A Fast Approximate Hypervolume Calculation Method by a Novel Decomposition Strategy

Weisen Tang, Hailin Liu(✉), and Lei Chen

Guangdong University of Technology, Guangzhou, China
hlliu@gdut.edu.cn

Abstract. In this paper, we present a new method to fast approximate the hypervolume measurement by improving the classical Monte Carlo sampling method. Hypervolume value can be used as a quality indicator or selection indicator for multiobjective evolutionary algorithms (MOEAs), and thus the efficiency of calculating this measurement is of crucial importance especially in the case of large sets or many dimensional objective spaces. To fast calculate hypervolume, we develop a new Monte Carlo sampling method by decreasing the amount of Monte Carlo sample points using a novel decomposition strategy in this paper. We first analyze the complexity of the proposed algorithm in theory, and then execute a series experiments to further test its efficiency. Both simulation experiments and theoretical analysis verify the effectiveness and efficiency of the proposed method.

Keywords: Hypervolume · MOEAs · Multiobjective optimization · Performance metrics

1 Introduction

In this paper we consider the problem of approximate hypervolume of the space dominated by a set of *d*–dimensional points. This hypervolume is often used as quality indicator in multi–objective evolutionary algorithms (MOEAs). Multi–objective evolutionary algorithms such as NSGA–II [1], MOEA/D [2, 3] have been successfully applied in various bi–objective and tri–objective optimization scenarios. However, they all appear to encounter difficulties when the number of objectives increases to more than three, which is known as many–objective optimization problems (MaOPs) [4].

As a consequence, researchers have tried to develop some alternative methods, and one of them is to use set quality measures [5], or quality indicators as the measurement to select the next generation population. Hypervolume [6] (or S–metric, Lebesgue) measure is one of the most popular quality indicator that can be fully sensitive to Pareto dominance and population diversity even when more than three objectives are involved. It was originally proposed and employed by Zitzler et al. [6] to quantitatively compare the outcomes of different EMO algorithms. In [6], the indicator was denoted as 'size of the space covered', and later also termed as 'S–metric', 'hypervolume indicator' [7], and hypervolume measure [8].

D.-S. Huang et al. (Eds.): ICIC 2017, Part I, LNCS 10361, pp. 14–25, 2017.
DOI: 10.1007/978-3-319-63309-1_2

It has been identified that the computational effort of computing hypervolume metric increases exponentially with the increasing number of objectives. Almost all the recent studies about hypervolume indicator are about how to fast calculate the hypervolume.

The "HV4D" algorithm [10, 11] is the most efficient in the case $d = 4$, see the computational results e.g. [12]. Above $d = 4$, the walking fish group (WFG) [13] and quick hypervolume (QHV) [12] are currently two of the most efficient with the computational experiments. Although both the complexity of them are exponential, they have acceptable runtime within $d = 14$.

Recently, some improved version about these two algorithms were proposed. Quick hypervolume was extended in computing the contributor of each point and speed it up with parallel computation [15]. Jaszkiewicz used a similar scheme as in the original quick hypervolume algorithm called QHV–II [14]. IWFG algorithm was proposed in [23] which used the adaptive slicing scheme, it processes less than a second that sets containing a thousand points in 10–13 objectives. On the other hand, Lacour [16] proposed another approach called 'HBDA' to calculate the computation of the hypervolume indicator. There are a couple of other methods that partly perfect the analysis and provement of computing runtime, e.g. [9, 17, 24].

In this paper, we mainly discuss not the exact hypervolume but the approximation. A first attempt in this direction presented in [18, 22]. The main idea is to estimate–by means of Monte Carlo simulation–the ranking of the individuals that is induced by the hypervolume indicator and not to determine the exact indicator values in the optimization problems. In [25] there are some performance analysis about this approximation method and hence proved the effectiveness of Monte Carlo simulation. This method is widely used such as HypE [19], SMS–EMOA [8], but few researchers give it a performance boost (e.g. [21] is one of them).

We design an new approximate hypervolume calculation method which greatly improve the approximate speed. The main idea is to decrease the number of Monte Carlo sample points under the precondition of retaining the precision by a novel decomposition strategy. This novel decomposition strategy is also an exact hypervolume calculation method and is similar to QHV but different. The accordingly complexity analyses will be given and this method will compare with the previous Monte Carlo simulation method [22] to show the faster calculating speed.

This paper is organized as follows. Section 2 briefly introduces the many–objective optimization problems and some basic concept of hypervolume. Section 3 gives a detail description of the proposed hypervolume calculation method. Some theoretical analysis proposes in Sect. 4. At last we conduct simulation experiments in 8, 10, 13 dimensional space respectively, and compare with the Monte Carlo simulation method in Sect. 5 to verify the effectiveness and efficiency of proposed method. Section 6 concludes this paper.

2 Problem Definition

Without loss of generality, the multi–objective optimization problem can be stated as follows

$$\begin{aligned} &\min F(x) = (f_1(x), f_2(x), \ldots, f_d(x))^T \\ &s.t. x \in \Omega. \end{aligned} \tag{1}$$

Where $\Omega \subset R^m$ is the decision space and m is the dimensionality of the decision variable x. $F : \Omega \to R^d$ consists d real–value objective functions and R^d is called the objective space. When $d \geq 4$, the problem (1) is regarded as a many–objective optimization problem.

Let $u = (u_1, \ldots, u_d)^T$ and $v = (v_1, \ldots, v_d) \in R^d$ are images of two solutions in the objective space, u is said to dominate v ($u \prec v$) if and only if $u_i \leq v_i$ for all $i = 1, \ldots, d$ and $u \neq v$. x^* is called Pareto optimal solution if there is no solution $x \in \Omega$ such that $F(x) \prec F(x^*)$. The set of all Pareto optimal solutions in Ω is denoted as $E(f, D)$ and we called that Pareto Front (PF) in the objective space.

According to the definition above, in the d–dimensional objective space, we define the dominated region namely hypervolume measure for given any set $\mathcal{F}$ of n points as follows

$$HV(\mathcal{F}) = VOL\left(\bigcup_{(x_1, x_2, \cdots, x_d) \in \mathcal{F}} [x_1, b_1] \times \ldots \times [x_d, b_d] \right) \tag{2}$$

where $(b_1, b_2, \cdots, b_d)$ is a bound of $\mathcal{F}$ which is dominated by all the solutions in $\mathcal{F}$ and $VOL(x) = \prod_{i=1}^{d} (x_i)$. Similarly, we define $VOL(x|b) = \prod_{i=1}^{d} (b_i - x_i)$ for convenience. Actually, hypervolume measure is the volume of the union of the boxes which is made up by the region between a point and the coordinate axis. This metric value is to be maximized, the larger measure of a population, the better it is.

3 The Proposed Fast Approximate Hypervolume Algorithm

In this section, we describe how our algorithm to fast calculate the approximate hypervolume indicator in high dimensions objective space. The main idea of our algorithm is that we use a novel decomposition strategy called "partial precision and partial approximation (PPPA)" to decrease the number of points that need to be sampled for hypervolume calculation.

3.1 A Calculating Exact Hypervolume Algorithm

First of all, we will introduce a pivot divide and conquer method which provides accurate calculation of hypervolume measure, and it works as follows:

1. Select a pivot point which matches certain definition or condition, calculate the volume of pivot point and add up the hypervolume measure;
2. To produce the sub–problems, we divide the space according to the pivot, classify other points into the possible space regions;
3. Recursively solve each of the sub–problems.

The methodology of this method is similar to QHV and QHV–II, but in a different way. The differences are mainly reflected in the splitting way and the pivot selecting way. Figure 1 illustrates the proposed method in an 3–dimension example for maximize objectives. In Fig. 1, each dimension d_i of pivot point would be segmented. The points which greater than the *Pivot* in one dimension will divide into two parts, one part get added to subset $\mathcal{Q}_i$, while the other part join the segmentation of next segmentation. The method is presented in Algorithm 1, where `F[i]` represents the individual i in population $\mathcal{F}$ and we denote as $\mathcal{F}^i$.

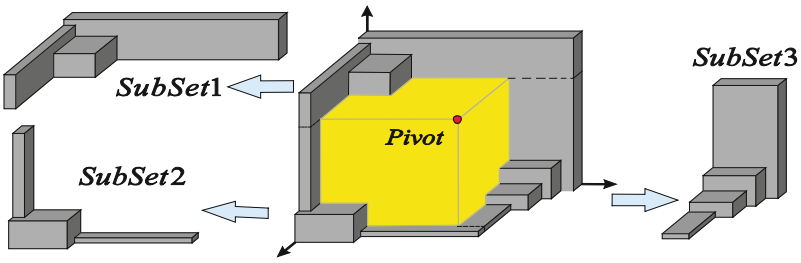

Fig. 1. Illustration of dividing a set in 3D case with Maximizing in three objectives relative to the origin.

The function `FindPivot(F,B)` in line 5 is to find the pivot. In Algorithm 1, the pivot is produced by the maximum–minimum method because this point is the midpoint of the set. That means $\mathcal{F}^p$ satisfies

$$\begin{aligned} g^{te}(\mathcal{F}^i) &= max\{\mathcal{F}^i_1, \mathcal{F}^i_2, \cdots, \mathcal{F}^i_d\} \\ g^{te}(\mathcal{F}^p) &= min\{g^{te}(\mathcal{F}^1), \cdots, g^{te}(\mathcal{F}^n)\}. \end{aligned} \tag{3}$$

In Algorithm 2, maximum volume point is chosen as pivot as it would reduce volume which needs to be approximate as larger as possible. In other words, $\mathcal{F}^p$ satisfies following equation

$$VOL(\mathcal{F}^p|b) = max\{VOL(\mathcal{F}^1|b), \cdots, VOL(\mathcal{F}^n|b)\}. \tag{4}$$

Algorithm 1. Pivot Divide to Calculate Hypervolume in d Dimension

```
PDC_HV(F,B)
  Input
    F: A set of n points;
    B: the point of computational boundary in Eq.(2);
  Output
    HV: the value of hypervolume measure for F;
  Begin
    If(n == 1)
      HV := VOL(F[1],B);      //Direct calculation of volume
    Else
      p := FindPivot(F,B);    //Find the pivot
      HV := VOL(F[p],B);
      For(i = 1 to d)
        Q := [];              //Initialize the subset of F
        For(j = 1 to n)
          If(F[j][i] < F[p][i])
            F[j][i] := B[i] – (F[p][i]- F[j][i]);
            Q := [Q, F[j]]; F[j][i] := F[p][i];
          End If
        End For
        HV := HV + PDC_HV(Q,B)
      End For
    End If
    Return HV;
end
```

3.2 The Proposed Approximate Hypervolume Method in High Dimension

We introduce a new approximate hypervolume method according to above description. Notice that the Algorithm 1 is also a good strategy to approximate hypervolume measure. After k layers recursion, d^k subsets can be obtained. Besides every recursion, this method would calculate a new volume of pivot and add to total volume of hypervolume. Therefore, we have the following formula

$$HV(\mathcal{F}) = \sum VOL(Pivot|b) + \sum_{i=1}^{d^k} HV(\mathcal{Q}_i) = V_p + V_{\mathcal{Q}} \tag{5}$$

Where $V_p = \sum VOL(Pivot|b)$ is the sum of the volume of all pivots, $\mathcal{Q}_i$ traverses all subsets which is obtained by the k–layer segmentation and $V_{\mathcal{Q}}$ is the sum of the hypervolume measure of all subsets. That is why we call this novel decomposition

strategy "partial precision and partial approximation". Because after k layers recursion, Eq. (5) could be regarded as an equation with error term that approximate hypervolume measure. The former is the precise calculation of partial hypervolume and the latter is error term. Obviously, the former is easy to calculate for it is the multiplication of a series of number continuously, and the latter is difficult to calculate. Thus, a new method is proposed to approximate to the error term of Eq. (5).

In Eq. (5), k represents the layer of recursion that needs be presented. The larger the k value is, the more the layers of recursion is and the more accurate results are. But it also the more complex of computation because the more subsets will be generated. According to Eq. (5), the volume of this series of subsets have been reduced and then we can approximately compute them by the method as follows.

Algorithm 2. Partial Precision and Partial Approximation of Hypervolume

```
PPPA_HV(F,B,p,Iter)
  Input
    F and B is the Same as Algorithm 1;
    p: the number of sample points in unit volume;
    Iter: the layer of recursion;
  Output
    Same as Algorithm 1;
  Begin
    If(Iter <= k)
      Execute Algorithm 1 line 4 to 13;
      HV := HV + PPPA_HV(F,B,p,Iter+1);
    Else
      S := [];                    //Initialize the set of Sample
      For(i = 1 to n)
        Delete points in S which is dominated by F[i];
        s := p * VOL(F[i],B);  //Sample size of F[i]
        Produce S' which Contains s points within F[i];
        S := [S, S'];
      End For
    End If
    Return |S|/p;
end
```

We now consider the Monte Carlo simulation method which is mentioned in [18, 19]. If the set $\mathcal{F}$ accounts for a small proportion in sample range $[0, b_1] \times \ldots \times [0, b_d]$, most of the sample points will tend to be beyond the set $\mathcal{F}$, and it will waste computation effort. The subsets $\mathcal{Q}_i$ obtained by Eq. (5) also have this situation. Therefore, Monte Carlo simulation has disadvantage in computing the above mentioned subsets. To overcome this, we develop a new method to compute the hypervolume measure of this class of subsets like $\mathcal{Q}_i$. The method associates the numbers of sample points with the size of hypervolume, and hence can reduce the numbers of sample points. It is

described in Algorithm 2, where k is the default of max recursive layer, which often greater than 2, the parameter *Iter* is set as 1 and $|\mathcal{S}|$ represents the number of $\mathcal{S}$.

It is obvious that the complexity of the Algorithm 2 is highly related to the size of volume. The complexity of Algorithm 2 will be discussed in the next section. The essential principle of this method is the relationship of density and volume. As is well known, the mass equals the density times the volume. With density unchanged, we add points to the dominated region which is consisted by the set $\mathcal{Q}_i$ and the boundary b, until the dominated region is saturated. Then the hypervolume measure equals the numbers of total points in the dominated region divided by the density. It can be concluded that when the hypervolume measure of a set $\mathcal{F}$ satisfies $VOL(\mathcal{F}^i|b) < 1$, the algorithm would need less number of sample points, and computing efficiency can be increased.

4 Theoretical Analysis of the Proposed Algorithms

We prove that the complexity of Algorithm 2 is $O(dn\rho(V_{\mathcal{Q}})^{1+\varepsilon} + (dn)^k)$, where the definition of $V_{\mathcal{Q}}$ is similar to the last term (error term) of Eq. (5), ε is an arbitrarily small number and ρ is the sample points per unit volume. The greater the ρ, the more accurate the measured value will be, and vice versa.

Proposition 1. As described, the computational complexity of Algorithm 2 is $O(dn\rho(V_{\mathcal{Q}})^{1+\varepsilon} + (dn)^k)$.

Proof(of proposition). In general, let the sample boundary be $(b_1, \cdots, b_d) = (1, \cdots, 1)$ because arbitrary point set can be mapped in $[0,1]^d$ by scaling. Let $\mathcal{F} = \{x^1, x^2, \cdots, x^n\}$, $V_i = VOL(x^i|b)$, $V_i < 1$, $i = 1, 2, \cdots, n$. Notice that in Algorithm 2 if we guarantee

$$\frac{S_1}{V_1} = \frac{S_2}{V_2} = \cdots = \frac{S_n}{V_n} = \rho$$

And then we uniformly generate $S_i (i = 1, 2, \cdots, n)$ sample points in the sample range $[x^i, b]^d$ and add into the set $\mathcal{S}$. So we just need to compare the dominated relationship between x^i and the sample points which in the $\mathcal{S}$, if the set $\mathcal{S}$ contains $|\mathcal{S}|$ sample points, the computational complexity is $O(|\mathcal{S}|d)$. Besides, because $|\mathcal{S}|/\rho$ converges to the measure value hv of hypervolume which means $||\mathcal{S}|/\rho - hv| < \varepsilon$. Therefore $|\mathcal{S}| < \rho(hv)^{1+\varepsilon}$ can be deduced. Such an operation have to be repeated n times, so the complexity of approximate calculation in Algorithm 2 is $O(dn\rho(hv)^{1+\varepsilon})$.

According to Algorithm 1, the cost of one time segmentation is $O(dn)$. A segmentation can generated d subsets. After k layers segmentation, d^k subsets are generated and the cost of segmentation is equal to $1 + d + \cdots + d^{k-1} = O((dn)^k)$.

We suppose that after segmentation the hypervolume measure of subset Q_i is hv_i, then the cost of calculate a subset Q_i is $O(dn\rho(hv_i)^{1+\varepsilon})$. Meanwhile the cost of calculate all subsets $Q_1, Q_2, \cdots, Q_{d^k}$ is

$$\sum_{i=1}^{d^k} O(dn\rho(hv_i)^{1+\varepsilon}) < O\left(dn\rho\left(\sum_{i=1}^{d^k}(hv_i)^{1+\varepsilon}\right)\right) < O(dn\rho(V_{\mathcal{Q}})^{1+\varepsilon}) \tag{6}$$

So the complexity of Algorithm 2 is $O(dn\rho(V_{\mathcal{Q}})^{1+\varepsilon} + (dn)^k)$.

5 Experimental Studies

In order to demonstrate the feasibility and effectiveness of the proposed method, we do a series of comparative experiments. To be specific, the proposed Algorithm 2 is compared to Monte Carlo simulation method in terms of speed and accuracy. We calculate each instances set 20 times from 8 to 13 dimension, instances include the following four instance types:

(C) Concave or so-called spherical instances;
(X) Convex instances;
(L) Linear instances;
(D) Degeneration instances;

Instances are obtained by drawing uniformly points from the open hypercube $(0, 1)^d$. Then each point z is modified as follows:

(C) $z_j \leftarrow \frac{z_j}{\sqrt{\sum_{k=1}^{d} z_k^2}}$, for each $j \in \{1, \ldots, d\}$, i.e. the component values of z are divided by their ℓ_2–norm.

(X) $z_j \leftarrow 1 - \frac{z_j}{\sqrt{\sum_{k=1}^{d} z_k^2}}$, for each $j \in \{1, \ldots, d\}$.

(L) $z_j \leftarrow \frac{z_j}{\sum_{k=1}^{d} z_k}$, for each $j \in \{1, \ldots, d\}$, i.e. the component values of z are divided by their ℓ_1–norm.

(D) $z_j \leftarrow \frac{1}{l}\sum_{k=1}^{l} z_k$, for each $j \in \{1, \ldots, l\}$, then $z_j \leftarrow \frac{z_j}{\sqrt{\sum_{k=1}^{d} z_k^2}}$, for each $j \in \{1, \ldots, d\}$, l is the degree of degradation, in experiment we let $l = \lceil \frac{d}{2} \rceil$.

We followed the suggestion of Russo and Francisco [12] to project uniformly distributed points on a hypersphere. In our study, 10, 000, 000 sample points per unit volume are used to ensure the accuracy, and the max segmentation recursive layer k is set 2. We use the following formula to calculate the accuracy

$$Accuracy = \frac{|hv^* - HV(\mathcal{F})|}{HV(\mathcal{F})}$$

Where $HV(\mathcal{F})$ is the exact hypervolume measure of the set $\mathcal{F}$ and hv^* is the approximation by proposed Algorithm 2 or Monte Carlo simulation. The results are shown in Fig. 2 and Table 1, Fig. 2 shows the running time comparison, and Table 1 shows the accuracy comparison for four kinds of different non-dominated sets.

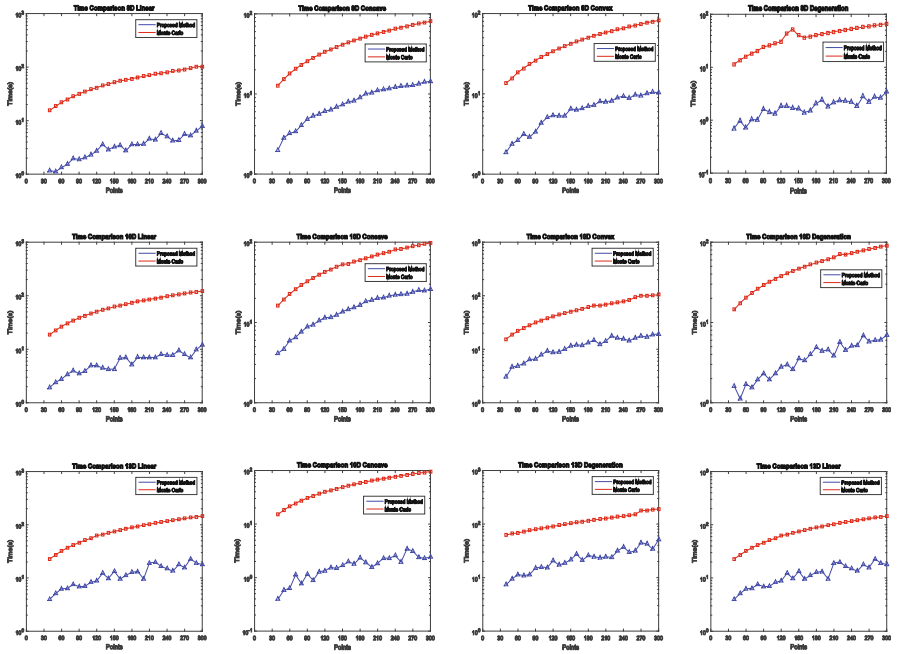

Fig. 2. The running time comparison for 8D, 10D, 13D with different instances set (seconds).

From those figures and table, we can conclude that our approximation algorithm outperforms Monte Carlo simulation in both speed and accuracy. The runtime of our algorithm is 10 times or more faster than Monte Carlo simulation, we only takes less than 10 s to perform a calculation and Monte Carlo simulation needs about 100 s or more. Besides, the accuracy of our algorithm is also far better than Monte Carlo. When the dimension is higher than 13, Monte Carlo simulation couldn't meet the requirement of accuracy, but our proposed algorithm is constant no matter what set it is.

We also have several disadvantages, As can be seen from the figure, our algorithm is not quite stable and the runtime would be fluctuate. Despite the fluctuation is in a certain range, it can't be predict perfectly. The reason is that our algorithm is influenced by the distribution and the number of the set. Obviously, this problem won't happen to Monte Carlo simulation.

Table 1. The accuracy comparison in different instances set(%).

Instances		Linear		Convex	
Algorithms		Proposed algorithm	Monte Carlo	Proposed algorithm	Monte Carlo
8D	100	5.617E-05	7.901E-05	1.034E-04	2.078E-04
	200	1.041E-04	7.350E-05	1.172E-04	2.336E-04
	300	4.763E-06	8.488E-05	3.945E-04	9.869E-05
10D	100	1.150E-04	1.051E-04	1.847E-04	1.013E-04
	200	1.074E-04	8.830E-05	7.575E-05	3.284E-04
	300	1.030E-04	1.084E-04	2.444E-04	2.764E-04
13D	100	1.653E-04	1.458E-04	1.412E-04	2.653E-01
	200	5.918E-06	1.364E-04	1.620E-04	5.381E-01
	300	4.447E-05	1.546E-04	1.029E-04	2.832E-01
Instances		Concave		Degeneration	
Algorithms		Proposed algorithm	Monte Carlo	Proposed algorithm	Monte Carlo
8D	100	1.341E-04	6.840E-04	4.200E-05	7.664E-03
	200	1.028E-04	7.542E-04	4.535E-05	3.335E-02
	300	1.601E-04	5.772E-04	1.618E-04	2.803E-02
10D	100	6.850E-05	1.537E-03	5.371E-05	1.740E-01
	200	2.349E-04	2.207E-03	6.587E-05	1.539E-01
	300	2.089E-04	1.490E-03	2.889E-04	3.316E-01
13D	100	5.474E-04	1.531E-01	8.870E-04	4.279E-02
	200	8.154E-05	2.601E-03	3.587E-05	1.539E-01
	300	4.089E-04	1.287E-01	5.933E-04	3.240E-01

6 Conclusion

In this paper, we propose a new method to approximate the hypervolume value, which can effectively decrease the running time of hypervolume indicator calculation in high dimension objective space. The performance of the proposed approximation hypervolume calculation method is verified by comparing it with the classical Monte Carlo sampling method on PFs of widely-used MaOPs test problems.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Grant 61673121, in part by the Projects of Science and Technology of Guangzhou under Grant 201508010008, and in part by the China Scholarship Council.

References

1. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms, vol. 2(3), pp. 509. John Wiley & Sons, Inc., New York (2001)
2. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)

3. Liu, H., Gu, F., Zhang, Q.: Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. IEEE Trans. Evol. Comput. **18**(3), 450–455 (2014)
4. Schutze, O., Lara, A., Coello, C.A.C.: On the influence of the number of objectives on the hardness of a multiobjective optimization problem. IEEE Trans. Evol. Comput. **15**(4), 444–455 (2011)
5. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30217-9_84
6. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms — a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998). doi:10.1007/BFb0056872
7. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)
8. Beume, N., Naujoks, B.: SMS–MOA: Multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
9. Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 436–447. Springer, Heidelberg (2008). doi:10.1007/978-3-540-92182-0_40
10. Cox, W., While, L.: Improving and extending the HV4D algorithm for calculating hypervolume exactly. In: Kang, B.H., Bai, Q. (eds.) AI 2016. LNCS, vol. 9992, pp. 243–254. Springer, Cham (2016). doi:10.1007/978-3-319-50127-7_20
11. Guerreiro, A.P., Fonseca, C.M., Emmerich, M.T.: A fast dimension–sweep algorithm for the hypervolume indicator in four dimensions. In: Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012), pp. 77–82 (2012)
12. Russo, Francisco, A.P.: Quick hypervolume. IEEE Trans. Evol. Comput. **18**(4), 481–502 (2014)
13. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. IEEE Trans. Evol. Comput. **16**(1), 86–95 (2012)
14. Jaszkiewicz: Improved quick hypervolume algorithm. In: Asilomar Conference on Signals, Systems and Computers (2017)
15. Russo, L.M.S., Francisco, A.P.: Extending quick hypervolume. J. Heuristics **22**(3), 245–271 (2016)
16. Lacour, R., Klamroth, K., Fonseca, C.M.: A box decomposition algorithm to compute the hypervolume indicator. Comput. Oper. Res. (2015)
17. Beume, N., Carlos, M.F.: On the complexity of computing the hypervolume indicator. IEEE Trans. Evol. Comput. **13**(5), 1075–1082 (2009)
18. Bader, J., Deb, K., Zitzler, E.: Faster hypervolume–based search using monte carlo sampling. In: Ehrgott, M., Naujoks, B., Stewart, T., Wallenius, J. (eds.) Conference on Multiple Criteria Decision Making (MCDM 2008). LNEMS, vol. 634, pp. 313–326. Springer, Heidelberg (2008)
19. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume–based many–objective optimization. Evol. Comput. **19**(1), 45–76 (2011)
20. Naujoks, B.: S–metric calculation by considering dominated hypervolume as Klee's measure problem. Evol. Comput. **17**(4), 477–492 (2009)
21. Bringmann, K., Friedrich, T.: Approximation quality of the hypervolume indicator. Artif. Intell. **195**(1), 265–290 (2013)

22. Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 436–447. Springer, Heidelberg (2008). doi:10.1007/978-3-540-92182-0_40
23. Cox, W., While, L.: Improving the IWFG algorithm for calculating incremental hypervolume. In: IEEE Congress on Evolutionary Computation IEEE, pp. 3969–3976 (2016)
24. Bringmann, K., Friedrich, T.: Parameterized average–case complexity of the hypervolume indicator. In: Conference on Genetic and Evolutionary Computation, pp. 575–582 (2013)
25. Nowak, K., Martens, M., Izzo, D.: Empirical performance of the approximation of the least hypervolume contributor. In: International Conference on Parallel Problem Solving From Nature, pp. 662–671 (2014)

http://www.springer.com/978-3-319-63308-4

Intelligent Computing Theories and Application
13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part I
Huang, D.-S.; Bevilacqua, V.; Premaratne, P.; Gupta, P. (Eds.)
2017, XXI, 815 p. 330 illus., Softcover
ISBN: 978-3-319-63308-4