-hoc measures have been proposed in the literature, they either fail to generalize or involve parameters that are non-intuitive for a decision maker. In this paper, we propose a performance metric that is simple to implement, inexpensive to compute, and most importantly, does not involve any parameters. The so called *expanding hypercube metric* has been designed to extend the concepts of convergence and diversity to preference optimization. We demonstrate its effectiveness through constructed preference solution sets in two and three objectives. The proposed metric is then used to compare two popular reference-point based evolutionary algorithms on benchmark optimization problems up to 20 objectives.

CCS CONCEPTS

• **General and reference** → **Metrics**; **Performance**; • **Applied computing** → **Multi-criterion optimization and decision-making**; • **Theory of computation** → *Evolutionary algorithms*;

KEYWORDS

multi-objective optimization, decision making, reference point, performance metric, comparison

ACM Reference Format:
Sunith Bandaru and Henrik Smedberg. 2019. A Parameterless Performance Metric for Reference-Point Based Multi-Objective Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3321707.3321757

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321757

1 INTRODUCTION

Real-world applications of multi-objective optimization often implicitly include decision making, i.e. the process of obtaining a single solution to be implemented from the set of all possible Pareto-optimal solutions. This process requires a decision maker (DM) (or a group of unanimous decision makers) to provide preference information that can be used to select/generate desirable Pareto-optimal solution(s). The field of multi-criteria decision making (MCDM) identifies three classes of preference-based optimization methods depending on when the DM participates in the solution process. These are: (i) *a priori methods*, in which the DM's preferences are known in advance and the search involves finding a Pareto-optimal solution that best satisfies those preferences, (ii) *a posteriori methods*, in which representative Pareto-optimal solutions are generated first, which are then analyzed by a DM to select a solution that best fits his/her preferences, and (iii) *interactive methods*, which use the DM's preferences iteratively to guide the search towards a desirable region of the Pareto-optimal front.

Each class of MCDM methods has its advantages and drawbacks. A priori methods can be fast, but the DM may not be able to provide preferences for an unfamiliar problem. A posteriori methods can give the DM a better sense of the available trade-off, but generating enough solutions to adequately represent the Pareto-optimal front can be computationally expensive, especially when many objectives are involved. Moreover, when the number of solutions to be analyzed is large, it may be difficult for the DM to evaluate each of them against his/her preferences. Interactive methods require inputs from the DM at multiple stages, and therefore offer a certain degree of flexibility in decision making. On the downside, interactive methods place a high cognitive load on the DM and are prone to inconsistent or contradicting preference information.

1.1 Preference Modeling Methods

In addition to the question of *when* to incorporate preferences, it is also important to consider *how* the preferences are articulated. Various preference models are used in the literature. These can generally be categorized as one of the following [1, 17]:

(1) *Goals*: The most basic preference information comes in the form of a goal vector, which consists of desired target values for each objective. The intention here is to obtain a solution that minimizes deviations from the goal vector. Goals can also be defined through bounds on the objective function values. The vector containing the bounds for all objectives is called the *reservation point*.
(2) *Reference points*: A reference point is a vector composed of the aspiration levels of the DM for each objective. The

intention here is to obtain a solution on the Pareto-optimal front that minimizes deviations from the reference point.

(3) *Weights*: Weights can be used to assign a relative importance level for each objective. The final solution is usually obtained by optimizing an aggregation function in which objectives are weighted by their corresponding weights. Some preference models use the term 'reference direction' in relation to the reference point, but the components of this direction vector can be interpreted as weights.

(4) *Objective-ranking*: Objective-ranking is used when the DM considers some objectives to be more important than others, but is unable to assign importance levels (weights). Some preference models are used in this category: (i) *lexicographic order*, where the DM ranks all objectives in the order of importance, (ii) *preference relations*, where the DM assigns pairwise relationships between the objectives, such as 'less important', 'more important', 'equally important' or 'indifferent'. (iii) *Fuzzy logic* has also been used to model preferences expressed in linguistic terms.
Preference models in this category can sometimes be further processed to derive weights for the objectives.

(5) *Desirability functions*: These functions are non-linear mappings of objective functions to the desirability domain [0, 1], where 0 corresponds to the reservation point component and 1 corresponds to the reference point component for each objective. The DM can then articulate his/her preference through the desirability values.

(6) *Solution-ranking*: Some preference models aim to rank the solutions rather than the objectives. Two common methods in this category are: (i) *pairwise comparison of solutions*, where the DM is presented with a few representative solutions and asked to provide pairwise preferences. (ii) *Utility functions*, where the global preference structure of the DM is represented by a utility function (sometimes also called value function). Weighted aggregation function discussed above is essentially a linear utility function.

(7) *Trade-offs*: Preferences can also be modeled in terms of acceptable trade-offs between two or more objectives. For example, a statement from the DM such as "one unit improvement in f_i is at most worth a_{ji} units of degradation in f_j" can be used to redefine dominance.

(8) *Outranking*: Outranking method is widely used in MCDM. It requires specification of preference and indifference thresholds from the DM for each objective, which are used to compare pairs of candidate solutions. Once all pairs are considered, a preference flow is created which identifies the final solution.

(9) *Implicit models*: These models aim to find solutions on the Pareto-optimal front that should be naturally preferred, such as the knee point, a robust solution or the ideal solution (if feasible).

2 REFERENCE-POINT BASED MOEAS

Among the models described above, reference points are the most popular method of preference articulation [3]. Originally developed within MCDM by Wierzbicki [19], the idea of a reference point was first adopted in a multi-objective evolutionary algorithm (MOEA) in [9]. Since then it has been used extensively in many a priori preference-based MOEAs. The reasons for the popularity of this trio combination are: (i) a reference point is intuitively understood and allows a Pareto-optimal solution to be found, (ii) a priori methods are very practical because they are computationally cheaper than a posteriori methods, and put lower cognitive load on the DM than interactive methods, (iii) MOEAs, with their advantage of using a population, can generate multiple *preference solutions* close to the reference point instead of a single solution like most traditional MCDM methods. This gives the DM the benefit of a posteriori choice in the vicinity of the reference point, also called the *region of interest* (ROI).

Some of the well-known reference-point based MOEAs are (i) preference articulation in MOGA [9], (ii) reference-point based NSGA-II [7], (iii) light beam search based approach [4], (iv) g-dominance approach [13], (v) preference-based evolutionary algorithm [16], and (vi) r-dominance approach [15].

2.1 Performance Assessment

Despite the large number of reference-point based MOEAs available in the literature, there is a dearth of measures that can be used to evaluate the performance of these algorithms. Most comparative studies use scatter plots or parallel-coordinate plots to visually depict the obtained preference solutions. In addition to the subjectivity that this involves, the multiplicity of solutions can make visual assessment difficult. Moreover, since preference solutions are obtained from multiple runs of the algorithms, several plots have to be analyzed. Also, the qualitative nature of visual assessment means that statistical analysis cannot be performed on the algorithms.

As demonstrated in [11], measures such as hypervolume (HV) [21] and inverted generational distance (IGD) [2], that have been developed for evaluating approximations of the whole Pareto-optimal front, cannot be directly used to reliably assess preference solution sets. In fact, any metric that does not take the preference information into account is bound to give misleading results. A survey of the literature found only a handful of studies that propose performance metrics specially designed for this purpose. These metrics are briefly described in the next section. It is worth noting that all of the following studies use reference points for preference articulation.

2.2 Related Work

The earliest work in this regard is probably that of [18], where the authors adapt the hypervolume metric for preference-based optimization. The preference solution sets from all algorithms to be compared are first combined and the solution that is closest to the ideal point is identified. Next, a ROI is defined around this solution using a parameter δ and all solutions outside of this region are discarded. The hypervolume is calculated for the remaining solutions in each set using their common nadir point as reference. Since this method does not consider the preference information at all, it can lead to false conclusions as shown in [11].

The metric proposed in [12] uses the non-dominated solutions from the combined solution sets to make a composite front, which the authors call *User-Preference Composite Front* (UPCF). The closest

solution to each reference point is then found and a ROI is defined around it using a parameter r. IGD and hypervolume are then calculated based on the solutions inside the preferred regions. The IGD value is calculated using sample points from the composite front. The main issue with this approach is that all solutions outside the preference region are considered to be equally worse. For an illustrative example, see [11].

The metric proposed in [14] works by generating a grid of points, called User based Front (UbF), with the reference point at the center. Two measurements are defined: (i) iGD is the average over the shortest distances from each solution in the preference set to any of the grid points on the UbF, and (ii) iIGD is the average over the shortest distances from each grid point on the UbF to any of the solutions in the preference set. The authors claim that while iGD represents convergence, iIGD represents both convergence and diversity. The grid is analogous to ROI. The size of the grid is a parameter which defines this ROI.

The R-metric, first proposed in [6] and generalized in [11], uses concepts of MCDM in the design of the metric. It works by identifying a ROI around a pivot solution selected from the preference set based on an achievement scalarization function (ASF). The ASF is then used to transfer all solutions in the ROI to a virtual position, and finally the R-metric for a solution set is calculated using the baseline metric of either HV or IGD for all the solutions within the ROI. Solutions outside of the ROI are discarded. The resulting R-metric is denoted, R-HV or R-IGD based on which baseline is used. R-metric requires the decision maker to supply (i) a parameter Δ which determines the size of the ROI, (ii) a weight vector $\mathbf{w}$ which determines the relative importance of the objectives and (iii) a worst point $\mathbf{z}^{\mathbf{w}}$ which is used to create the reference direction for the ASF.

The PMDA-metric (Preference Metric based on Distances and Angles) presented in [20] is calculated in four steps. First, a preferred region is defined using $M + 1$ *light beams* that originate from the ideal point. The central beam passes through the reference point, while the others are inclined towards each of the M objectives based on a user-defined parameter ϵ. The intersection of these beams with the normalized hyperplane gives a point set Q. Next, a preference-based hyperplane is defined based on solutions in the preferred region and a parameter Ω. Q is mapped to this hyperplane as Q'. Then the minimum distance and angle for each solution in the preference set are calculated from the points in Q' and from the central beam, respectively. Finally, the PMDA-metric is calculated by aggregating the distances and angles over all solutions. Note that the parameter ϵ essentially controls the ROI.

The metric proposed in [10] first constructs a hyperplane on the reference point with the orientation given by a user-defined weight vector. The preference solution set is mapped onto the hyperplane and a ROI is defined on the hyperplane with the reference point as the center. The metric is defined as the Euclidean distances between the mapped points and the reference point, where solutions outside of the ROI get penalized by a penalty coefficient. The calculation requires a parameter r to define the ROI, a parameter w as the weight vector that describes the relative importance of each objective and a parameter k as the penalty coefficient. Unlike most other performance metrics, this metric does not require the objective space to be normalized to $[0, 1]$.

2.3 The Problem with ROI

Note that all the above described performance metrics involve one or more parameters. A recurring parameter is the size of ROI, which is to be defined by the DM. Consider the case when two algorithms, A and B, have a similar distribution of solutions inside the ROI, but immediately outside the ROI, B has a better distribution of non-dominated solutions. In practice, B would be the better algorithm. But the metrics described above will not be able to detect this unless the size of the ROI is increased. The problem with discarding solutions outside the ROI is that the metric becomes sensitive to ROI. Thus, even with normalization of the objectives, setting the parameter for ROI is not intuitive. A related issue is that when the ROI contains no solutions, then nothing can be said about the performance of the algorithms.

Another problem with ROI can be observed in high-dimensional objective spaces, where the solutions are sparsely distributed. In order to calculate the above metrics, a larger ROI must be used so that at least a few solutions are included in it. However, it is difficult to estimate a suitable value.

3 EXPANDING HYPERCUBE METRIC

Our proposed solution to the problem of defining the ROI is to *not* define it at all! The ROI is instead replaced by an *expanding hypercube*, which originates at the reference point and expands (with the reference point at its center) until it envelops all solutions. Thus, none of the non-dominated solutions are discarded and each of them plays a part in the calculation of the metric. As the hypercube expands, we record two characteristic values: (i) size of the hypercube, and (ii) fraction of (unique) solutions from the preference set that are inside the hypercube. The size of the hypercube can be thought of as a measure of convergence. The smaller this size, the closer the preference solutions are to the reference point. Similarly, the fraction of (unique) solutions inside the hypercube at any given point can be thought of as a measure of diversity. The larger this fraction, the higher the diversity of preference solutions. Thus, a trade-off curve can be generated by plotting the two recorded characteristic values.

Definition: The Expanding Hypercube metric (EH-metric) is defined as the area under the trade-off curve generated by expanding a hypercube from the reference point and plotting the fraction of enveloped points versus the size of the hypercube.

Figure 1 illustrates the generation of trade-off curve and the calculation of EH-metric. The preference set has 15 solutions. The hypercube (square in this case) starts to expand from a size of zero units with the reference at its center. When the hypercube is one unit in size, no points are enveloped. So we record the coordinates of the trade-off curve as $(1, 0)$. At a size of two units, the hypercube envelops one point, and the corresponding point on the trade-off curve is $(2, 1/15)$. Continuing in this fashion, we generate the following trade-off points (3,4/15), (4,9/15), (5,11/15) and finally (6,1). The area under this trade-off curve is easily calculated to be 25/15 or 1.67 which is the EH-metric for the preference set shown in Figure 1.

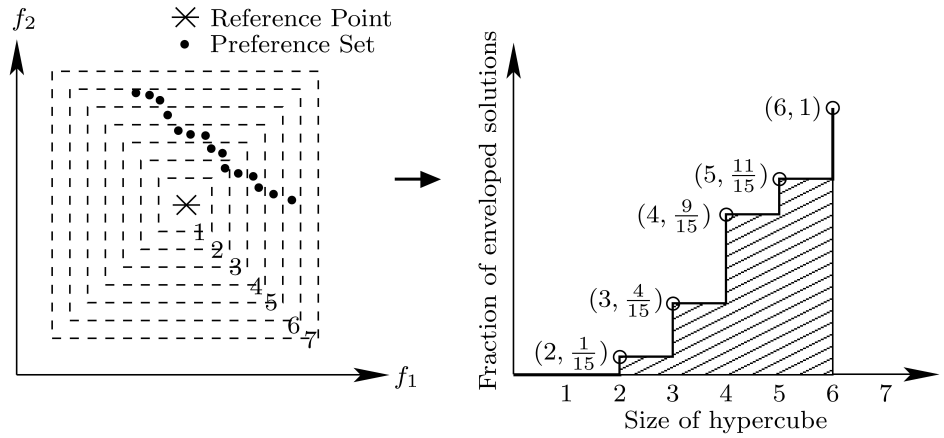


Figure 1: Generation of trade-off curve from an expanding hypercube. The shaded area represents the EH-metric.

The shaded area can be thought of as the hypervolume of the bi-objective problem involving minimizing the size of the hypercube (i.e. improving convergence) and maximizing the fraction of enveloped solutions (i.e. improving diversity). Thus, the EH-metric represents the simultaneous optimization of convergence and diversity.

In order to use the EH-metric to compare two or more preference sets from different algorithms, the first step is to remove duplicate solutions from each set, combine all preference sets and then remove all dominated solutions. We refer to these three tasks together as *prescreening*. After the prescreening step, the trade-off curves for each preference set are generated using the expanding hypercube as described above. Note that these curves must all be generated on the same plot, so that when calculating the EH-metric for each preference set, the same final size of the hypercube is used. This is similar in principle to using the same reference point[1] when calculating hypervolumes for different Pareto-front approximations.

3.1 Illustrative Preference Sets

In this section, we visually demonstrate the effectiveness of the EH-metric using constructed preference sets. Figure 2 shows 10 preference sets in a two-objective space, each set containing 40 points. The reference point $[0.35, 0.65]$ is shown as ✳. As seen in Figure 3, the EH-metric is maximal for the preference set that is closest to the reference point, i.e. the third set. A similar observation can be made from Figures 4 and 5, which illustrate preference sets in three objectives. Among the 21 preference sets, set 15 is closest to the reference point $[0.7, 0.9, 0.2]$ and therefore has the highest value for the EH-metric. In both examples, it is worth noting that as the preference sets get further away from the reference point, the EH-metric decreases.

3.2 Algorithmic Calculation of the EH-Metric

The calculation of the EH-metric does not involve any user-defined parameters. The only inputs required are: (i) the list of preference sets, *solutionSets*, obtained from different reference-point based

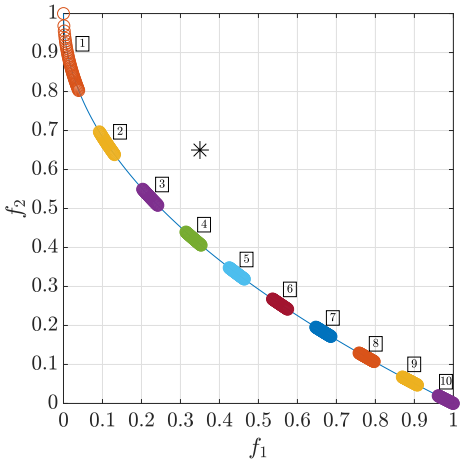


Figure 2: Constructed preference sets with two objectives.

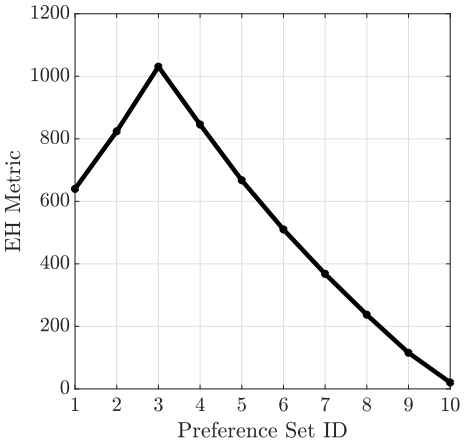


Figure 3: EH-Metric for the preference sets in Figure 2.

[1]Not to be confused with the reference point that is used for preference articulation.

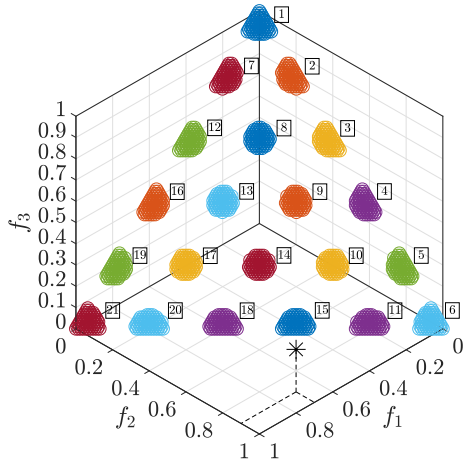


Figure 4: Constructed preference sets with three objectives.

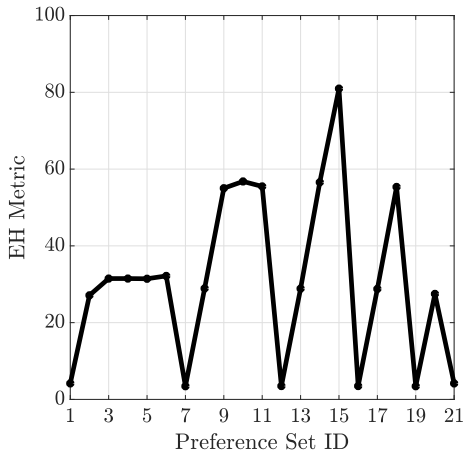


Figure 5: EH-Metric for the preference sets in Figure 4.

optimization algorithms that need to be compared. Each set contains all solutions from the final generation of one algorithm; (ii) the reference point, $referencePoint$, used by the algorithms.

The EH-metric is calculated in two steps as shown in Algorithm 1. The first step is *prescreening* the solutions sets. The prescreening

Algorithm 1 EH-Metric

Require: $solutionSets$, $referencePoint$
1: $solutionSets \leftarrow$ Prescreening($solutionSets$)
2: $setAreas \leftarrow$ GetAreas($solutionSets, referencePoint$)
3: **return** $setAreas$

routine can be found in Algorithm 2. It first removes duplicate solutions from all solution sets individually, through the RemoveDuplicates( ) function. This ensures that only unique solutions are used to calculate the fraction of enveloped solutions shown in Figure 1. Next, the prescreening routine combines the solution sets and removes all dominated solutions from each set. This step ensures that only the non-dominated solutions are used to calculate the EH-metric. Note that some solutions sets may become empty during this step. Their corresponding EH-metric values will be zero.

Algorithm 2 Prescreening

Require: $solutionSets$
1: **for all** $set \in solutionSets$ **do**
2: RemoveDuplicates(set)
3: **for all** $s_p \in set_p \in solutionSets$ **do**
4: **for all** $s_q \in set_q \in solutionSets$ **do**
5: **if** $s_p \prec s_q$ **then**
6: $set_q \leftarrow set_q \setminus \{s_q\}$
7: **if** $s_q \prec s_p$ **then**
8: $set_p \leftarrow set_p \setminus \{s_p\}$
9: **return** $solutionSets$

The second and final step in the calculation of EH-metric is to determine the area under the trade-off curve generated for each preference set as explained through Figure 1. This is done by the GetAreas( ) routine shown in Algorithm 3. For each *solution* in a given *set*, it uses the GetHypercubeSize( ) sub-routine to calculate the minimum size of the hypercube (centered at the reference point) that is required to envelope that solution. These sizes are stored in the set *hypercubeSizes*, which is sorted in ascending order to get *orderedHypercubeSizes*. The area under the trade-off curve is calculated incrementally by stepping through *orderedHypercubeSizes*. In Line 13, $i/|set|$ represents the fraction of enveloped solutions at the current step, while $(s - previousHypercubeSize)$ denotes the incremental increase in the size of the hypercube from the previous step. The multiplication of these two terms is akin to the calculation of the areas of successive shaded rectangles in Figure 1.

The GetHypercubeSize( ) sub-routine is shown in Algorithm 4, where M denotes the number of objectives. The sub-routine simply returns the largest coordinate of the absolute difference between the $referencePoint$ and the given *solution*. The largest coordinate corresponds to the minimum size of the hypercube centered at $referencePoint$ that is required to envelope *solution*.

For all sets in *solutionSets*, the area under the trade-off curve are stored in *setAreas* and the final hypercube sizes are stored in *setSizes*, as shown in Lines 16 and 17 of Algorithm 3. Note that the final hypercube sizes may be different for each solution set. Let *maxSize* denote the maximum hypercube size in *setSizes*. As stated before, in order to correctly compare the area under the trade-off curves for different solution sets, it is important to extend the trade-off curves of all solution sets to *maxSize*. This means that an additional area of Δ, as calculated in Line 20 of Algorithm 3, is added to each area in *setAreas*. The EH-metric values for all solutions sets are now available in *setAreas*.

3.3 Time Complexity Analysis

The duplicate solution removal in Prescreening( ) can be implemented to have a time complexity of $O(N^2L)$, where N is the typical size of each solution set, and L is the number of solution sets. The time complexity of the dominance comparisons is $O(MN^2L^2)$, where M is the number of objectives. Thus, the time complexity of prescreening step is $O(MN^2L^2)$.

Algorithm 3 GetAreas

Require: $solutionSets, referencePoint$
1: $setAreas \leftarrow \emptyset$
2: $setSizes \leftarrow \emptyset$
3: **for all** $set \in solutionSets$ **do**
4: $i \leftarrow 0$
5: $area \leftarrow 0$
6: $previousHypercubeSize \leftarrow 0$
7: $hypercubeSizes \leftarrow \emptyset$
8: **for all** $solution \in set$ **do**
9: $s \leftarrow$ GetHypercubeSize($solution, referencePoint$)
10: $hypercubeSizes \leftarrow hypercubeSizes \cup s$
11: $orderedHypercubeSizes \leftarrow$ Sort($hypercubeSizes$)
12: **for all** $s \in orderedHypercubeSizes$ **do**
13: $area \leftarrow area + (i/|set| * (s - previousHypercubeSize))$
14: $previousHypercubeSize \leftarrow s$
15: $i \leftarrow i + 1$
16: $setAreas \leftarrow setAreas \cup area$
17: $setSizes \leftarrow setSizes \cup$ Max($hypercubeSizes$)
18: $maxSize \leftarrow$ Max($setSizes$)
19: **for** $i \leftarrow 1$ to $|setAreas|$ **do**
20: $\Delta \leftarrow maxSize - setSizes_i$
21: $setAreas_i \leftarrow setAreas_i + \Delta$
22: **return** $setAreas$

Algorithm 4 GetHypercubeSize

Require: $solution, referencePoint$
1: $largest \leftarrow 0$
2: **for** $i \leftarrow 1$ to M **do**
3: $d \leftarrow |solution_i - referencePoint_i|$
4: $largest \leftarrow$ Max($largest, d$)
5: **return** $largest$

The time complexity for GetAreas( ) in Algorithm 3 is mainly influenced by (i) GetHypercubeSize( ) in Line 9, (ii) sorting in Line 11, (iii) calculation of incremental areas in Line 13, and (iv) calculation of additional areas Δ in Line 20. GetHypercubeSize( ) requires $O(MNL)$ operations to generate *hypercubeSizes*. Mergesort is used in Line 11. It has a time complexity of $O(N \log(N))$ in both the average and worst case. The time complexity for sorting L solution sets becomes $O(N \log(N)L)$. The calculation of incremental areas requires $O(NL)$ operations, while the calculation of Δ for each solution set requires $O(L)$ operations. The time complexity of GetAreas( ) is therefore $O(MNL)$.

The above analysis shows that the prescreening step dominates the computation of the EH-metric. The overall time complexity is therefore $O(MN^2L^2)$.

4 EXPERIMENTAL SETUP

In the rest of the paper, we use EH-metric and R-HV (i.e. R-metric for hypervolume) described in Section 2.2, to assess the performance of two popular reference-point based MOEAs, namely R-NSGA-II [7] and g-NSGA-II [13]. Both algorithms modify the well-known NSGA-II algorithm [5] for preference based multi-objective optimization.

R-NSGA-II modifies the crowding operator of NSGA-II in a way such that solutions closer to the reference point are preferred. The algorithm uses a minimum distance ϵ to produce a spread among the solutions close to the reference point. The higher the value of ϵ, the greater will be the spread of solutions around the reference point. In this study ϵ was set to 0.002 for all experimental runs.

g-NSGA-II modifies NSGA-II with a new dominance behavior called g-dominance which flags solutions as 1 if they dominate or are dominated by the reference point, and as 0 otherwise. The algorithm then lets all flag 1 solutions dominate all flag 0 solutions before regular non-dominated sorting is applied.

The three parameters for calculating the R-HV metric are set as follows based on the recommendations in [11]: (i) $\Delta = 0.5$ for defining the ROI, (ii) $\mathbf{w} = [1/\sqrt{M}, \ldots, 1/\sqrt{M}]$ for giving equal importance to all objectives, and (iii) $\mathbf{z^w} = referencePoint + 2.0 \times \mathbf{w}$ for creating the reference direction for ASF and for calculating the hypervolume.

A zero value for R-HV metric indicates that all solutions in the preference set lie outside the ROI defined by the R-metric. A zero value for EH-metric indicates that all solutions in the preference set are dominated by the solutions of other preference set and therefore get removed in the prescreening step.

4.1 Test Problems

Four instances from the scalable DTLZ test suite [8] are used to compare the the performance of the two algorithms. The instances are DTLZ1-4 with $M = 3, 5, 10, 15$ and 20 objectives. Both unattainable and attainable reference points are used. Though both algorithms are able to handle multiple reference points, we only use one reference point in all experimental runs.

In order to specify the reference point, we first create a point $\mathbf{w}$ on the Pareto-optimal fronts of each problem instance. For DTLZ1, $\mathbf{w}$ is defined by $w_i = 0.5 * i/(M^2 + M)/2, \forall i \in \{1, \ldots, M\}$. For DTLZ2-4, $\mathbf{w}$ is defined by $w_i = i/\sqrt{\sum_{j=1}^{M} j^2}, \forall i \in \{1, \ldots, M\}$. The unattainable reference point for each problem instance is specified by subtracting 0.1 from each component of $\mathbf{w}$, while the attainable reference point is specified by adding 0.1 to each component of $\mathbf{w}$.

4.2 Optimization Parameters

For crossover and mutation, both R-NSGA-II and g-NSGA-II use Simulated Binary Crossover (SBX) and polynomial mutation. Both algorithms use the same parameters for these operators. For SBX, the probability of crossover p_c is set to 0.9 and the distribution index η_c is set to 10. For polynomial mutation, the probability of mutation p_m is set to $1/n$, where n is the number of variables, and the distribution index η_m is set to 20. The population size is $P = 100$ in all cases. The evaluation budget for DTLZ1, DTLZ2 and DTLZ4 is set to $100 * P * M$, and for DTLZ3, due to its complexity, is set to $200 * P * M$.

5 RESULTS AND DISCUSSION

Each algorithm (R-NSGA-II and g-NSGA-II) is run 31 times on each problem instance (DTLZ1-4 with $M = \{3, 5, 10, 15, 20\}$) with unattainable and attainable reference points and the two performance metrics (EH-metric and R-HV) are calculated.

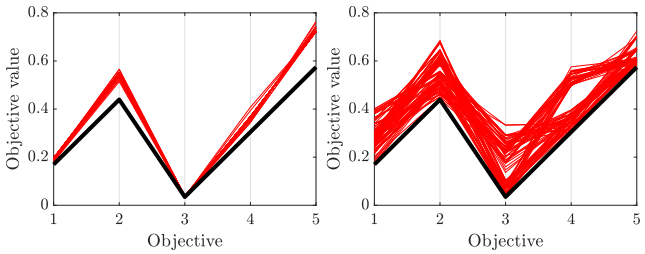
Table 1 shows the median values of the two metrics on various problem instances for the unattainable reference point mentioned above. According to the EH-metric, R-NSGA-II is significantly better than g-NSGA-II on all problem instances except DTLZ1 with $M = \{15, 20\}$ and DTLZ4 with $M = 3$, where the difference is not statistically significant.

Table 1: Median values of EH-metric and R-HV metric over 31 runs of R-NSGA-II and g-NSGA-II with an unattainable reference point.

		EH-Metric		R-HV	
	M	R-NSGA-II	g-NSGA-II	R-NSGA-II	g-NSGA-II
DTLZ1	3	**1.863E-01**	1.244E-01	7.076E+00	**7.816E+00**
	5	**1.934E+02**	1.384E+02	**2.569E+01**	0
	10	**1.262E+02**	9.225E+01	**7.154E+02**	0
	15	**7.036E+01**	5.097E+01	**1.431E+04**	0
	20	**4.634E+01**	3.065E+01	**3.117E+05**	0
DTLZ2	3	**2.139E-01**	1.404E-01	7.194E+00	**7.846E+00**
	5	**1.834E-01**	1.202E-01	2.745E+01	**3.042E+01**
	10	**2.161E-01**	1.422E-01	7.604E+02	**8.694E+02**
	15	**4.918E-01**	2.573E-01	**2.219E+04**	1.512E+04
	20	**3.600E-01**	2.991E-01	**1.164E+06**	5.146E+05
DTLZ3	3	**1.135E+00**	2.703E-01	**7.209E+00**	6.151E+00
	5	**1.048E+01**	3.736E+00	**2.755E+01**	0
	10	**4.583E+02**	2.278E+02	**7.571E+02**	0
	15	**4.572E+02**	2.625E+02	**2.621E+02**	0
	20	**3.646E+02**	2.002E+02	0	0
DTLZ4	3	**2.086E-01**	1.609E-01	**8.000E+00**	7.829E+00
	5	**3.422E-01**	2.860E-01	2.763E+01	**3.052E+01**
	10	**4.015E-01**	2.214E-01	9.464E+02	**1.058E+03**
	15	**6.064E-01**	3.117E-01	3.497E+02	**1.934E+04**
	20	**7.182E-01**	4.434E-01	**5.402E+02**	1.275E+01

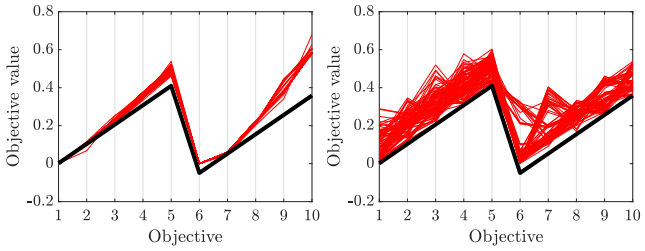
The better (higher) median value for each experiment is shown in bold for both metrics. A gray background denotes that the difference between the metric values of the two algorithms is statistically significant according to Wilcoxon's rank sum test at a significance level of 0.05.

The EH-metric in Table 1 agrees with R-HV metric for 12 of the problem instances in terms of the median values. However, for DTLZ1 with $M = 3$ and DTLZ2 with $M = \{3, 5, 10\}$, EH-metric and R-HV lead to different conclusions about the performance of R-NSGA-II and g-NSGA-II. While EH-metric indicates significantly better performance of R-NSGA-II, R-HV indicates significantly better performance of g-NSGA-II. We can further investigate these cases through visual assessment of the best preference set according to each metric obtained in each case. Figures 6a and 6b show the best preference sets for DTLZ2 with $M = 5$ objectives obtained by R-NSGA-II and g-NSGA-II respectively. For R-NSGA-II, we choose the best preference set according to EH-metric and for g-NSGA-II, we choose the best preference set according to R-HV. Clearly, the preference sets from R-NSGA-II agree better with the reference point (shown as a black line) than those from g-NSGA-II. The EH-metric is able to identify R-NSGA-II as significantly better than g-NSGA-II, whereas R-HV metric suggests the opposite.



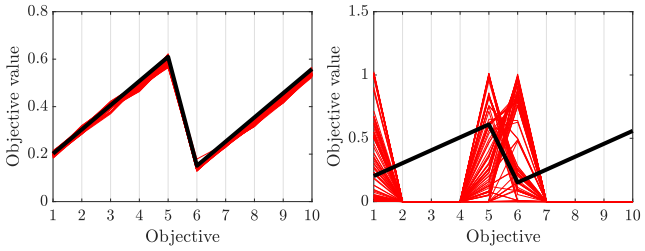
(a) R-NSGA-II (Best EH-metric) (b) g-NSGA-II (Best R-HV)

Figure 6: Best preference sets for DTLZ2 with $M = 5$.



(a) R-NSGA-II (Best EH-metric) (b) g-NSGA-II (Best R-HV)

Figure 7: Best preference sets for DTLZ2 with $M = 10$.



(a) R-NSGA-II (Best EH-metric) (b) g-NSGA-II (Best R-HV)

Figure 8: Best preference sets for DTLZ4 with $M = 10$.

Similarly, Figures 7a and 7b show the best preference sets for DTLZ2 with $M = 10$ objectives obtained by R-NSGA-II and g-NSGA-II respectively. Again, we observe that the algorithm identified to be better by the EH-metric conforms well with the reference point.

Table 2 shows the median values of the two metrics for the attainable reference point mentioned before. According to the EH-metric, R-NSGA-II is significantly better than g-NSGA-II on all problem instances except DTLZ1 with $M = 3$, where the difference is not statistically significant.

The EH-metric in Table 2 agrees with R-HV metric for 10 of the problem instances in terms of the median values. However, for DTLZ1 with $M = 3$, DTLZ2 with $M = \{3, 5, 10\}$ and DTLZ4 with $M = \{3, 5, 10\}$, EH-metric and R-HV again lead to different conclusions. We can perform a visual assessment as before. Figures 8a and 8b show the best preference sets for DTLZ4 with $M = 10$ objectives obtained by R-NSGA-II and g-NSGA-II respectively. It is

Table 2: Median values of EH-metric and R-HV metric over 31 runs of R-NSGA-II and g-NSGA-II with an attainable reference point.

		EH-Metric		R-HV	
	M	R-NSGA-II	g-NSGA-II	R-NSGA-II	g-NSGA-II
DTLZ1	3	**8.658E-02**	7.419E-02	9.437E+00	**1.042E+01**
	5	**3.948E-02**	0	**3.970E+01**	0
	10	**3.324E-02**	0	**1.247E+03**	0
	15	**3.566E-02**	0	**3.945E+04**	0
	20	**3.066E-02**	0	**1.208E+06**	0
DTLZ2	3	**2.255E-01**	1.447E-01	9.551E+00	**1.134E+01**
	5	**1.867E-01**	1.070E-01	4.028E+01	**5.363E+01**
	10	**5.723E-01**	1.360E-01	1.262E+03	**2.272E+03**
	15	**6.723E-01**	2.293E-01	4.029E+04	**8.681E+04**
	20	**6.741E-01**	2.310E-01	1.323E+06	**3.758E+06**
DTLZ3	3	**3.122E-02**	0	**9.402E+00**	0
	5	**4.020E-02**	0	**3.960E+01**	0
	10	**2.116E-02**	0	**1.135E+03**	0
	15	**3.308E-02**	0	**3.576E+04**	0
	20	**2.488E-02**	0	**1.170E+06**	0
DTLZ4	3	**2.241E-01**	1.387E-01	9.436E+00	**1.140E+01**
	5	**3.793E-01**	1.884E-01	4.021E+01	**5.572E+01**
	10	**8.339E-01**	1.728E-01	1.291E+03	**2.769E+03**
	15	**1.300E+00**	3.076E-01	4.117E+04	**8.929E+04**
	20	**1.486E+00**	4.262E-01	**1.302E+06**	3.269E+02

The better (higher) median value for each experiment is shown in bold for both metrics. A gray background denotes that the difference between the metric values of the two algorithms is statistically significant according to Wilcoxon's rank sum test at a significance level of 0.05.

easy to see here that R-NSGA-II is the better algorithm as correctly identified by the EH-metric.

6 CONCLUSIONS

While many reference-point based MOEAs have been proposed in the literature, there are very few performance assessment metrics for quantitatively comparing the algorithms. Our survey only yielded six such metrics, all of which rely on a DM-defined parameter for the size of region of interest (ROI). This parameter is not intuitive to set for an arbitrary problem, especially when many objectives are involved. Additionally, these six metrics involve other parameters that must also be set by the decision maker, making them difficult to be used in practice. Our goal in this paper was to propose and evaluate a parameterless performance metric called the Expanding Hypercube (EH) metric. It is defined as the area under the trade-off curve generated by expanding a hypercube from the reference point and plotting the number of enveloped points versus the size of the hypercube. The metric is easy to implement and inexpensive to compute. We demonstrated the effectiveness of the EH-metric through both constructed preference sets and experiments using two popular reference-point based MOEAs, namely R-NSGA-II and g-NSGA-II, on the first four DTLZ problems up to 20 objectives. The assessments from EH-metric and R-HV agree with each other for only about half of the problem instances. When their assessments differed, visualization of the preference sets revealed that the EH-metric is more consistent in identifying the better algorithm. We believe that in future the EH-metric can be extended to assess preference optimization methods that use other forms of preference articulation.

REFERENCES

[1] Slim Bechikh, Marouane Kessentini, Lamjed Ben Said, and Khaled Ghédira. 2015. Preference incorporation in evolutionary multiobjective optimization: a survey of the state-of-the-art. In *Advances in Computers*. Vol. 98. Elsevier, 141–207.

[2] Peter AN Bosman and Dirk Thierens. 2003. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE transactions on evolutionary computation* 7, 2 (2003), 174–188.

[3] Jürgen Branke. 2008. Consideration of partial user preferences in evolutionary multiobjective optimization. In *Multiobjective optimization - Interactive and Evolutionary Approaches*. Springer, 157–178.

[4] Kalyanmoy Deb and Abhay Kumar. 2007. Light beam search based multi-objective optimization using evolutionary algorithms.. In *IEEE Congress on Evolutionary Computation*. 2125–2132.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[6] Kalyanmoy Deb, Florian Siegmund, and Amos HC Ng. 2014. R-HV: A metric for computing hyper-volume for reference point based EMOs. In *International Conference on Swarm, Evolutionary, and Memetic Computing*. Springer, 98–110.

[7] Kalyanmoy Deb and J Sundar. 2006. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 635–642.

[8] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2005. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*. Springer, 105–145.

[9] CM Fonesca and Peter J Fleming. 1993. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the Fifth International Conference On Genetic Algorithms*. 415–423.

[10] Zhanglu Hou, Shengxiang Yang, Juan Zou, Jinhua Zheng, Guo Yu, and Gan Ruan. 2018. A performance indicator for reference-point-based multiobjective evolutionary optimization. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1571–1578.

[11] Ke Li, Kalyanmoy Deb, and Xin Yao. 2018. R-Metric: Evaluating the Performance of Preference-Based Evolutionary Multiobjective Optimization Using Reference Points. *IEEE Transactions on Evolutionary Computation* 22, 6 (2018), 821–835.

[12] Asad Mohammadi, Mohammad Nabi Omidvar, and Xiaodong Li. 2013. A new performance metric for user-preference based multi-objective evolutionary algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2825–2832.

[13] Julián Molina, Luis V Santana, Alfredo G Hernández-Díaz, Carlos A Coello Coello, and Rafael Caballero. 2009. g-dominance: Reference point based dominance for multiobjective metaheuristics. *European Journal of Operational Research* 197, 2 (2009), 685–692.

[14] Long Nguyen, Hung Nguyen Xuan, and Lam Thu Bui. 2015. Performance Measurement for Interactive Multi-objective Evolutionary Algorithms. In *Knowledge and Systems Engineering (KSE), 2015 Seventh International Conference on*. IEEE, 302–305.

[15] Lamjed Ben Said, Slim Bechikh, and Khaled Ghédira. 2010. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making. *IEEE Transactions on Evolutionary Computation* 14, 5 (2010), 801–818.

[16] Lothar Thiele, Kaisa Miettinen, Pekka J Korhonen, and Julian Molina. 2009. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation* 17, 3 (2009), 411–436.

[17] Handing Wang, Markus Olhofer, and Yaochu Jin. 2017. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex & Intelligent Systems* 3, 4 (2017), 233–245.

[18] Upali K Wickramasinghe, Robert Carrese, and Xiaodong Li. 2010. Designing airfoils using a reference point based evolutionary many-objective particle swarm optimization algorithm. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–8.

[19] Andrzej P Wierzbicki. 1986. On the completeness and constructiveness of parametric characterizations to vector optimization problems. *Operations-Research-Spektrum* 8, 2 (1986), 73–87.

[20] Guo Yu, Jinhua Zheng, and Xiaodong Li. 2015. An improved performance metric for multiobjective evolutionary algorithms with user preferences. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 908–915.

[21] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3, 4 (1999), 257–271.