Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

A Comparison of Different Algorithms for the Calculation of Dominated Hypervolumes

Christopher Priester, Kaname Narukawa, Tobias Rodemann

2013

Preprint:

This is an accepted article published in Genetic and Evolutionary Computation Conf. (GECCO). The final authenticated version is available online at: https://doi.org/[DOI not available]

A Comparison of Different Algorithms for the Calculation of Dominated Hypervolumes

Christopher Priester
Institute for Condensed Matter Physics
Technische Universität Darmstadt
Darmstadt, Germany
christopher@fkp.tu-darmstadt.de

Kaname Narukawa
Honda Research Institute Europe
Offenbach/Main
Germany
kaname.narukawa@honda-ri.de

Tobias Rodemann
Honda Research Institute Europe
Offenbach/Main
Germany
tobias.rodemann@honda-ri.de

ABSTRACT

In the fields of multi- and many-objective optimization methods, the hypervolume of a set of solutions is a very useful measure for assessing the current state of the optimization process. It is also the fundamental quality criterion for the well-known SMS-EMOA (S-metric selection evolutionary multi-objective optimization), which is one of the best many-objective optimization algorithms known at the moment. Unfortunately, the computation of the hypervolume for a given set of solutions is a time-consuming effort which scales unfavorably with the number of objectives and the size of the population. In this work we analyzed a number of algorithms for hypervolume computation and systematically measured their computational effort for different numbers of objectives and population size. We compared three established standard algorithms that are used in the Shark optimization library and a recent approach by While et al.. We also included an approximation computation algorithm proposed by Ishibuchi et al., where we additionally evaluated the precision of the approximation computation and its impact on the selection process within an optimization run. Our findings indicate that the algorithm by While et al. outperforms the three other exact algorithms for a wide range of settings. The Ishibuchi algorithm was shown to have a slightly negative effect on the selection process, but for very large population sizes or number of objectives, the approximation method might be the only viable alternative.

General Terms

Algorithms; Performance

Keywords

Dominated Hypervolume; Many-Objective Optimization; Computational Performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.
Copyright 2013 ACM 978-1-4503-1963-8/13/07 ...$15.00.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search: Heuristic methods

1. INTRODUCTION

With an increasing complexity of systems to be optimized the number of objectives to consider is constantly increasing. A recent example [10] presents an optimization of airplanes considering 10 objectives (which the authors scaled down from an original 30). It has, from an application point of view, become relevant to investigate many-objective optimization methods for problems with substantially more than two or three objectives.

In a recent paper [6] we compared a number of many-objective optimization (MAO) algorithms for a real-world optimization problem with 7 objectives. We found that the best performing algorithm was the SMS-EMOA [1]. The substantial performance advantage compared to other methods like CDAS (Controlling Dominance Area of Solutions) [9] or MOEA/D [13] (Multiobjective Evolutionary Algorithm Based on Decomposition) comes at the prize of a drastic increase in computation time of the optimization method itself. The reason is the computationally expensive hypervolume calculation. In our application we found that for a standard implementation using the hypervolume algorithm in the Shark[1] machine learning library already a small population of 100 individuals and 7 objectives requires more time to compute the hypervolume than the fitness computation. We therefore found that for many real-world applications faster algorithms for hypervolume computation are necessary in order to employ the very efficient SMS-EMOA.

The quantity investigated in this paper is the dominated hypervolume or S-metric. This scalar variable measures the hypervolume dominated between a set of solutions and a reference point. The selection is performed by replacing the solution which contributes least to the hypervolume of the population.

We compare a number of exact hypervolume calculation algorithms, two from the Shark library version 2.3, one from version 3.0, and a newly proposed method by While et al. [12]. We also compare the results to an approximative algorithm recently proposed by Ishibuchi et al. [5].

[1]http://shark-project.sourceforge.net

For all algorithms we compute the total calculation time for different values of population size (number of data points) and number of objectives (dimension of objective space). For the approximative algorithm we define a measure of similarity to results of the exact algorithm and also investigate the impact of the approximation error on the optimization performance.

2. COMPARED ALGORITHMS

In this section we shortly outline the employed hypervolume calculation algorithms. For details we refer to the respective publications.

2.1 Overmars

The algorithm proposed by Overmars and Yap [7] was originally designed to solve the so-called "Klee's measure problem" which is the problem of calculating the union of a set of rectangular intervals in arbitrary dimension. Since the calculation of the hypervolume dominated by a set of solutions is closely related to this problem, the algorithm can be applied to both problems. It represents a dimension-sweep approach. The d-dimensional space is cut into $(d-1)$-dimensional slices along an arbitrarily chosen axis x_i. For each solution the coordinate p_i marks the end of a slice and the beginning of the next one. The measure (hypervolume) of a slice in d dimensions can then be calculated using the $(d-1)$-dimensional measure at the beginning of the slice and the extent of the slice along the x_i-axis given by the coordinates of the solutions limiting it. The measure in the remaining $(d-1)$ dimensions is calculated by dividing the space using a partition tree. Each leaf of the tree represents a cell which is a part of the $(d-1)$-dimensional space. All cells are disjoint and their union forms the whole $(d-1)$-dimensional space. Furthermore the cells are chosen in a way such that each cell only contains trellises of the hypervolumes of which the measure is to be calculated. The measure of these trellises can be calculated easily which then allows the calculation of the whole hypervolume by summing over the leaves of the partition tree. The implementation by Overmars and Yap achieves a time-complexity of $\mathcal{O}\left(n \log n + n^{\frac{d}{2}} \cdot \log n\right)$ where d is the number of objectives and n is the number of points spanning the hypervolume. We took the implementation from the Shark 2.3 library.

2.2 Fonseca

The algorithm proposed by Fonseca et al. [4] benefit from an algorithm by Paquete et al. [8] which has a good worst-case complexity of $\mathcal{O}(n \log n)$ for hypervolume calculations in $d = 3$. This allows to omit one level of recursion in the dimension-sweeping compared to algorithms with $d = 2$ as lowest dimension to calculate the measure of a slice. The dimension-sweep for dimensions above three is performed using circular doubly-linked lists, one for each dimension. Each of these lists consists of all points, sorted by their coordinate in the respective dimension. Starting from the maximum coordinate in direction d the list is iterated in a descending order. All occurring points get deleted out of all lists until the minimum coordinate along the d-axis is reached. This procedure is repeated with the remaining points along each of the remaining axes until $d = 3$ is reached where the measure of the remaining point(s) is calculated. Then the iterations through the lists are performed in reverse order. Points are inserted and the measures of their corresponding slices is calculated by multiplying the lower-dimensional measure with the extent of the slice in the respectively next higher dimension. With this method Fonseca et al. achieve a time-complexity of $\mathcal{O}\left(n^{d-2} \log n\right)$.

2.3 Beume

The latest version 3.0 of the Shark library[2] features a hypervolume computation algorithm by Beume and Rudolph [2]. It is an improved version of the algorithm by Overmars and Yap exploiting the difference between Klee's measure problem and hypervolume calculation in the context of SMS-EMOA. In the latter, all rectangular hypervolumes spanned by the solutions in the population share the reference point as limiting point. Beume and Rudolph are able to save a factor of $\mathcal{O}(\log n)$ in the complexity compared to Overmars and Yap, resulting in a complexity of $\mathcal{O}\left(n \log n + n^{\frac{d}{2}}\right)$.

2.4 While

Recently, While et al. [12] proposed an algorithm for the fast calculation of exact hypervolumes. We used the authors code which is freely available from their website[3]. The approach is based on two concepts:

1. The hypervolume dominated by any set of solutions can be calculated as sum of exclusive hypervolumes starting from the hypervolume of a single solution, adding the hypervolume exclusively dominated by a second solution (not by the first one) and so forth.

2. Only few solutions are needed to calculate exclusive hypervolumes. Parts of the objective space that are not dominated by a solution $\mathbf{x}$ whose exclusive hypervolume is to be calculated can not contribute to that hypervolume. Therefore all other solutions except $\mathbf{x}$ can be shifted so none of their coordinates exceeds (for a maximization problem) the respective coordinate of $\mathbf{x}$. This can cause points to become dominated which allows to neglect these dominated points during the calculation of the exclusive hypervolume.

Apart from these concepts, the algorithm is also based on dimension-sweeping and allows the usage of different heuristics to improve its performance. Due to the dependency on the shape of the front there is no closed formula for the complexity of the algorithm. In their publication [12] While et al. prove that the worst-case complexity is exponential.

2.5 Ishibuchi

An approximative approach for hypervolume computation was proposed by Ishibuchi et al. [5]. It uses the achievement scalarizing function

$$g(\mathbf{x}, \mathbf{r}, \mathbf{w}) = \max_{i=1,\ldots,d} w_i \left(f_i(\mathbf{x}) - r_i\right) \quad . \tag{1}$$

The minimum $\min_{\mathbf{x} \in S} g(\mathbf{x}, \mathbf{r}, \mathbf{w})$ is used to calculate the distance between the reference point $\mathbf{r}$ and the attainment surface limiting the hypervolume in the objective space dominated by a solution set $S = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^n\right\}$ in a certain direction (defined by the weights $w_1, \ldots, w_d$). If a non-dominated

[2] http://image.diku.dk/shark/

[3] http://www.wfg.csse.uwa.edu.au/

solution is removed from the solution set, a part of the attainment surface is shifted closer to the reference point. The magnitude of this shift, that is the difference of the distances between the attainment surface and the reference point once taking the whole solution set and once excluding a single solution, is a measure for the contribution of the respective solution to the whole dominated hypervolume. This contribution of solution $\mathbf{x}$ can be calculated by setting the weight-vector as $w_i = \frac{1}{r_i - x_i}$ and calculating the distance between the attainment surface and the reference point using the achievement scalarizing function once taking the minimum over the complete solution set S and once using $S \backslash \mathbf{x}$. Since this only captures the shift along a single direction which might be biased, it is possible to use several weight vectors distributed around the central weight $w_i = \frac{1}{r_i - x_i}$ and take the average over the resulting shifts. This of course increases the computation time of the algorithm. We used our own implementation of the algorithm using the description in the publication by Ishibuchi et al.[5] and following them as closely as possible.

3. RESULTS

We tested the algorithms described in the last section using two different kinds of tests. Firstly we measured the computation time by calculating the hypervolume contributions of solutions in artificially created sets. Secondly we took the fastest exact algorithm and the approximative algorithm to compare the performance of the algorithms when used within an SMS-EMOA. Of course the results will depend on the actual implementation of each algorithm and there is no guarantee that the implementations we use are optimal. However, since all implementations except for the algorithm by Ishibuchi et. al. are obtained either directly from the authors or from shark library, these should be the most commonly used implementations. Furthermore, the important differences in performance to be shown are high enough that possible deviations from different implementations can be neglected.

3.1 Systematic Benchmarks

We performed an extensive set of tests of the algorithms for a variety of configurations. We tested for population sizes between 50 and 1000 solutions ranging from 3 up to 14 objectives. For the algorithms included in the Shark library, whose performance does not depend on the shape of the Pareto front, the solutions were distributed randomly in objective space, but for the algorithms by While et al. and Ishibuchi et al. we generated solution sets with the following shapes:

- **Linear**: For each solution, each coordinate is drawn from a uniform distribution [0,1] and afterwards normalized by the sum over the coordinates.
- **Spherical**: For each solution, each coordinate is drawn from a **Gaussian** distribution and afterwards normalized by the euclidean norm of the coordinates.
- **Random**: For each solution, each coordinate is drawn from a uniform distribution [0,1].

These differently shaped fronts simulate an early population in an optimization (random) and late populations close to two differently shaped optimal fronts (linear and spherical). While et. al. use these (and more shapes) to test the computation time of their algorithm [12], we additionally use them to test if and how the success of the approximating algorithm depends on the shape of the front. All experiments in this first part of the results were repeated between 5 and several hundred times (experiments with higher calculation time were repeated less often) and shown results are averages over all runs. All calculations were performed on a PC with Intel(R) Core(TM) i5-2400 CPU 3.1 GHz.

3.1.1 Comparison of algorithms in Shark library

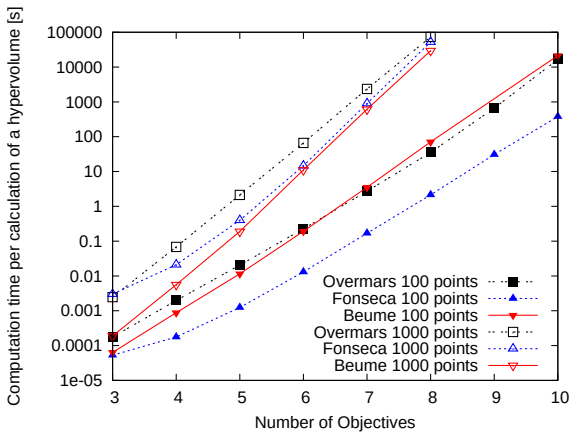


Figure 1: Average computation time for calculation of the hypervolume with the algorithms from [7, 4, 2] for solution sets of size 100 (solid symbols) and 1000 solutions (unfilled symbols) as a function of the number of objectives.

We compared the three algorithms by Overmars and Yap, Fonseca et al. and Beume et al. as they are implemented in Shark library. Fig. 1 shows the computation time for the algorithms applied to solution sets of sizes 100 and 1000 depending on the number of objectives. Although having different complexities in theory, all algorithms perform on a comparable level, with the exception of smaller solution sets where the algorithm by Fonseca et al. is approximately 10 times faster for five or more objectives.

3.1.2 Comparison with the algorithm by While

Since the performance of the algorithms from Shark library were comparable Fig. 2 shows the computation time of only one of them (Fonseca et al.) compared to the algorithm by While et al.. The latter clearly performs better except for small solution sets with 3 or 4 objectives. For the solution set of size 100 the algorithm by While et al. needs less than 10 seconds per calculation even for the case of 14 objectives. In contrast the algorithm by Fonseca et al. becomes slower by more than an order of magnitude for each additional objective above 4.

3.1.3 Analysis of approximative algorithm

In this section we investigate the performance of the approximative algorithm, compared to the fastest exact algorithm, the one by While et al.. First we analyze the computation time for different configurations, again averaged over 200 independent runs.

From Figures 3 and 4 we see that the approximative algorithm using a single line per solution is substantially quicker

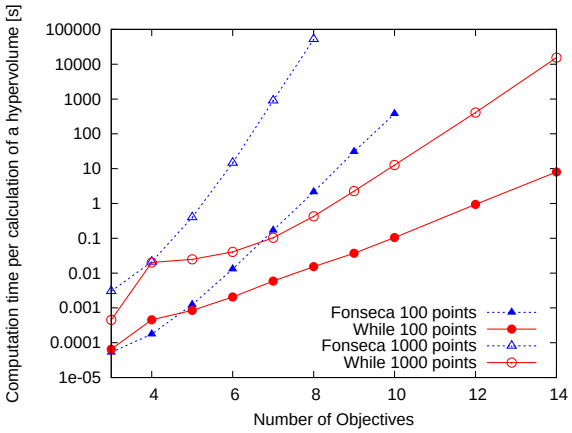


Figure 2: Average computation time for calculation of the hypervolume with the algorithms from [12, 2] for solution sets of 100 and 1000 solutions as a function of the number of objectives.

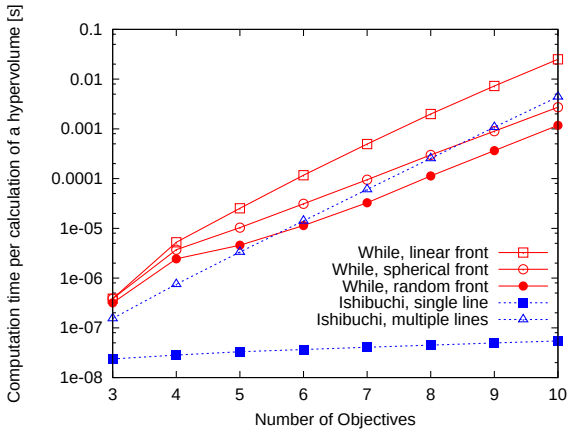


Figure 3: Average computation time per single hypervolume calculation using the exact algorithm by While et al. and the approximating algorithm by Ishibuchi et al. either using a single line or multiple lines for a set of 100 solutions on differently shaped fronts (linear, spherical, and random) as a function of the number of objectives. Only one data set is shown for algorithm by Ishibuchi et al. since its performance does not depend on the shape of the front. For more details on the number of lines, see main text.

than the fastest of the exact algorithms used in our tests. It also scales very favorably with the number of objectives and at least on a comparable level with the size of the solution set. Increasing the number of lines used for approximation also increases the computation time and affects the scaling behavior. Since the number of these lines is not controlled directly and depends on a free integer parameter as well as the number of objectives in a nontrivial way (in order to cope with the increasing dimensionality of the attainment surface), each data point of the respective curves in Fig. 3 and 5 uses a different number of lines. These are for 3 to 10 objectives as follows: 10, 35, 126, 462, 1716, 6435, 24310, 92378. This exponential increase in the number of lines explains the different scaling behavior compared to the far better scaling obtained for a single line.

We note that as expected the algorithm by While et al. shows a dependency of the computation time on the shape of the population front since it reduces its computation time by creating dominated solutions and discarding them. The computation time of the approximative method does not depend on the shape of the front which is the reason why only one of the data sets is shown.

We now take a look at a different aspect of the performance of the approximative method. In Fig. 5 and 6 we show the observed frequency to find one of the 10 % least contributing solutions in the set as approximated least contributor, once using a single line per solution and using multiple lines per solution. The decision for this criterion and in particular the value of 10%, is to some extent arbitrary. However, in our opinion finding the very least contributor is not strictly necessary for the SMS-EMOA as long as the discarded solutions are among the set of least contributors in their population. Corresponding tests can be found in the next part of this section.

For the pure benchmarks (Fig. 5 and 6) we see that for the randomly shaped front, performance is excellent, while for linear or spherical fronts a strong degradation with increasing number of objectives is observed which saturates to a level between 20% and 30% for 6 or more objectives. This does not seem to be particularly high but it is clearly higher than the value of 10% representing random selection of solutions.

From Fig. 6 we see that the success of the approximation does not systematically decrease with the size of the solution set. Comparing the approximation using multiple lines or a single line there is no systematic increase but this does not automatically imply a worse performance. This is due to the fact that the distribution of approximated least contributors also changes inside the set of the 10% least contributors. Nevertheless the improvement using multiple lines appears to be small compared to the associated increase in computation time. Because of their dependency on the shape of the front the two algorithms considered in this section need to be tested on more differently shaped fronts as done here. While et al. investigated the runtime of their algorithm using more front shapes in [12].

3.2 Performance in the SMS-EMOA

The question addressed in this section is whether the approximative hypervolume calculation affects the progress of the actual optimization. In order to assess the impact of the reduced precision we used the exact method by While et al. and the approximative method by Ishibuchi et al. within an SMS-EMOA applied to the DTLZ-1 problem [3] with three or six objectives. All experiments were averaged over several independent runs (200 for three objectives, 20 for six objectives). The parameters used in the optimization can be found in Table 1.

Since the Pareto optimal front is known analytically as the (hyper-)plane defined by the equation

$$\sum_{j=1,\ldots,d} f_j(\mathbf{x}) = 0.5$$

we can compute the average distance of the population to

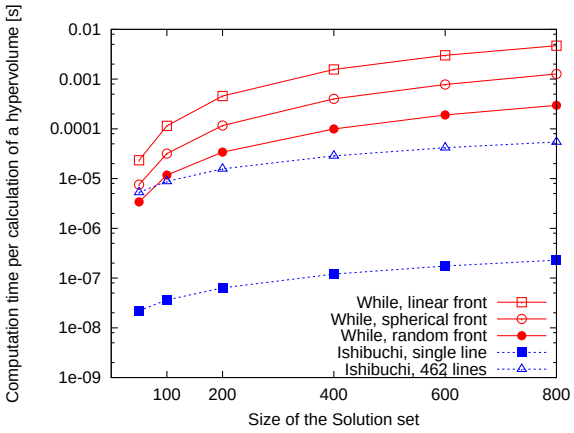


Figure 4: Average computation time per single hypervolume calculation using the same algorithms as in Fig. 3 for six objectives as a function of the size of the solution set.

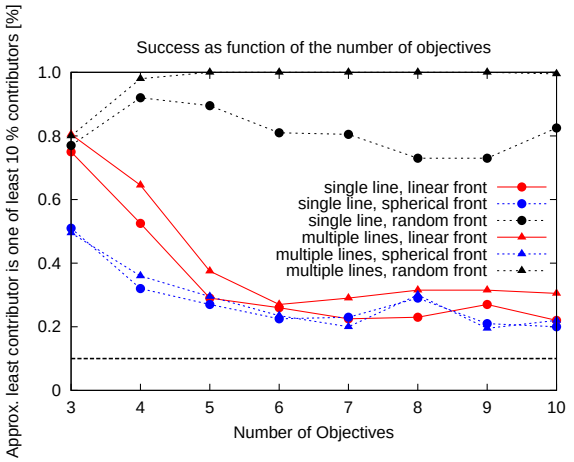


Figure 5: Success of the approximating algorithm by Ishibuchi et al. measured by the observed frequency of identifying a solution as least contributor which is really one of the 10% least contributors in a population of size 100 as function of the number of objectives.

the Pareto optimal front given by

$$D = \frac{1}{n} \sum_{i=1,\ldots,n} \left(\sum_{j=1,\ldots,d} \left| f_j\left(x^i\right) \right| - 0.5 \right)$$

at any point during the optimization. Furthermore we measure the exact hypervolume dominated by the solution set, which is independent from the hypervolume value used in the optimization. Since the algorithm by Ishibuchi et al. does not calculate the exact dominated hypervolume we use the algorithm by While et al. to compare the increase of the exact hypervolume during the optimization. Both of these convergence measures were already used in Ref. [5] and discussed in Ref. [11].

In order to see small differences between the optimization runs using different algorithms for hypervolume calculation we restrict the region of interest to the hypervolume below the fixed d-dimensional reference point $\mathbf{r} = (0.7, 0.7, \ldots)^T$.

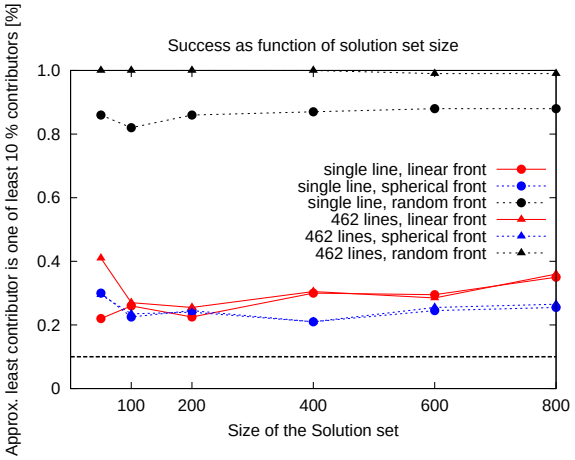


Figure 6: Success of the approximating algorithm by Ishibuchi et al. measured as the observed frequency of runs in which the algorithm selects a point as least contributor which really is one of the 10% least contributors as function of the size of the solution set with a fixed number of six objectives.

Parameter	Value
Population Size	100
Crossover Probability	1.0
Mutation Probability	$\frac{1}{7}$
η_m	20
η_c	15
Stopping Condition	30000 fitness evaluations

Table 1: Settings used in the SMS-EMOA

Any solution which does not dominate the reference point does not contribute to the hypervolume shown in Fig. 8. Since the solution set is unconstrained it is not guaranteed that there will even be one solution dominating this fixed reference point at all, not even if there were such solutions in an earlier population. However, since the hypervolume only increases during the optimization, this reduced hypervolume with respect to the reference point $\mathbf{r}$ is a lower bound on the actual hypervolume. It is important to mention that this hypervolume measure is not to be confused with the actual hypervolume used in the optimization. It is calculated independently and solely for the purpose of being able to compare the performance of the optimization with exact and approximative hypervolume calculation.

The distance D to the Pareto optimal front is shown in Fig. 7. As visible in the upper panel for 3 objectives the approximative method is not substantially worse than the exact method as both quickly approach zero with increasing number of fitness evaluations. The approximation using ten lines per solution seems to perform worse but if one takes into account Fig. 8 showing the fraction of the hypervolume missing to the maximally possible, there is only a small difference. This leads to the conclusion that the approximation using ten lines to some extent optimizes the hypervolume by increasing the distance between the solutions rather than by increasing the distance to the reference point. This is plausible since the additional lines cover the change of the at-

tainment surface around the solution whose shape depends on the position of neighboring solutions.

With regard to hypervolume there is no significant improvement from using a single line to using multiple lines which is in accord with the comparable performance seen earlier. The fact that the approximating algorithm achieves a larger hypervolume in the earlier phase of the optimization is likely to be caused either by insufficient statistics or the unequal distribution of solutions inside and outside the selected area of interest (as defined by the reference point $\mathbf{r}$). If this area is chosen large enough and one excludes effects caused by the randomness in the generation of new solutions by averaging over independent runs, the exact algorithm has to perform better since it always discards the least contributing solution.

However, for six objectives (lower panels of Fig. 7 and 8) we see a bigger difference between the optimizations using different algorithms for hypervolume calculation. The distance to the Pareto optimal front converges faster using the algorithm by While et al. Furthermore the hypervolume is still increasing after 30000 fitness evaluations using the exact algorithm in contrast to the optimizations using the algorithm by Ishibuchi et al. which do not show any increase of the hypervolume in the observed region of interest after 9300 fitness evaluations when using a single line per solution (5100 fitness evaluations when using 462 lines). The final values of the hypervolume are summarized in Table 2. As for the case of three objectives the improvement of using multiple lines compared to a single line is vanishingly small.

	DTLZ-1, 3 obj.	DTLZ-1, 6 obj.
Exact	0.98282%	0.99733%
Approx., single line	0.96284%	0.97306%
Approx., multiple lines	0.96410%	0.97398 %

Table 2: Fraction of the maximally achievable hypervolume after 30000 fitness evaluations in the SMS-EMOA using different algorithms for hypervolume calculation in the DTLZ-1 test problem with 3 or 6 objectives.

In order to not only test the performance on artificial test problems we used the real-world example with 11 decision- and 7 objective-variables described in Ref. [6]. The parameters of the optimization are the same as given in Table 1 except for the mutation probability which is $\frac{1}{11}$ and the stopping condition of now 10000 fitness evaluations for this example. Due to time restrictions these results only stem from a single run each. Since the Pareto optimal front is unknown, the only meaningful measure for comparison (shown in Fig. 9) is the fraction of hypervolume missing to the maximum hypervolume. As done for the DTLZ-1 problem we restricted the region of interest by using a fixed reference point but as there is no analytical value for the maximum hypervolume we use the maximum occurring in one of the three runs, which is the final value for the approximation using 84 lines per solution.

In contrast to the DTLZ-1 we see a significant difference of the increase in hypervolume especially between the data belonging to the approximations with single and multiple lines which were almost equal for the optimization of the DTLZ-1. This can be a hint, that the shape of the Pareto optimal front for the real-world example makes an approximation with multiple lines more beneficial than a linear shape does. Again, the worse performance of the exact algorithm is probably caused by the chosen reference point and the lack of averaging over more optimization runs.

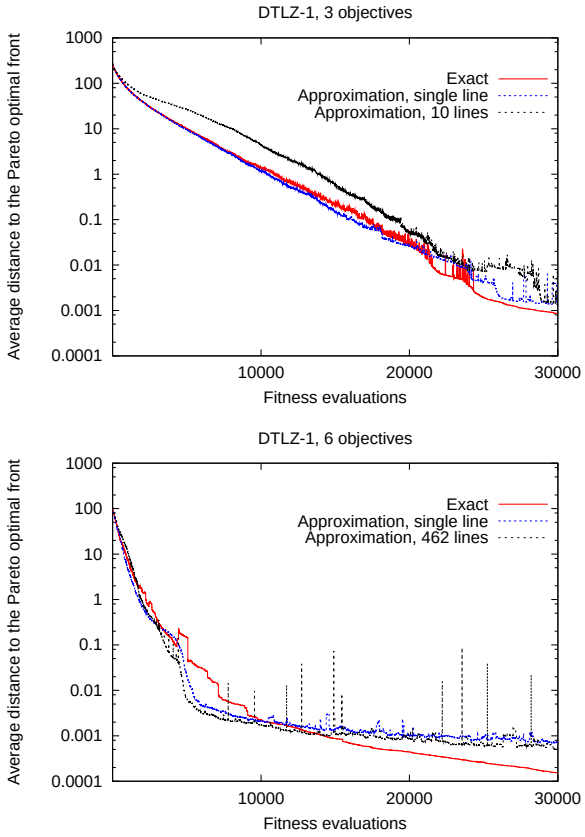
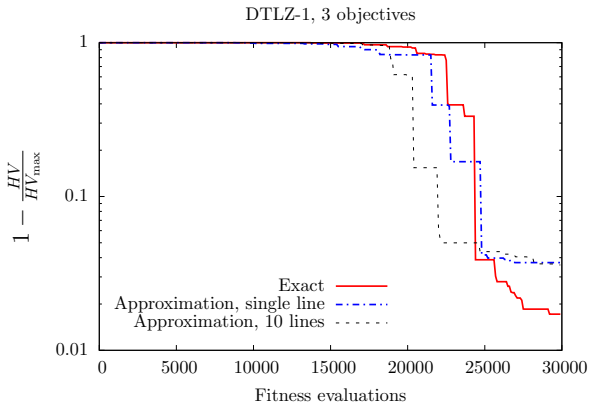


Figure 7: Average distance of the population to the Pareto optimal front (DTLZ-1 with three objectives (top) or six objectives (bottom)) during the optimization with SMS-EMOA using either the exact algorithm by While et al. or the approximative algorithm by Ishibuchi et al. (using a single line or multiple lines per point) and the parameters listed in Tab. 1.

4. CONCLUSION AND OUTLOOK

In this work we compared four exact and one approximative algorithms for fast hypervolume computation for three and more objectives. Criteria of the comparison were the dependency of the computation time on the number of objectives, the size of the solution set as well as its shape. For the approximative algorithm we tested the dependency of a successful approximation on the same parameters (number of objectives, size of the solution set, shape of the solution set). We found that the algorithms by Overmars and Yap [7], Fonseca et al.[4] and Beume et al.[2] perform on a comparable level but all of them become unfeasible beyond 6 or 7 objectives. A drastic speed-up can be gained using the algorithm by While et al.[12], especially for higher number of objectives. The approximative method by Ishibuchi et al.[5] allows a further speed-up, but compromises on quality



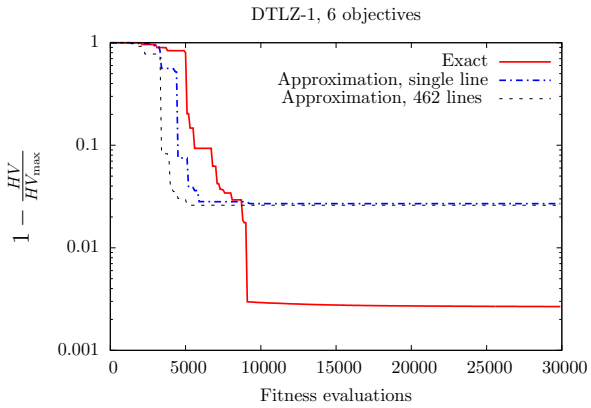


Figure 8: Average fraction of the hypervolume missing to the maximally achievable dominated hypervolume during the optimization of the DTLZ-1 test problem with three objectives (top) and six objectives (bottom) using the SMS-EMOA with the algorithms by While et al. or Ishibuchi et al. for the calculation of hypervolume contributions and the parameters listed in Tab. 1.

of the approximation. This method's runtime scales very favorably with number of objectives but we found that the success of finding one of the least contributing solutions is not very high for many objectives. Attempts to increase this rate of success results in computation times comparable to the exact algorithm. Finally we used the best-performing exact and the approximative algorithm in the SMS-EMOA in order to assess the impact of the approximation on the performance of the optimization. We found that the approximative algorithm performs comparable in the early phase of the optimization but worse when the solution set approaches the Pareto optimal front. Using multiple lines only causes a minor improvement for the DTLZ-1 problem but a significant improvement in the optimization of the real-world example. This could be examined by further tests of the approximating algorithm on more differently shaped fronts. Especially since the scaling of the computation time with the increasing number of objectives leads to the conclusion that for problems with a high number of objectives $d \gg 10$ the approximative method may be the only feasible option left.

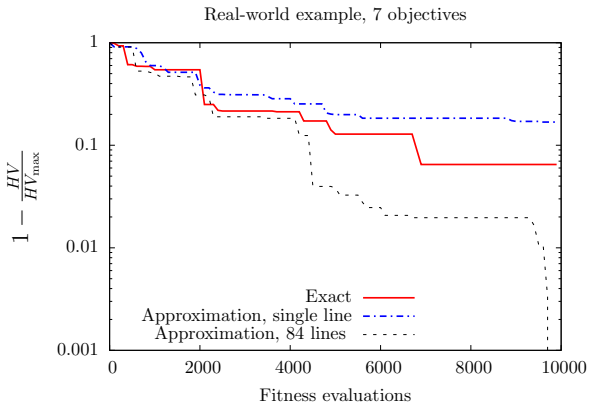


Figure 9: Fraction of the hypervolume missing to the maximum occurring value (final value of the approximation using 84 lines) during a single optimization of the real-world example with 11 decision variables and 7 objectives using SMS-EMOA with the algorithms by While et al. or Ishibuchi et al. for the calculation of hypervolume contributions.

Acknowledgments

We thank Mr. Sebastian Schmitt for his support with the implementation of the hypervolume computation algorithms.

5. REFERENCES

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

[2] Beume, N., and Rudolph, G. Faster s-metric calculation by considering dominated hypervolume as klee's measure problem. Tech. rep., Sonderforschungsbereich 531 Computational Intelligence, Universität Dortmund, 2006.

[3] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA)* (2002), pp. 825–830.

[4] Fonseca, C., Paquete, L., and Lopez-Ibanez, M. An improved dimension-sweep algorithm for the hypervolume indicator. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* (2006).

[5] Ishibuchi, H., Tsukamoto, N., Sakane, Y., and Nojima, Y. Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation* (2010), pp. 527–534.

[6] Narukawa, K., and Rodemann, T. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on* (2012), pp. 316–319.

[7] Overmars, M., and Yap, C.-K. New upper bounds in klee's measure problem. In *Foundations of*

Computer Science, 1988., 29th Annual Symposium on (1988), pp. 550–556.

[8] Paquete, L., Fonseca, C. M., and López-Ibáñez, M. An optimal algorithm for a special case of Klee's measure problem in three dimensions. Tech. rep., CSI, Universidade do Algarve, 2006.

[9] Sato, H., Aguirre, H. E., and Tanaka, K. Controlling dominance area of solutions and its impact on the performance of moeas. In *Evolutionary Multi-Criterion Optimization* (2007), pp. 5–20.

[10] Shah, R. A., Reed, P. M., and Simpson, T. W. Many-objective evolutionary optimisation and visual analytics for product family design. In *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer, 2011, pp. 137–159.

[11] Wagner, T., Beume, N., and Naujoks, B. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Evolutionary Multi-Criterion Optimization* (2007), pp. 742–756.

[12] While, L., Bradstreet, L., and Barone, L. A fast way of calculating exact hypervolumes. *Evolutionary Computation, IEEE Transactions on 16* (2012), 86–95.

[13] Zhang, Q., and Li, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on 11*, 6 (2007), 712–731.