Interactive Multiobjective Optimisation: Preference Changes and Algorithm Responsiveness

Kendall Taylor
School of Science, RMIT University
Melbourne, Victoria, Australia
s9004570@student.rmit.edu.au

Xiaodong Li
School of Science, RMIT University
Melbourne, Victoria, Australia
xiaodong.li@rmit.edu.au

ABSTRACT

For optimisation problems with multiple objectives and large search spaces, it may not be feasible to find all optimal solutions. Even if possible, a decision maker (DM) is only interested in a small number of these solutions. Incorporating a DM's solution preferences into the process reduces the problem's search space by focusing only on regions of interest. Allowing a DM to interact and alter their preferences during a single optimisation run facilitates learning and mistake correction, and improves the search for desired solutions. In this paper, we apply an interactive framework to four leading multi-objective evolutionary algorithms (MOEAs), which use reference points to model preferences. Furthermore, we propose a new performance metric for algorithm responsiveness to preference changes, and evaluate these algorithms using this metric. Interactive algorithms must respond to changes in DM preferences and we show how our new metric is able to differentiate between the four algorithms when run on the ZDT suite of test problems. Finally, we identify characteristics of these methods that determine their level of response to change.

CCS CONCEPTS

• **Applied computing → Multi-criterion optimization and decision-making**;

KEYWORDS

Interactive optimisation, Preference modeling, Multiobjective optimisation, Evolutionary computation

ACM Reference Format:
Kendall Taylor and Xiaodong Li. 2018. Interactive Multiobjective Optimisation: Preference Changes and Algorithm Responsiveness. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205624

1 INTRODUCTION

Any real-world optimisation problem will involve a Decision Maker (DM) who has a vested interest in solving the problem and implementing the results. In the case of a single objective problem with a single optimal solution, the DM has a simple choice to either accept the solution or reject it. However solving problems with multiple objectives involves finding a *set* of solutions representing the best trade-offs between conflicting objectives. The solution sets found by multiobjective optimisation algorithms can be large, but are usually presented to the DM regardless, who is then expected to select a small subset of solutions for further evaluation.

Aside from the computational waste of this *a posteriori* approach, a significant problem exists when a set of optimal solutions to a multiobjective problem is too vast for a DM to effectively choose from[3, 11]. *Preference-based* algorithms aim to ameliorate this problem by incorporating the DM's preferences and expert knowledge to reduce the search space and focus the search only on a preferred region of interest[5, 37]. While many preference-based methods involve the DM at the start (*a priori*), or at the end (*a posteriori*) of an algorithm's run, there is an increasing interest in eliciting preferences interactively throughout the optimisation process [11]. The advantages of an interactive method are particularly apparent when the DM is uncertain about their preferences and the interactive process is able to educate the DM and allow mistakes to be corrected [5, 37].

The ability of an optimisation algorithm to facilitate the correction of DM mistakes is a key attribute of interactive techniques [18, 33, 34]. However to respond to such corrections requires an algorithm to quickly re-focus on a new region of interest, without sacrificing convergence momentum. Further, an algorithm's sensitivity to preference change determines its suitability for interactive application. As a result, this work develops a new performance metric for interactivity which allows comparison of algorithm responsiveness.

To achieve this goal we first examine how algorithms adapt and respond to preference changes during optimisation. When a reference point based MOEA is interrupted and the DM changes the reference point, the problem becomes *dynamic* with one of the key inputs altered.

Dynamic multiobjective optimisation problems (DMOPs) contain constraints, objectives or parameters that change over time and potentially alter the problem's Pareto front[12, 30]. In the case of interactive reference point methods, the reference point is a discrete dynamic parameter which changes during optimisation. This change then guides the search toward solutions in a region of interest defined by the new point. Unlike a DMOP, the problem landscape and Pareto front are not altered in the case of a reference point change. However the need to respond to a changed environment applies to both, and the role of the diversity of the population in responding to change remains.

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205624

The ability of an interactive algorithm to respond to preference changes is important for the benefits of the technique to be realised. An objective measure is therefore required to assess and compare the relative performance of interactive MOEAs in responding to DM preference changes. Using such a metric, the characteristics of an algorithm's ability to respond can be identified. By assessing the performance using a series of standard benchmark problems we may find algorithmic traits that lead to better results.

This work provides a new performance metric for interactivity which allows comparison of algorithm responsiveness. Involving the DM during optimisation is becoming more popular as problems become more complex. An interactive MOEA needs to respond to a DM's change of preferences yet there is no current measure to determine how well a method responds to change. The accumulated hypervolume metric developed here assists in identifying the characteristics of different MOEAs and how they perform in an interactive, dynamic environment.

2 BACKGROUND

2.1 Multiobjective optimisation

MOEAs have had considerable success in solving optimisation problems where multiple objectives are in conflict. They use a stochastic approach which evolves a *population* of individuals (candidate solutions), and evaluates them with a *fitness* function. Each iteration or *generation* of this process seeks to find solutions that have 'better' fitness than previous ones. The optimisation process continues until a termination criteria is reached and a final (optimal), *set* of solutions found.

Due to an MOP's conflicting objectives and the resultant solutions representing trade-offs between objectives, the identification of a single 'best' solution cannot be made. Further, the concept of ordering these solutions in a meaningful way for the DM to chose from is difficult. To overcome this problem, most multi-objective optimisation algorithms use the concept of dominance to distinguish between optimality of solutions[6, 7].

A solution to a MOP is a vector of real-valued or discrete *decision variables* containing decision alternatives. Given a solution vector $\vec{x}$ with n decision variables $(x_1, x2, ..., x_n)$, and an *objective function* $f(\vec{x})$ that evaluates solutions, the general form of a MOP is formulated as[6, 7, 37]:

$$\text{Minimise/Maximise} \quad [f_1(x), ..., f_m(x)]; \tag{1}$$

$$\text{subject to} \quad x \in \Omega \tag{2}$$

where Ω is the *decision space* denoting the set of all possible solutions. The objective functions f_i map values from the decision space to the *objective space* $f_i : \Omega \rightarrow R$ where $i = 1, 2, ..., m$, R^m is the objective space and m is the number of objective functions. *Feasible* solutions within the decision space Ω are defined by a series of inequality constraints restricting values of the decision variable x_i. The set of all *feasible* solutions is called the *search space*.

2.2 Preference models

To represent DM information, preference models have been developed to quantify information so that they may be incorporated into the optimisation process. Some models require input from a DM *a priori*, while others can be used during optimisation in an interactive way, or to help select desirable solutions after optimisation (*a posteriori*).

The types of preference models differ in terms of the type of information required from the DM, and the type of information that is shown to the DM [21]. The use of weightings, trade-offs, pair-wise comparisons, outranking and thresholds are common approaches that perform well in eliciting DM preferences[2, 3]. Unfortunately for problems where three or more objectives need to be considered, the demands, or burden on the DM rapidly increases[2, 4, 17]. This is undesirable, and a compromise is required between a model's sophistication and its ability to scale to larger problems. The use of *reference points* can address this problem by using a simple point vector to represent a DM's desires or aspirations for each objective[2, 8, 17].

The *reference point* model[13, 14] has proven useful in multi-objective optimisation and involves the DM supplying a *reference point* comprising a value for each objective. The use of a reference point at the start of the optimisation process (*a priori*), focuses the initial search effort to the region of interest and significantly reduces computation. However this assumes that the DM has prior knowledge of the problem and knows where to start the search for their preferred solution. Where such knowledge is lacking, an interactive approach where the DM can adjust reference points *during* optimisation, allows the DM to explore the search space and learn about the problem and the inter-dependencies between objectives [21]. The increased involvement of the DM in the optimisation process can also encourage greater ownership and satisfaction of final solutions[10] and make selection of single solutions more effective.

2.3 Interactivity

Progressive incorporation of preferences has a number of advantages over *a priori* and *a posteriori* techniques[22, 34]. An interactive optimisation approach comprises a series of iterations whereby the DM specifies preferences progressively during optimisation, directing the search for desired solutions. Using this method only a part of the Pareto front needs to be explored and evaluated as the directed search focuses on the DM's preferred region. While reducing computation is beneficial, the key advantage of interactivity is that the DM can learn from the process by exploring the search space, refining their preferences and honing-in on preferred solutions. This also means that the DM does not require prior knowledge nor have any global preference structure before searching[22]. The DM's preferences can be changed during the search, reducing mis-direction by human error, and allowing greater search control [3, 33].

2.4 Performance metrics for preference based MOEAs

For preference based MOEAs which generate a sub-set of pareto optimal solutions, the use of standard performance metrics that are based on the complete Pareto front are not suitable. The problem lies with the need to define a preferred region of interest independent of the Pareto front. Mohammadi et al. proposes constructing a *composite front* from merged solution sets of compared MOEAs. A

preferred region of interest is defined on this front and existing MOEA performance metrics are applied to assess performance [23].

Nguyen et al. [29] suggest a similar approach defining a *User based Front* generated from a preferred region which then uses Generational Distance (GD) and Inverse Generational Distance (IGD) to measure performance [29]. More recently Li and Deb [17] have proposed a systematic method named *R-metric* which uses an approach of pre-processing the preferred optimal set and measuring performance using adapted Inverted Generational Distance (IGD) and Hypervolume (HV) unary metrics [17]. The advantage of the HV measure is its ability to both identify set dominance and reward solution diversity. However calculating HV can be computationally expensive especially when many solutions need to be evaluated[1].

The use of reference points and interactive preference based MOEAs is a relatively new field of investigation. Consequently, metrics to measure performance of these techniques are currently under-developed.

3 ACCUMULATED HYPERVOLUME

The new metric *AHv* is proposed here, based on the hypervolume indicator[35] but calculated using solutions contained in a region of interest rather than the whole Pareto Front. This use of a reduced set of solutions minimises computational complexity and allows HV to be determined and aggregated in a reasonable timeframe during optimisation.

At each generation of a run comprising two or more algorithms, a region of interest (or preferred region), is defined using non-dominated solutions from all algorithms. This composite front of solutions is then used to calculate a *nadir* point representing the worst objective values of the set. The hypervolume of an individual algorithm can then be calculated using this nadir point and the algorithm's non-dominated solutions that fall within the preferred region. Figure 1 illustrates this process by identifying a set of non-dominated solutions, located within a preferred region defined around a mid point solution closest to the reference point. The hypervolume of these solutions is determined by the sum of the areas constructed with reference to the nadir point.

Given a run with T generations in total, at each generation t (where $t = 1, 2, ..., T$), a new preferred region Pr_t is determined and the hypervolume HV of contained solutions is calculated. On completion of a full run, the individual hypervolume values for each generation are summed into an accumulated hypervolume metric (*AHv*):

$$AHv = \sum_{t=1}^{T} HV_t(R_t, nd_t) \tag{3}$$

where for each generation t we have a set R_t comprising non-dominated solutions located within the preferred region Pr_t, and a *nadir* point nd_t. The hypervolume HV_t is the union of the areas defined by the solutions in R_t and the nadir point nd_t, such that $HV_t = volume(\bigcup_{i=1}^{|R_t|} v_i)$, where i is a solution in R_t and v_i is a hypercube defined with reference to nd_t[20].

The premise is that solutions found using a reference point based MOEA will converge to the DM's region of interest and maximise its hypervolume indicator. If the reference point is changed during the optimisation process (after convergence), the hypervolume of the next generation will be minimal. As subsequent solutions are generated closer to the new reference point the hypervolume value will increase. *AHv* is at a minimum a binary metric requiring at least two algorithm solution sets as parameters. Following a reference point change, an algorithm that finds closer solutions faster will have a higher *AHv* than other algorithms run at the same time.

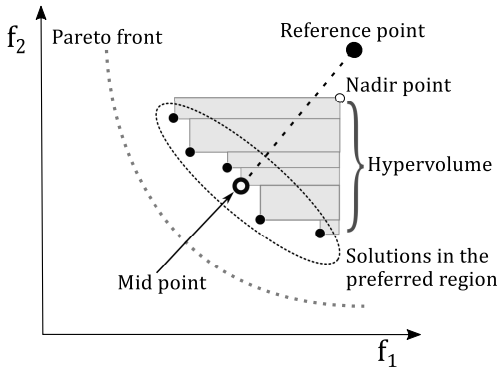

Figure 1: Preference Region hypervolume for a single generation

To identify a preferred region for a given generation of solutions we have used the metric developed by Mohammadi et al. [23] called the user-preference metric based on a composite front (UPCF). This metric is designed specifically for reference based MOEAs and creates a *preferred region* using a replacement Pareto Front called the *composite front*. The preferred region is centered on the closest solution in the composite front to the reference point, and defined by a radius determined by a user-specified parameter r. At the completion of each generation for all algorithms, the preferred region is determined using the following steps:

(1) Combine solutions from all algorithms into a single front (Composite front),
(2) Remove dominated solutions from the composite front,
(3) Find the solution in the Composite front that is closest to the reference point (ie. the mid point),
(4) Construct a preferred region centered on the mid point using a supplied *radius*.

3.1 Calculating *AHv* using the preferred region

Algorithm 1 provides an outline of inputs and the steps involved to implement to define the preferred region and calculate the *AHv* metric. At each generation of the algorithms' execution, a new preferred region is defined based on composite front of non-dominated solutions. This region is then used to calculate the hypervolume for each individual algorithm's results. The stages involved include the following:

(1) Assign a reference point and initialise all algorithms.
(2) For each algorithm, at the end of each generation calculate the composite front and preferred region.
(3) Calculate a *Nadir* point by finding the worst objective values of the composite front solutions contained in the preferred region.
(4) Determine which solutions from each algorithm fall within the region of interest.

(5) Calculate the hypervolume of solutions in the ROI for each algorithm individually using the generation's Nadir point.
(6) After termination or the required number of generations, sum the hypervolume values across all generations to produce an **accumulated hypervolume** for each algorithm.

Algorithm 1 *AHv* - Calculation of the accumulated hypervolume using multiple reference point changes

Inputs:
A: a set of reference point based MOEAs where $|A| \geq 2$;
R: a sequence of reference point vectors;
I: a set of generation counts indicating when to interact where $|I| = |R| - 1$ (as the first reference point in the sequence is used as the initial reference point).
max: the maximum number of generations before termination.

function CALCULATE$AHv(A,R,I,max)$
  for each algorithm $a \in A$ **do**
    initialize a
    $a_{ref} \leftarrow R_{first}$ ▷ set the initial reference point
  end for
  initialise t
  t $\leftarrow 1$
  while $t \leq max$ **do** ▷ begin evolving generations
    for each algorithm $a \in A$ **do**
      $a_{result} \leftarrow$ Evaluate and evolve population of a
    end for
    $allResults \leftarrow \{$ for all $a_{results}$ in $A\}$
    $cf \leftarrow$ BUILDCOMPOSITEFRONT$(allResults)$ ▷ also removes dominated solutions
    $pr \leftarrow$ FINDPREFERREDREGION(cf)
    $nd \leftarrow$ FINDNADIRPOINT$(cf \in pr)$
    for each algorithm $a \in A$ **do**
      $prs \leftarrow a_{results} \in pr$
      $a_{hv} \leftarrow$ HYPERVOLUME(prs, nd)
      $a_{ahv} \leftarrow a_{ahv} + a_{hv}$ ▷ accumulate each algorithm's hypervolume
    end for
    t^{++}
  end while
  return a_{ahv} for all $a \in A$
end function

The larger an algorithm's accumulated hypervolume, the better the method is at focusing on the reference point. Similarly, when the reference point is changed during an algorithm's run, a better responding algorithm will have a larger accumulated hypervolume as well. When the reference point location is changed it is expected that the hypervolume for the next generation should be lower than that before the change. The longer a reference point remains in the same spot the greater the convergence of solutions, therefore when the reference point is moved there should be few (if any) solutions close to the changed point and the hypervolume should decrease. Fast responding algorithms will be able to gain hypervolume quicker than slow responding methods.

4 EVALUATION

4.1 Reference point based multi-objective algorithms

Four reference point based MOEAs have been selected to evaluate this new metric. The first of these, R-NSGA-II is the earliest approach and the most popular having been successfully used in real-world scenarios[26] and large scale industrial systems simulation[32]. It is based upon the well known NSGA-II algorithm[9] and uses two fitness functions to select solutions, pareto optimality and crowding distance. The primary criteria is pareto optimality, however where dominance between solutions cannot be determined the crowding distance is used to ensure the selected set of solutions remains diverse [28]. The second (g-NSGA-II) and third (r-NSGA-II) are also based upon NSGA-II while the fourth (R-MEAD) extends the decomposition method MOEA/D[36]:

(1) *Preference distance (R-NSGA-II)*
R-NSGA-II replaces the crowding distance function of NSGA-II with a *preference distance* function which prefers solutions closer to the DM-supplied reference point[11]. Using the preference distance as a secondary criteria (while maintaining pareto optimality as the first), guides the algorithm to find pareto optimal solutions close to the reference point.
(2) *g-dominance (g-NSGA-II)*
This approach combines the notion of pareto efficiency with a reference point (representing DM aspiration levels), and finds solutions in a region of interest defined by that point[27]. The region of interest is determined by splitting the objective space into two regions and preferring solutions in the region that satisfy *all*, or *none* of the aspiration levels over those that satisfy only some levels[17, 18].
(3) *r-dominance (r-NSGA-II)*
Reference solution-based dominance (r-dominance) *adapts* the primary dominance relation of the NSGA-II to prefer solutions close to a DM-provided reference point while maintaining the order induced by pareto dominance[31]. The weighted Euclidean distance is used to determine the proximity of solutions to the reference point which is used to differentiate non-dominated solutions.
(4) *Reference point based decomposition (R-MEAD)*
In the base algorithm MOEA/D, weight vectors are generated to cover the entire pareto front; R-MEAD instead generates a smaller set of weights to find solutions in the region of interest. It achieves this by evolving an initial sub-optimal weight vector until a solution is found as close as possible to the reference point and Pareto Front. At each generation, neighbourhood weight vectors that are not close to the reference point are updated by moving them toward the region of interest[25].

While the first three are built on NSGA-II, they each replace or add to the method's dominance functions with their own dominance relation based on location relative to a reference point. r-NSGA-II and R-NSGA-II are very similar algorithms but differ in how they incorporate their distance-based fitness functions. R-NSGA-II can also use multiple reference points but this feature is not relevant in this work. R-MEAD[25] is an adaption of MOEA/D whereby subproblem weights are generated around the region of interest. Consequently we have two algorithm classes, domination-based and decomposition-based.

4.2 Experimental setup

Using the MOEA Java Framework[16] the base algorithms NSGA-II and MOEA/D were modified into the four reference point based algorithms. Each algorithm was adapted to allow the reference point to be changed interactively during execution.

To assess responsiveness, we provide a case study using a series of four reference points created for each problem in the ZDT suite

Table 1: Experimental setup - base parameters

Setting	Value(s)				
Algorithms (4)	R-NSGA-II, g-NSGA-II, r-NSGA-II, R-MEAD				
Problems (5)	ZDT1, ZDT2, ZDT3, ZDT4, ZDT6				
Number of runs	30				
Generations	600				
Population size	100				
Radius of pref. region	0.1				
Reference Points	ZDT1	0.9, 0.1	0.3, 0.7	0.77, 0.06	0.4, 0.2
	ZDT2	0.1, 0.9	0.75, 0.3	0.1, 1.1	0.8, 0.5
	ZDT3	0.0, 0.8	0.4, -0.1	0.5, -0.25	0.8, -0.25
	ZDT4	0.05, 0.75	0.8, 0.2	0.8, 0.0	0.4, 0.6
	ZDT6	1.0, 0.4	0.7, 0.6	0.9, 0.1	0.8, 0.5
Interactions (3) at	150, 300, and 450 generations				

Table 2: Algorithm parameters used for evaluations

R-NSGA-II, g-NSGA-II, and r-NSGA-II		R-MEAD	
Parameter	Value	Parameter	Value
SBX rate	1.0	Crossover rate	0.1
SBX dist. index	15.0	Step size	0.5
PM rate	0.03	Neighbourhood size	10
PM dist. index	20	delta	0.9
		eta	0.01

(see Table 1). The first point is used as the initial point, and during an optimisation run the reference point is changed three times. The set of reference points represent a mix of feasible and infeasible locations and care was taken that sequences were not too close together so as to highlight responsive ability.

The parameter settings presented in Table 2 reflect the default values used in the MOEA Framework taken from best practices in relevant literature[15]. In addition to these settings, r-NSGA-II has a parameter δ (set to 0.5) to control the spread of solutions in the region of interest, and R-NSGA-II has the parameter $\epsilon - clearing$ which controls the extent of solutions found (set to 0.008). No archive populations were used, nor were any algorithms allowed to re-start during a run.

5 RESULTS

5.1 Response to preference changes

Using hypervolume calculated at each generation we can explore the differing abilities of algorithms to respond when reference points change. Algorithms were run for 600 generations with reference point changes at 150, 300, and 450 generations. This number of generations is large but necessary to allow all methods enough time to converge to a region of interest. Convergence is important before a reference point change occurs as dispersed solutions may be serendipitously located close to the new reference point providing an unfair advantage to a unresponsive algorithm.

Following 30 runs of each algorithm, the hypervolume for each generation is averaged, plotted and compared. Figure 2a displays results from runs using the ZDT1 problem with the four interactive reference point algorithms. An initial reference point is set at generation zero, and changes made at intervals of 150 generations until termination at 600. It is clear that dramatic changes to each algorithm's hypervolume occurred when the reference point was changed.

The first thing to note in Figure 2a is the similarity in generational hypervolumes of the r-NSGA-II and R-NSGA-II algorithms. Both of these algorithms obtained large hypervolumes when they converged, and experienced a very steep drop-off at each reference point change. They also managed to re-converge at new regions of interest within a reasonable number of generations. The solutions sets found by R-MEAD were generally less stable than the other algorithms and this is reflected in its lower hypervolume values. Nonetheless, R-MEAD appears to respond to change faster than the other methods, picking up hypervolume in fewer generations after reference point changes.

The behaviour of g-NSGA-II differed from the other algorithms on the ZDT1 problem where the method appears to converge toward the initial reference point, but does not respond to the first reference point change at generation 150. g-NSGA-II did not generate any solutions within the preferred region after this change and did not regain hypervolume until the next change when previous solutions became relevant again. It is not until the final change that the algorithm appears to respond and then it is very slow compared to the other methods. Further investigation reveals g-NSGA-II is much slower to respond to change and appears to become 'stuck' for many generations even on simple problems such as ZDT1.

When run using the ZDT2 problem (see Figure 2b), results were very similar to ZDT1 with R-NSGA-II and r-NSGA-II performing similarly well and R-MEAD responding quickly to reference point changes. However on the ZDT3 problem with a discontinuous Pareto Front, R-MEAD responded poorly. Figure 2c shows how R-MEAD failed to find solutions close to the initial reference point, and failed to respond after the final change at generation 450. Again both r-NSGA-II and R-NSGA-II responded to all changes but with weaker performance with the last two changes. This problem presents obvious difficulties for our algorithms, responding and changing focus once converged is more difficult, due to the discontinuous nature of the Pareto Front and the lack of diversity within the solution set.

The ZDT4 problem (see figure 2d) contains many local Pareto fronts and is considered a difficult problem with algorithms easily becoming stuck by sub-optimal fronts. While g-NSGA-II performed well with the initial and last reference points, it failed to respond to the three changes in between. r-NSGA-II clearly outperformed the other methods including the state-of-the-art, yet similar to R-NSGA-II. Unlike R-NSGA-II, r-NSGA-II retains the crowding distance function to help maintain diverse solutions. This feature appears to allow r-NSGA-II to look beyond the many local fronts and find solutions closer to the Pareto front. ZDT6 (see figure 2e) is also considered a difficult problem with R-NSGA-II, r-NSGA-II and g-NSGA-II all performing similarly. The main point of note however was R-MEAD's ability to rapidly gain hypervolume and find solutions in the preferred region in the initial stages of the runs. This quick response to change was R-MEAD's strength across most tests even though its overall convergence was weaker than the other methods.

5.2 Measuring responsiveness

Given an interactive reference point based MOEA where reference points are changed during optimisation, the hypervolume from each

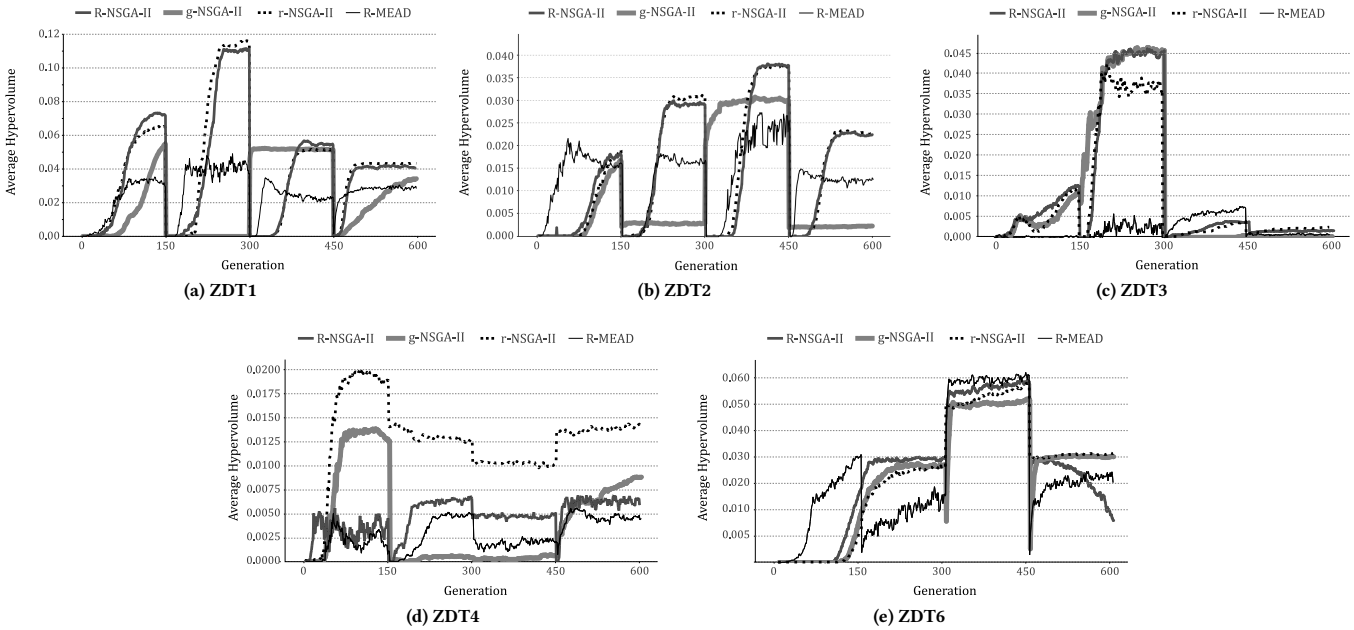

Figure 2: Average hypervolume by generation - 30 runs with problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6.

generation during a run is summed as the *AHv* metric (accumulated hypervolume). Assuming the algorithm responds to change, the larger the *AHv* the more responsive the algorithm is to interaction.

Averaging the *AHv* over 30 runs on problem ZDT1 reveals R-NSGA-II as the algorithm with the highest value, although the difference between it and r-NSGA-II is minor (see Table 3). The box plot of results from ZDT1 (Figure 3a), reveals the *AHv* of g-NSGA-II to be left-skewed with larger variability than the other algorithms. The distribution of results for r-NSGA-II, R-NSGA-II and R-MEAD were more compact with considerable overlap between r-NSGA-II and R-NSGA-II.

The *t*-test results for *AHv* when running the ZDT2 problem show that g-NSGA-II has the only statistically significant difference when compared with each of the other methods. Figure 3b shows that the distribution of values returned by g-NSGA-II was also more dispersed and it had the lowest median. The performance of R-MEAD increased with this problem compared with ZDT1 with results similar to both r-NSGA-II and R-NSGA-II.

Running on the ZDT3 problem the R-MEAD algorithm performed consistently poorly with the lowest mean (1.22), while R-NSGA-II achieved the highest (6.78), although only slightly larger than g-NSGA-II's (6.59) (see figure 3c) .

Figure 3d shows the box plot of results for the ZDT4 problem. The first point of note is that all methods have a minimum *AHv* of zero or close to it, indicative of the difficultly in finding solutions close to the reference point. The *AHv* results from r-NSGA-II were significantly different to R-NSGA-II with a *p-value* of 0.0012. As with all the algorithms on this problem, r-NSGA-II's results are skewed to the right, but with much higher variability and greater mean.

Table 3: Accumulated Hypervolume over 30 runs - means and standard deviations

	Mean and (standard deviation)			
Problem	R-NSGA-II	g-NSGA-II	r-NSGA-II	R-MEAD
ZDT1	24.94 (1.416)	19.60 (2.898)	24.83 (1.201)	17.20 (0.957)
ZDT2	8.71 (1.409)	5.86 (2.253)	8.86 (1.277)	8.21 (1.818)
ZDT3	6.78 (0.924)	6.59 (0.766)	5.68 (0.875)	1.22 (0.366)
ZDT4	1.70 (3.687)	2.47 (3.305)	7.61 (8.534)	2.83 (4.883)
ZDT6	18.28 (0.868)	17.30 (0.609)	17.58 (0.467)	17.55 (2.679)

Table 4: Accumulated Hypervolume over 30 runs. The t-test *p*-values at the 0.05 significance level for R-NSGA-II compared with other algorithms.

	p-values for R-NSGA-II compared to:		
Problem	g-NSGA-II	r-NSGA-II	R-MEAD
ZDT1	2.5337E-11	0.7394	6.8989E-27
ZDT2	4.3677E-6	0.6710	0.1126
ZDT3	0.3786	9.1183E-5	4.9503E-37
ZDT4	0.3945	0.0012	0.3131
ZDT6	4.8383E-12	7.3623E-8	0.1592

Results from the final problem in the ZDT series, ZDT6, reveal high variability of *AHv* values for R-MEAD and an inter-quartile range that overlaps that of all comparison algorithms (see figure

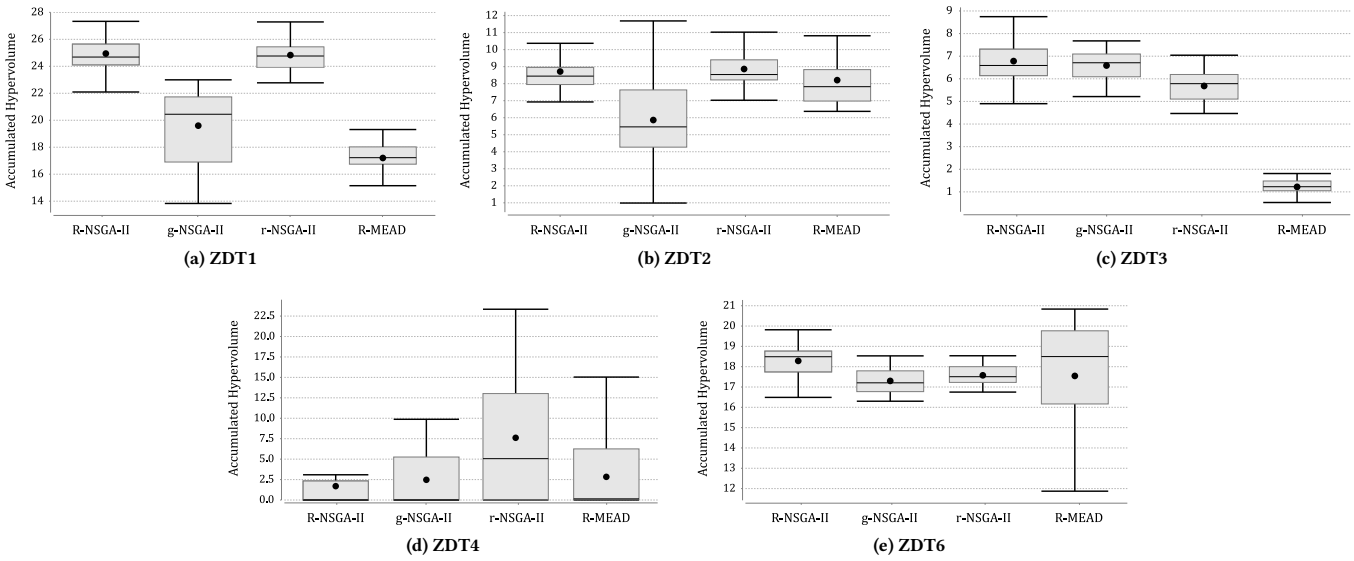

Figure 3: Box plots of accumulated hypervolume (*AHv*), over 30 runs with problems ZDT2, ZDT3, ZDT4 and ZDT6.

3e) . Consequently, R-MEAD's results are not significantly different to the other methods at the 95% confidence level. R-NSGA-II is however significantly different to both g-NSGA-II and r-NSGA-II.

Overall r-NSGA-II and R-NSGA-II responded more consistently across the ZDT problem set. r-NSGA-II had better performance on ZDT2 and ZDT4 and R-NSGA-II showed strength on ZDT3 and ZDT6, both were very similar on ZDT1. Obviously the type of problem has a significant effect upon an algorithm's ability to respond to preference changes.

5.3 Behaviour analysis

All four algorithms have been shown to respond to preference changes during optimisation, however their level of responsiveness varied. When tested on the ZDT problems the g-dominance algorithm (g-NSGA-II) converged toward the initial reference point but responded poorly to subsequent changes. Figure 2a shows how g-NSGA-II did not increase its hypervolume after the change at 150 generations until the change at 300 generations where its hypervolume rose immediately and stabilised. This behaviour was unique to g-NSGA-II and experienced to varying degrees with all test problems (see also Figure 2b), and can be attributed to slow response to reference point changes and subsequent poor convergence to the new area of interest. As a result, this algorithm is probably more suited to *a priori* methods of interaction as convergence to an initial point was consistent.

The superior performance of the other NSGA-II based algorithms, r-NSGA-II and R-NSGA-II show that the problem with g-NSGA-II is with its dominance relation and a lack of selection pressure following preference changes. Both r-NSGA-II and R-NSGA-II responded well to change and converged toward new regions of interest rapidly on both the ZDT1 and ZDT2 problems. On the more difficult ZDT3, ZDT4 and ZDT6 problems all methods struggled to respond fast enough and could not converge to new regions before the reference point was changed again.

R-MEAD did not converge as well as r-NSGA-II and R-NSGA-II and its approximation sets were less stable between generations than all other algorithms. R-MEAD was however the quickest to respond to reference point changes. Figures 2a, 2b and 2e illustrate this where on each change in reference point (at generations 150, 300, and 450) R-MEAD clearly gains hypervolume faster than the state-of-the-art R-NSGA-II algorithm. Following a change in reference point, R-MEAD's hypervolume drops immediately and in few generations rises considerably.

Ignoring the quality of solutions, R-MEAD is a fast responder to change and can rapidly explore a problem's search space. This may be attributable to the fact that decomposition-based methods generally converge faster than dominance-based algorithms like NSGA-II.

6 CONCLUSIONS

An algorithm that is responsive to preference changes facilitates exploration and learning while allowing for 'mistakes' or misdirections to be corrected. In order to compare responsiveness of interactive algorithms we have developed a new performance metric. This new metric is called *AHv* and is an aggregation of the hypervolume calculated for each generation of a MOEA run. Using *AHv* we have compared four reference point based MOEAs with problems from the ZDT series. Not only does the metric allow us to determine how well an algorithm responds to reference point changes, but due to the nature of the hypervolume metric a larger value also indicates better solution quality in terms of convergence and diversity.

Evaluations reveal how well the selected reference based algorithms perform and their ability to adapt and respond to preference

changes. As expected, there was variability found amongst all methods; some such as g-NSGA-II were slower to adapt, while others like R-MEAD were fast but with poor convergence. The most consistent performers were r-NSGA-II and the state of the art R-NSGA-II.

A number of characteristics have been identified in the algorithms chosen for testing. The level of responsiveness varied with g-NSGA-II found to be the slowest responder. It was determined that the dominance relation was responsible as other NSGA-II based methods performed much better. In fact, the other two NSGA-II based algorithms r-NSGA-II and R-NSGA-II consistently performed the best. The decomposition algorithm R-MEAD responded very quickly to changes in the reference point but did not converge as well as r-NSGA-II and R-NSGA-II.

The obvious extension to this work is the evaluation of the four interactive algorithms with problems of higher dimensionality. It is expected that the performance metric will scale to more objectives, however the performance of different algorithms to respond to change with high dimension problems is unknown. The evaluation with the new metric of other decomposition based algorithms such as R-MEAD2[24] and MOEA/D-STM[19] would also help identify the relative strengths of dominance versus decomposition based techniques. Open questions remain with regard to optimal interactivity patterns. Little work has been conducted on the frequency of interactions during optimisation, nor the best times to interrupt the process to allow the DM to inspect results and update their preferences.

REFERENCES

[1] JM Bader, Eckart Zitzler, and Gunter Rudolph. 2010. Hypervolume-based search for multiobjective optimization: theory and methods. *Optimization* (2010), 1–310.

[2] Slim Bechikh, Marouane Kessentini, Lamjed Ben Said, and Khaled Ghedira. 2015. Preference Incorporation in Evolutionary Multiobjective Optimization: A Survey of the State-of-the-Art. In *Advances in Computers*. 141–207.

[3] Jurgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowinski (Eds.). 2008. *Multiobjective Optimization - Interactive and Evolutionary Approaches*. Vol. 5252 LNCS. Springer-Verlag.

[4] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2016. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* (2016).

[5] Daniel Cinalli, Luis Martí, A C B Garcia, Daniel Cinalli, Luis Martí, and Ana Cristina Bicharra. 2015. Collaborative preferences in multi-objective evolutionary algorithms . Collective Preferences in Evolutionary Multi-Objective Optimization : Techniques and Potential Contributions of Collective Intelligence. November (2015).

[6] Carlos A Coello Coello, Gary B Lamont, and David a Van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 800 pages.

[7] Kalyanmoy. Deb. 2001. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons.

[8] Kalyanmoy Deb and Abhishek Kumar. 2007. Interactive evolutionary multi-objective optimization and decision-making using reference direction method. *Proceedings of the 9th annual conference on Genetic and evolutionary computation GECCO 07* (2007), 781.

[9] 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.

[10] Kalyanmoy Deb, Ankur Sinha, Pekka J. Korhonen, and Jyrki Wallenius. 2010. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *Trans. Evol. Comp* 14, 5 (oct 2010), 723–739.

[11] Kalyanmoy Deb, J Sundar, Udaya Bhaskara Rao N, and Shamik Chaudhuri. 2006. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. 2, 3 (2006), 273–286.

[12] M. Farina, K. Deb, and P. Amato. 2004. Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications. *IEEE Transactions on Evolutionary Computation* 8, 5 (oct 2004), 425–442.

[13] Carlos M Fonseca and Peter J Fleming. 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Icga* 93, July (1993), 416–423.

[14] Carlos M. Fonseca and Peter J. Fleming. 1998. Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*. 28, 1 (1998), 26–37.

[15] D Hadka. 2016. *Beginner's Guide to the MOEA Framework*. CreateSpace Independent Publishing Platform.

[16] David Hadka. 2016. MOEA Framework, a Java library for multiobjective evolutionary algorithms. (2016).

[17] Ke Li and Kalyanmoy Deb. 2016. Performance Assessment for Preference-Based Evolutionary Multi-Objective Optimization Using Reference Points. (2016), 1–23.

[18] Ke Li, Kalyanmoy Deb, and Yao Xin. 2017. Integration of Preferences in Decomposition. (2017), 1–25.

[19] Ke Li, Qingfu Zhang, and Sam Kwong. 2014. Stable Matching-Based Selection in Evolutionary. 18, 6 (2014), 909–923.

[20] Xiaodong Li, Jurgen Jürgen Branke, and Michael Kirley. 2007. On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems. *Proc.IEEE CEC* (2007), 576–583.

[21] Mariano Luque, Kaisa Miettinen, Petri Eskelinen, and Francisco Ruiz. 2009. Incorporating preference information in interactive reference point methods for multiobjective optimization. 37 (2009), 450–462.

[22] Kaisa Miettinen, Francisco Ruiz, and Andrzej P. Wierzbicki. 2008. Introduction to multiobjective optimization: Interactive approaches. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 27–57.

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

[24] Asad Mohammadi, Mohammad Nabi Omidvar, Xiaodong Li, and Kalyanmoy Deb. 2014. Integrating User Preferences and Decomposition methods for Many-objective Optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. 421–428.

[25] A Mohammadi, M N Omidvar, and X D Li. 2012. Reference Point Based Multi-objective Optimization Through Decomposition. *2012 IEEE Congress on Evolutionary Computation* (2012), 1–8.

[26] Asad Mohammadpour, Abdolreza Dehghani Tafti, and Zaki Byagowi. 2013. Using R âĂŘ NSGA âĂŘ II in the Transmission Expansion Planning Problem for a Deregulated Power System with Wind Farms. *International Journal of Engineering Practical Research* 2, 4 (2013), 201–204.

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

[28] Amos H C Ng. 2012. Proceedings of the 2012 Winter Simulation Conference C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds. (2012), 1455–1465.

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

[30] N Riquelme, C Von Lucken, and B Baran. 2015. Performance metrics in multi-objective optimization. *2015 Latin American Computing Conference (CLEI)* 1 (2015), 1–11.

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

[32] Florian Siegmund, Jacob Bernedixen, Leif Pehrsson, Amos H. C. Ng, and Kalyanmoy Deb. 2012. Reference point-based evolutionary multi-objective optimization for industrial systems simulation. In *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, Berlin, Germany, 130:1—-130:11.

[33] Ankur Sinha. 2011. *Progressively Interactive Evolutionary Multiobjective Optimization*.

[34] Lothar Thiele, Lothar Thiele, Kaisa Miettinen, Kaisa Miettinen, Pekka J. Korhonen, Pekka J. Korhonen, Julian Molina, and Julian Molina. 2009. A Preference-Based Evolutionary Algorithm for Multi-Objective Optimization. *Evolutionary Computation* 17, 3 (sep 2009), 411–436.

[35] D.A. Van Veldhuizen and G.B. Lamont. 2000. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, Vol. 1. IEEE, 204–211.

[36] Q. Zhang and Hui Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731.

[37] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1, 1 (2011), 32–49.