3D Fast Convex-Hull-Based Evolutionary Multiobjective Optimization Algorithm

Jiaqi Zhao[a,b,*], Licheng Jiao[b], Fang Liu[b], Vitor Basto Fernandes[c,d], Iryna Yevseyeva[e], Shixiong Xia[a], Michael T. M. Emmerich[f]

[a] *School of Computer Science and Technology, China University of Mining and Technology, No 1, Daxue Road, Xuzhou, Jiangsu, 221116, China*
[b] *Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education, International Research Center for Intelligent Perception and Computation, Joint International Research Laboratory of Intelligent Perception and Computation, Xidian University, Xi'an Shaanxi Province 710071, China*
[c] *Instituto Universitário de Lisboa (ISCTE-IUL), University Institute of Lisbon, ISTAR-IUL, Av. das Forças Armadas, 1649-026 Lisboa, Portugal*
[d] *School of Technology and Management, Computer Science and Communications Research Centre, Polytechnic Institute of Leiria, 2411-901 Leiria, Portugal*
[e] *Faculty of Technology, De Montfort University, Gateway House 5.33, The Gateway, LE1 9BH Leicester, UK*
[f] *Multicriteria Optimization, Design, and Analytics Group, LIACS, Leiden University, Niels Bohrweg 1, 2333-CA Leiden, The Netherlands*

Abstract

The receiver operating characteristic (ROC) and detection error tradeoff (DET) curves have been widely used in the machine learning community to analyze the performance of classifiers. The area (or volume) under the convex hull has been used as a scalar indicator for the performance of a set of classifiers in ROC and DET space. Recently, 3D convex-hull-based evolutionary multiobjective optimization algorithm (3DCH-EMOA) has been proposed to maximize the volume of convex hull for binary classification combined with parsimony and three-way classification problems. However, 3DCH-EMOA revealed high consumption of computational resources due to redundant convex hull calculations and a frequent execution of nondominated sorting. In this paper, we introduce incremental convex hull calculation and a fast replacement for non-dominated sorting. While achieving the same high quality results, the computational effort of 3DCH-EMOA can be reduced by orders of magnitude. The average time complexity of 3DCH-EMOA in each generation is reduced from $O(n^2 \log n)$ to $O(n \log n)$ per iteration, where n is the population size. Six test function problems are used to test the performance of the newly proposed method, and the algorithms are compared to several state-of-the-art algorithms, including NSGA-III and MOPSO

variants, which were not compared to 3DCH-EMOA before. Experimental results show that the new version of the algorithm (3DFCH-EMOA) can speed up 3DCH-EMOA for about 30 times for a typical population size of 100 without reducing the performance of the method.
Keywords: Convex hull, area under ROC, classification, indicator-based evolutionary algorithm, multiobjective optimization, ROC analysis.

1. Introduction

Receiver operator characteristic (ROC) [1] and detection error tradeoff (DET) [2] curves are commonly used to evaluate the performance of binary classifiers in machine learning [3, 4]. ROC describes the relationship between true positive rate (*TPR*) and false negative rate (*FNR*). High value of *TPR* and small value of *FNR* are preferable, however, the performance of *TNR* and *FNR* are in conflict with each other. DET curves show tradeoff between false positive rate (*FPR*) and false negative rate (*FNR*). ROC convex hull (ROCCH) analysis, which covers potential optimal points for a given set of classifiers, has drawn much attention in [5, 6]. More recently, multiobjective optimization techniques became useful for maximizing ROCCH [7, 8, 9, 10, 11]. The aim of ROCCH maximization is to find a set of classifiers that perform well in the ROC space. The ROCCH maximization is a special case of a multiobjective optimization problem [7], as the maximization of *TPR* and minimization of *FNR* can be viewed as two conflicting objectives, and the parameters of a classifier can be viewed as decision variables.

Evolutionary multiobjective algorithms (EMOAs) [12, 13] are known to be good tools to deal with the tuning of machine learning problems [14, 15], image processing pipelines [16, 17], text message classifiers [18, 19]. Several EMOAs have been combined with genetic programming to maximize ROCCH in [7], including Nondominated Sorting Genetic Algorithm II (NSGA-II) [20], Multiobjective Evolutionary Algorithms Based on Decomposition (MOEA/D) [21, 22], Multiobjective selection based on dominated hypervolume (SMS-EMOA) [23, 24], and Approximation-Guided Evolutionary Multi-Objective (AG-EMOA) [25]. However, these methods do not consider

*Corresponding author. Tel.: +86 17368987876.
Email address: `jiaqizhao88@126.com` (Jiaqi Zhao)

special characteristics of ROC: The objective space is bounded by $(0, 0)$ and $(1, 1)$ and that concavities on the Pareto front can be healed by convexly combining classifiers [4]. Convex-hull-based multiobjective genetic programming (CH-MOGP) is proposed in [8] to maximize ROCCH for binary classifiers, which takes the convex hull of ROC into consideration. CH-MOGP is a tailor-made indicator-based evolutionary multiobjective algorithm (IBEA) [26] for computing a representation of a Pareto front of binary classifiers, using the area under the convex hull (*AUC*) as a performance indicator to guide the the evolution of a population.

CH-MOGP can only deal with binary classifiers, and is not able to address additional objectives, such as parsimony [27]. Moreover, it can not deal with multi-class and three-way classification [28]. 3D convex-hull-based EMOA (3DCH-EMOA) is proposed in [9]. It extends the ROCCH to triobjective problems by considering the classifier complexity ratio (*CCR*) of binary classifiers as the third objective in augmented DET space. *FPR* is plotted on the X-axis, *FNR* is plotted on the Y-axis, and *CCR* is plotted on the Z-axis in augmented DET space. Complexity can for instance be measured by the number of rules used by the classifier and it determines the average cost (in time) a classifier requires. Again, for instance by random linear combination, classifiers with average objective function values can be obtained for every point on a line segment that connects two classifiers - and therefore concave parts or linear parts of the Pareto front do not have to be represented. Moreover, it is taken into account that the construction of more complex classifier with the same DET performance is always possible, by adding redundant rules.

The potential classifiers lie on the surface of the augmented DET convex hull (ADCH). In 3DCH-EMOA the volume above DET surface (*VAS*) acts as a performance evaluation indicator of population quality at each generation of the algorithm. While dealing with 3D augmented DET convex hull maximization problems [9], 3DCH-EMOA can obtain solutions with good uniform distribution and also has good ability to cover only those points of a Pareto front, from which all other Pareto optimal points can be obtained by simple convex combination. No points are placed in concave parts, such as dents, as this would be a waste of computational resources. Experimental results in [9] show that 3DCH-EMOA outperforms NSGA-II [20], GDE3 (the third evolution step of Generalized Differential Evolution) [29], SPEA2 (Strength Pareto Evolutionary Algorithm 2) [30], MOEA/D [21] and SMS-EMOA [23] on the volume above surface (*VAS*) [31] performance indicator and in the *Gini* coefficient [32, 9] on the size of gaps – indicating how evenly distributed points are placed.

Also on application problems 3DCH-EMOA could obtain high quality results. More recently, 3DCH-EMOA has been successfully applied for sparse neural network optimization [9], in which, the performance of neural networks is evaluated in the augmented DET space and the sparsity is defined as the complexity objective to be optimized. 3DCH-EMOA can obtain better accuracy results than other algorithms in [9]. The three-way classification for SPAM detection was proposed in [28], in which the final user of an anti-spam filter could help in the detection task. 3DCH-EMOA was applied for SPAM detection and it performs much better than all other tested algorithms. 3DCH-EMOA has great potential for classification performance improvement in areas such as machine learning and computer vision.

However, 3DCH-EMOA performs worse than the compared methods in terms of computational time, that is when not considering the time required for function evaluations. 3DCH-EMOA revealed high consumption of computational resources due to redundant convex hull calculations. In particular, it needs to build a new convex hull many times, and at each iteration it ranks the individuals in different priority levels. Very recently, several algorithms have been developed for convex hull maximization [10, 11]. However, results focused so far on the 2D case and there was a lack of attention to efficient algorithms for the maximization of higher dimensional convex hulls. In this paper, a fast version of 3DCH-EMOA, which is denoted as 3DFCH-EMOA, is proposed by adopting incremental convex hull computation and several new strategies. The average computational time complexity of 3DCH-EMOA in each generation is improved from an average case complexity of $O(n^2 \log n)$ to $O(n \log n)$, where n is the size of population. For practical purposes, we only consider the three dimensional case because it has many applications [9, 28] and still allows the visualization of the convex hull.

In addition, this paper presents several modern algorithms for multiobjective optimization, which were not applied to this problem domain previously. Particle Swarm Optimization (PSO) is a bio-inspired metaheuristic mimicking the social behavior of bird flocking or fish schooling [33]. It has been widely used for solving multiobjective optimization problems [34, 35]. In this paper, two variants of PSO i.e., Optimized Multiobjective PSO (OMOPSO) [35] and Speed-constrained

Multiobjective PSO (SMPSO) [36] are applied to deal with multiobjective problems of augmented DET maximization. More recently, several studies focused on solving many-objective optimization problems, i.e., problems having four or more objectives [37]. NSGA-III, a reference-point based many-objective NSGA-II, was proposed in [38], which has good performance on various many-objective problems. In this paper, NSGA-III will be adopted to deal with ZED [31] and ZEJD [9] multiobjective optimization problems. Besides, Ens-MOEA/D, one of the variants of MOEA/D proposed in [39], will be applied to solve multiobjective ADCH maximization problems.

The remainder of this paper is organized as follows. The related work is introduced in Section 2. The details of 3DFCH-EMOA are described in Section 3. Section 4 presents discussion of performance evaluation results of the proposed algorithm and its comparison to the state-of-the-art and previously developed algorithms on six test functions. Section 5 provides conclusions and suggestions for future work.

2. Related Work

As it is discussed in [9], the ADCH maximization can be described as a multiobjective optimization problem, and it can be defined by Eq. (1).

$$\min_{\mathbf{x}\in\Omega} \mathbf{F}(\mathbf{x}) = \min_{\mathbf{x}\in\Omega} \Big(f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})\Big) \tag{1}$$

where $\mathbf{x}$ represents the optimization variables vector, and f_1, f_2, f_3 represent *FPR* (false positive rate), *FNR* (false negative rate) and *CCR* (classifier complexity ratio) [9], respectively. All functions have a co-domain of $[0, 1] \subset \mathbb{R}$. Usually, the points lie on the convex hull surface are non-dominated with respect to each other, but there can be non-dominated points belonging to the Pareto front that are not on the convex hull surface. This is a special characteristic of ADCH maximization problem. The aim of 3DCH-EMOA is to find a set of non-dominated solutions that covers the 3D convex hull surface, since the potential optimal classifiers lie on the convex hull surface.

The convex hull of a set of points is the smallest convex set that contains the points and it is a fundamental construction for mathematics and (computational) geometry [40, 41, 42]. The 3D convex hull T of a finite set $A \subset \mathbb{R}^3$ is given by Eq. (2),

$$T(A) \triangleq \{x : x = \sum_{i=1}^{|A|} \mathbf{a}_i \lambda_i, \sum \lambda_i = 1, 0 \leq \lambda_i \leq 1\}, \tag{2}$$

where $\mathbf{a}_i \in A$. The convex hull can be represented with a set of facets (F) and a set of adjacency ridges and vertices (V) for each facet [43]. Each ridge connects two adjacent facets, which are also called edges in 2D and 3D space. In this paper, we only consider the convex hull in 3D space, and the solutions of 3DCH-EMOA act as vertices on the convex hull surface. For a given convex hull surface (CH), we can obtain its facets, edges and vertices.

Several convex hull construction algorithms have been developed in the computational geometry community [40, 44]. The gift-wrapping algorithm presented in [41] achieves $O(n^2)$ computational time complexity. The divide-and-conquer method for 3D convex hulls, with expected $O(n \log n)$ performance was proposed in [45], however, it is difficult to implement [40]. The randomized incremental convex hull algorithm was proposed in [46], it repeatedly adds a point to the convex hull of the previously processed points. Three steps are needed to add a new point to an existing convex hull. Firstly, the visible facets for the new point and the horizon ridges on the visible facets should be found. Secondly, a cone of new facets from the point to its horizon ridges should be constructed. Thirdly, the visible facets should be deleted to form a new convex hull with the new point and the previously processed points. The computational complexity of the randomized incremental algorithm is analyzed in [47]. It has been proven that random insertions take expected time of $O(\log n)$ for 3D convex hulls. The incremental nature of this algorithm makes it attractive to be used in our algorithm.

The Quickhull algorithm was proposed in [43]. It has a time complexity of $O(n \log n)$ for 3D convex hulls. Empirical evidence was provided to show that the Quickhull algorithm uses less computer memory resources than most of the randomized incremental algorithms and executes faster for inputs with non-extreme points. Even though, the Quickhull algorithm can deal with convex hull with a certain set of points, it does not provide efficient mechanisms for dynamical updates.

The aim of 3DCH-EMOA is to find a set of solutions lying on the surface of 3D convex hull, which is constructed with population $P \subset \mathbb{R}^3$ (the population is described in objective space) and reference point(s) $R \subset \mathbb{R}^3$. We define the set of frontal solutions *(FS)* containing solutions that are located on the boundary of the convex hull, and denote it by Eq. (3).

$$FS(P) \triangleq \{p : p \in CH(P \cup R), p \in P\} \tag{3}$$

Similarly, we define the set of non-frontal solutions (non-*FS*), which is complementary to *FS* set of solutions located in the interior of the convex hull, and denote it by Eq. (4).

$$\text{non-}FS(P) \triangleq P \setminus FS(P) \tag{4}$$

The volume above DET convex hull surface (*VAS*) is defined as the volume of convex hull *CH*, and is denoted by Eq. (5).

$$\begin{aligned} VAS(P) &\triangleq Volume(CH(P \cup R)) \\ &= Volume\Big(CH(FS(P) \cup R)\Big) \end{aligned} \tag{5}$$

VAS is used as an indicator in 3DCH-EMOA to guide the evolution of the population. 3DCH-EMOA is time consuming, due to the Quickhull algorithm running many times in each generation to rank the solutions.

In this paper, we treat the procedure of evolution of 3DCH-EMOA as a process of randomized incremental 3D convex hull construction. Several strategies are adopted to speed up 3DCH-EMOA, details of these strategies are introduced in the next section.

3. 3D Fast Convex-Hull-Based EMOA

In this section, we describe the newly proposed fast version of 3DCH-EMOA, denoted as 3DFCH-EMOA. Several strategies are designed to accelerate the implementation of the algorithm:

- Firstly, we propose 3D incremental convex-hull-based (3DICH-based) sorting method, in which the solutions are ranked in two levels at most.

- Secondly, the age of the individuals in the non-*FS* set is considered for older individuals to be deleted (forgotten) first.

- Thirdly, we proposed a new method to calculate the contribution of each vertex to the volume of the convex hull by building a partial and usually small size convex hull rather than a convex hull composed by all points in the population, as it is done in 3DCH-EMOA.

- Finally, the idea of random incremental convex hull algorithm is adopted to take advantage of the prior convex hull data structure, which helps to reduce computational time by reusing the information of convex hull, rather than to rebuild the convex hull for each iteration, as it is done in 3DCH-EMOA.

3.1. 3DICH-based sorting

In 3DCH-EMOA the population is ranked into several levels with 3DCH-based (3D convex-hull-based) sorting without redundancy strategy. The redundant solutions here have the same performance in objective space as solutions in the non-redundant set. With the sorting strategy the redundant solutions will be ranked to the last priority level and will have the smallest chance to survive into the next generation. Non-redundant poor performing solutions will have a chance to survive, as the redundant solutions with good performance will be discarded to improve the diversity of the population. The procedure of ranking the solutions into several convex hull fronts is similar to non-dominance classification of the population in NSGA-II. For example, in Fig. 1 the population is sorted in three convex hull fronts, marked in different colors, by constructing different layers of convex hulls of the population.

Since only the solutions on the first level of convex hull (i.e., frontal solutions) contribute to the value of *VAS* of the whole population, it is not necessary to rank the solutions that are not on the first level of the convex hull, which is computationally expensive and doesn't contribute to *VAS*. The solutions in *FS* set obtained by 3DCH-EMOA lie on the surface of the convex hull. To obtain a good result 3DCH-EMOA should find a good approximation of the true convex hull, which not only has a large value of *VAS*, but also has a uniform distribution of vertices covering the whole

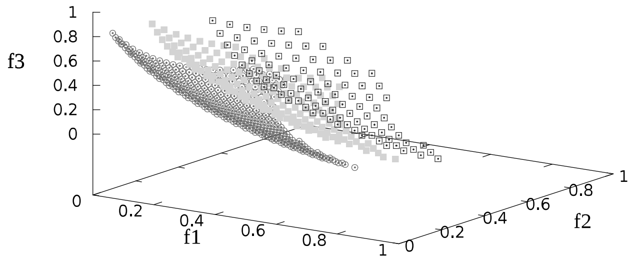


Figure 1: Ranking of population into three different levels with 3D convex-hull-based ranking without redundancy scheme in 3DCH-EMOA, the individuals in different levels are marked in different colors. The first level of solutions are marked in red, the second level of solutions are marked in green and the third level solutions are marked in blue

convex hull. Motivated by this idea, we designed a procedure of 3DFCH-EMOA to construct an incremental convex hull. In the procedure, we try to insert good solutions into the convex hull and remove bad solutions from it, while keeping the number of vertices on the convex hull equal to or less than the population size.

In this paper, we propose the 3D incremental convex-hull-based ranking (3DICH-based ranking) method. In 3DFCH-EMOA the population is classified in two sets, one is the *FS* set (*FSset*) that includes solutions on the first level of the convex hull surface (denoted as *CH* in this paper), the other one is the non-*FS* set (non-*FSset*) containing the remaining solutions, i.e., redundant solutions and solutions in the interior of the convex hull not contributing to the *VAS* and therefore not relevant to the final solution set. Solutions in *FS* set are marked in red and solutions in non-*FS* set are marked in green, as it is shown in Fig. 2. If the non-*FS* set appears to be empty, the population is ranked into one level only. The algorithm 3DICH-based sorting is described in Algorithm 1. In the algorithm, the population of solutions P and a set of reference points R are given. A convex hull CH is built with points in $P \cup R$. The solutions on the surface of CH are ranked in the first level, and the remaining solutions are ranked in the second level. Both of the ranked solution sets and the structure of CH are returned for further use. To rank a new solution in each generation we should judge whether the solution is in or out of the convex hull, which is built with the points in the first level and R. If a new solution that is out of the convex hull then it is first added to the CH and then ranked in the first level, otherwise it is ranked in the second level. We prefer to obtain a solution on the convex hull surface, as it has a chance to be a potential optimal classifier for the final decision. Generally, the time complexity of 3DICH-based sorting is equal to $O(\log n)$, where n denotes the number of vertices of CH. When the set of non-FS is empty n is equal to the population size.

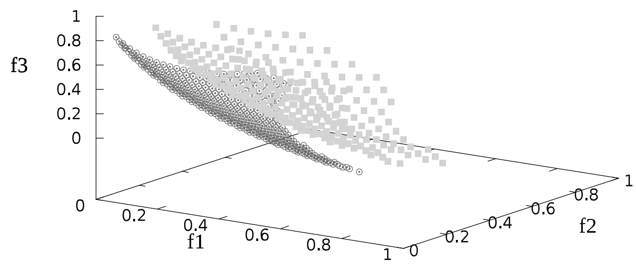


Figure 2: Ranking of the population into two different levels with 3DICH-based sorting in 3DFCH-EMOA. The individuals in different levels are marked in different colors. The first level of solutions is marked in red and the rest of solutions is marked in green

---

Algorithm 1 3DICH-based sorting *(P, R)*

Require:

$P \neq \emptyset$, $R \neq \emptyset$,
P is a solution set,
R is the set of reference points.

Ensure:

A solution set RS is ranked
and the convex hull CH is built.

$CH \leftarrow$ building convex hull with points $P \cup R$.
$FSset \leftarrow FS(P)$
non-$FSset \leftarrow P \setminus FSset$
$RS_0 \leftarrow FSset$, $RS_1 \leftarrow$ non-$FSset$.
return the ranked solution set RS={RS_0, RS_1}, and built CH.

3.2. Age-based selection

Similarly to 3DCH-EMOA, 3DFCH-EMOA adopts $(\mu + 1)$ strategy (i.e., steady state strategy), according to which a new solution is generated and added to the population in each generation. Recently it has been shown that the selection of a subset of k (k> 1) points from n points in three dimensional, maximizing the convex hull volume is NP complete [48]. This is why a single point replacement scheme is favored over a more general $(\mu + \lambda)$ selection. This yields a monotonically increasing volume. To keep the population of fixed size, a solution should be deleted in each generation. If the non-FS set is not empty, we delete the oldest individual in the set.

The age-based selection mechanism for individuals to participate in genetic operations was introduced for steady state strategies in [49, 50]. In addition, it was successfully used in Hupkens et al. [24] in the SMS-EMOA (replacing non-dominated sorting). The age of a newly generated individual is set to zero and it is increased by one at each generation. We use the age of individuals in the selection scheme, because it has low computational complexity of $O(1)$ and because more recently generated individuals are more likely to be closer to the non-dominated frontier than older ones [51].

Young individuals are selected to survive in the next generation and the oldest individual is the first element in the queue to be deleted at each generation. The age-based deletion strategy reduces the complexity of individual deletion in 3DCH-EMOA when the non-*FS* set is not empty. This process is comparably less resource consuming and requires time complexity of $O(1)$. An aging queue (*AgingQueue*) is defined to store non-*FS* solutions, in which the oldest individual is always at the head of the queue. The algorithm of age-based selection is described in Algorithm 2.

Algorithm 2 Age-based selection (*AgingQueue*, non-*FS*set)

Require:
AgingQueue that stores solutions in non-*FS*set,
non-*FS*set $\neq \varnothing$.

Ensure: *AgingQueue* and non-*FS*set are updated.

```
if non-FSset ≠ ∅ then
    q ← the first element in AgingQueue.
    Remove the first element in AgingQueue.
    non-FSset ← non-FSset \ q.
end if
return AgingQueue, non-FSset
```

3.3. Fast calculation of ΔVAS

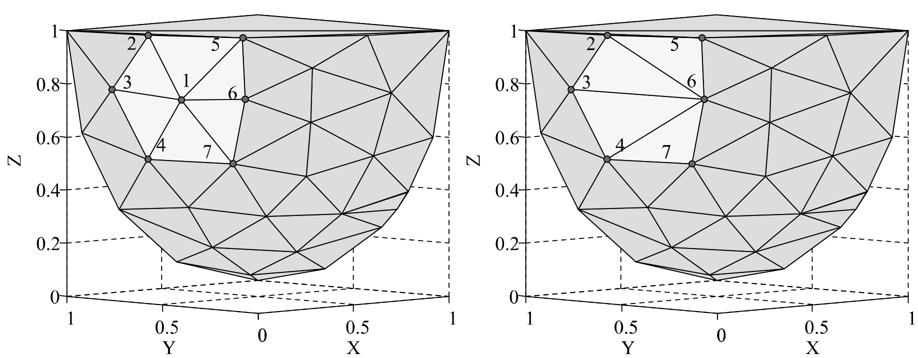
If the non-*FS* set is empty, we delete the solution that has the least contribution to the *VAS*. To rank the solutions in the *FS* set the ΔVAS of each solution should be calculated.

The theory of random incremental convex hulls [47] shows that while inserting or deleting one vertex on the convex hull, most of the vertices keep the same topological structure. Only vertices sharing the same facet with the changed (added/deleted) vertex change the connection with other vertices. As shown in Fig. 3, deletion of vertex 1 in Fig. 3(a) leads to a new convex hull in Fig. 3(b). Insertion of a new vertex 1 on the convex hull in Fig. 3(b) leads to the convex hull in Fig. 3(a). Only the local structure is changed when a vertex is inserted or deleted.

By comparing the two convex hulls in Fig. 3, we can conclude that with the insertion and deletion only the topology structure of related vertices changes. The related vertices (*RV*) are defined by the points on the convex hull that share the same facet with the vertex. The relationship of related vertices is denoted by Eq. (6).

$$RV(p) \triangleq \{q : p \in F_i, q \in F_i, p \neq q, F_i \in CH\} \tag{6}$$

where $i = 1, 2, \ldots, N_F$, N_F is the number of facets of convex hull *CH*. The algorithm to find the related vertices for a given vertex q is described in Algorithm 3. The time complexity of Algorithm 3 is $O(n)$, where n is the number of vertices on the convex hull.



(a) A convex hull with all vertices (b) A new convex hull without the vertex 1

Figure 3: Computing the *VAS* contribution for each vertex on the convex hull in 3DCH-EMOA

To make the algorithm effective, we preserve the structure of convex hull and the contribution to *VAS* of all the vertices for each generation. After insertion and deletion in each generation we only update the contribution of related vertices. To update n vertices of the convex hull, an average time complexity in $O(\log n)$ is required [40].

The importance of individuals in the convex hull is evaluated by their contribution to *VAS*, which is denoted as ΔVAS. In [31], the contribution of an individual p is obtained by subtracting the volume of a new convex hull that is constructed without the individual, from the volume of the

Algorithm 3 Finding related vertices *(CH, q)*

Require:

CH is a convex hull,

q is a vertex of CH,

N_F is the number of facets of CH,

F is the set of facets of CH.

Ensure: A set of related vertices RV is created.

```
1: RV ← ∅
2: for i ← 1 : N_F do
3:    for all p ∈ F_i do
4:       if p ≠ q and p ∉ RV then
5:          RV ← RV ∪ {p}
6:       end if
7:    end for
8: end for
9: return RV
```

initial convex hull that includes p. The contribution of solution p is calculated by Eq. (7).

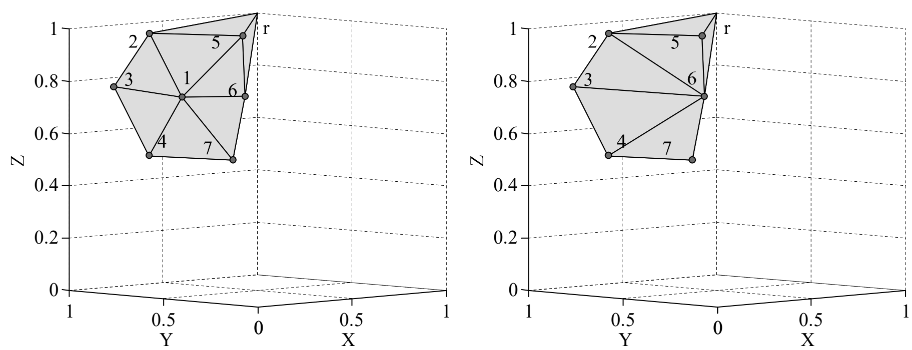
$$\Delta VAS(p) = VAS(P) - VAS(P \setminus \{p\}). \tag{7}$$

To update the contribution to VAS for each vertex, a new convex hull is built without the vertex. As shown in Fig. 3, most vertices on the convex hull keep the same topological structure with or without the vertex 1, except for vertices labeled 2, 3, 4, 5, 6 and 7, which are denoted as related vertices of vertex 1. We can calculate the contribution of vertex 1 only with each of its related vertex and each reference vertex r. The fast way to compute the contribution of vertex p is described in Eq. (8).

$$\Delta VAS_f(p) = Volume\Big(CH(RV(p) \cup \{p\} \cup \{r\})\Big) - Volume\Big(CH(RV(p) \cup \{r\})\Big) \tag{8}$$

where r is the reference vertex (r is point $(1, 1, 1)$ in the context of VAS). In the implementation of Eq. (8) the convex hull $CH(RV(p) \cup \{r\})$ is built first and then vertex p is added to CH to obtain $CH(RV(p) \cup \{p\} \cup \{r\})$.

A partial convex hull with just added vertex 1 and related vertices is shown in Fig. 4(a), another partial convex hull without vertex 1 is shown in Fig. 4(b). The contribution to VAS of vertex 1 can be obtained by calculating the *VAS* difference between the two partial convex hulls shown in Fig. 4. The approach allows reducing computational complexity especially when the size of the population is large. The algorithm of fast ΔVAS is described in Algorithm 4. We define the average number of points on the partial convex hull as m. The average time complexity of calculating a vertex's contribution is equal to $O(m \log m)$, where $m = \log n$. With the new strategy the average time complexity to update the contribution of a related vertex tends to $O(\log n)$.



(a) A partial convex hull (b) A partial convex hull without vertex 1

Figure 4: An example of calculating the *VAS* contribution of each vertex to the convex hull in 3DFCH-EMOA

Algorithm 4 Fast Δ*VAS (CH, q, r)* computation

Require:

CH is a convex hull,

q is a vertex of *CH*,

r is a reference vertex.

Ensure: *VAS* contribution of a population *q* is computed.

1: $RV \leftarrow$ Finding related vertices(CH, q).

2: $VAS_0 \leftarrow Volume(CH(RV \cup \{q\} \cup \{r\}))$.

3: $VAS_1 \leftarrow Volume(CH(RV \cup \{r\}))$.

4: $\Delta VAS \leftarrow VAS_0$-$VAS_1$.

5: **return** ΔVAS

3.4. Incremental convex hull computation

We use *CH* to denote the convex hull of the population. The information of *CH* such as facets, vertices and the contribution of each vertex to the volume of the whole convex hull is preserved in the *FS*set as discussed in Algorithm 4. The $(\mu + 1)$ selection strategy is employed in this algorithm. According to this steady state strategy only one new offspring q will be produced at each generation. When q is produced it will be judged whether it is in or out of the convex hull *CH*. If q is not yet in *CH*, it will be added to *CH*, i.e., q will be stored in the *FS*set. If q is inside *CH*, it will be stored in the non-*FS*set.

When adding q to the convex hull, some facets of convex hull will be changed, the contribution of related vertices to the convex hull volume will be affected and needs to be updated. Due to the changes caused by the introduction of q, the vertices not belonging to the convex hull *CH* will be removed from the *FS*-set and added to the end of the *AgingQueue*. The vertices not belonging to the *CH*, due to the changes caused by the introduction of q, will be removed from the convex hull. The details of adding a new point q to the convex hull *CH* are described in Algorithm 5. In the algorithm, the computational time complexity of adding a vertex to *CH* is equal to $O(\log n)$, where n is the population size. The time complexity of finding related vertices is equal to $O(n)$. And the average computational time complexity of updating the contribution of related vertices is equal to $O((\log n)^2)$. So the average computational time complexity of Algorithm 5 is equal to $O((\log n)^2)$.

To keep the population size of the algorithm constant (of size n), an individual needs to be deleted in each iteration. The head element of the *AgingQueue* will be deleted if the queue is not empty. If the *AgingQueue* is empty (all individuals are on the convex hull), the individual with least contribution to *VAS* will be deleted. Then, the convex hull will be rebuilt with the incremental convex hull algorithm and the contribution of each solution in *CH* will be updated. Details of deleting the solution q with least contribution to *VAS* from *FS*set are described in Algorithm 6. Similarly to Algorithm 5, the computational time complexity of finding related vertices is equal to $O(n)$. The computational time complexity of updating the contribution of related vertices is equal to $O((\log n)^2)$. The computational time complexity of rebuilding *CH* is equal to $O(n \log n)$. So the average computational time complexity of Algorithm 6 tends to $O(n \log n)$.

3.5. Computational time complexity of 3DFCH-EMOA

The framework of 3DFCH-EMOA is given in Algorithm 7. Both 3DCH-EMOA and 3DFCH-EMOA are general evolutionary algorithms, their computational time complexity can be described by considering one iteration of the entire algorithm. In this section, we consider the population

Algorithm 5 Adding a point to *CH* (*CH*, *FS*set, non-*FS*set, q)

Require:

CH is the convex hull,
*FS*set $\neq \varnothing$,
q is the new solution that will be added to *CH*.

Ensure:

The contribution to *CH*, *FS*set, non-*FS*set
and *AgingQueue* are updated.

```
CH ← Adding q to CH.
FSset ← FSset ∪{q}
for all p ∈ FSset do
   if p is not a vertex of CH then
      Add p to the end of AgingQueue
      non-FSset ← non-FSset ∪{p}
      FSset ← FSset \{p}
   end if
end for
RV ← Finding related vertices(CH,q).
CH.q.contribution ←Fast ΔVAS(CH,q, r).
for all p ∈RV do
   CH.p.contribution ←Fast ΔVAS(CH,p, r).
end for
return CH, FSset, non-FSset and AgingQueue.
```

Algorithm 6 Deleting a point from *CH* (*CH, FS*set, q)

Require:

CH is the convex hull,
*FS*set $\neq \varnothing$,
q is a solution that will be deleted.

Ensure: The contribution to *CH* and *FS*set are updated.

```
FSset ← FSset \{q}
RV ← Finding related vertices(CH, q).
Storing contribution of each solution of CH in TEMP.
Rebuilding CH without solution q.
Set contribution of each solution of new CH based on TEMP.
for all p ∈RV do
   CH.p.contribution ←Fast ΔVAS(CH, p, r) computation.
end for
return CH, FSset
```

Algorithm 7 3DFCH-EMOA (*MEs*, n)

Require: *MEs* (*MEs*>0) is the maximum of evaluations, n (n >0) is the population size.
Ensure: Frontal solution set (*FS*set) is created.

$P_0 \leftarrow$ init()
RS, CH $\leftarrow$ 3DICH-based sorting *(*P_0*, R)*
*FS*set $\leftarrow RS_0$, non-*FS*set $\leftarrow RS_1$
for all $p \in$ *FS*set **do**
    CH.p.contribution = Fast Δ*VAS (CH, q, r)* computation
end for
Add elements in non-*FS*set to *AgingQueue*
$t \leftarrow n$
while t < *MEs* **do**
    $t \leftarrow t + 1$, $q_t \leftarrow$ Mutate (Recombine (*FS*set $\cup$ non-*FS*set))
    if q_t is inside the convex hull *(CH)* **then**
        non-*FS*set $\leftarrow$ non-*FS*set $\cup\{q_t\}$
        Add q_t to the end of *AgingQueue*
    else
        *CH, FS*set, non-*FS*set, *AgingQueue* $\leftarrow$ Adding a point to *CH (CH, FS*set, non-*FS*set, q_t)
    end if
    if *AgingQueue* $\neq \varnothing$ **then**
        AgingQueue, non-*FS*set $\leftarrow$ Age-based selection (*AgingQueue*, non-*FS*set)
    else
        Finding the least contribution vertex p
        *CH, FS*set $\leftarrow$ Deleting a point from *CH (CH, p)*
    end if
end while
return *FS*set

to be of size n. In 3DCH-EMOA, the computational time complexity of variation operation for generating new offspring is equal to $O(n)$. 3DCH-based sorting without redundancy has the computational time complexity of $O(n^2 \log n)$. The computational time complexity of *VAS* contribution updating is equal to $O(n^2 \log n)$. The overall computational time complexity in each iteration of 3DCH-EMOA is equal to $O(n^2 \log n)$.

In 3DFCH-EMOA, the computational time complexity of variation operation for generating a new offspring is equal to $O(n)$. The average computational time complexity of 3DICH-based sorting is equal to $O(\log n)$. The computational time complexity of age-based selection is equal to $O(1)$. The computational time complexity of adding a point to *CH* is equal to $O((\log n)^2)$ and the computational time complexity of deleting a point from *CH* is equal to $O(n \log n)$. So the average computational time complexity of 3DFCH-EMOA in each iteration is equal to $O(n \log n)$.

4. Experimental Results

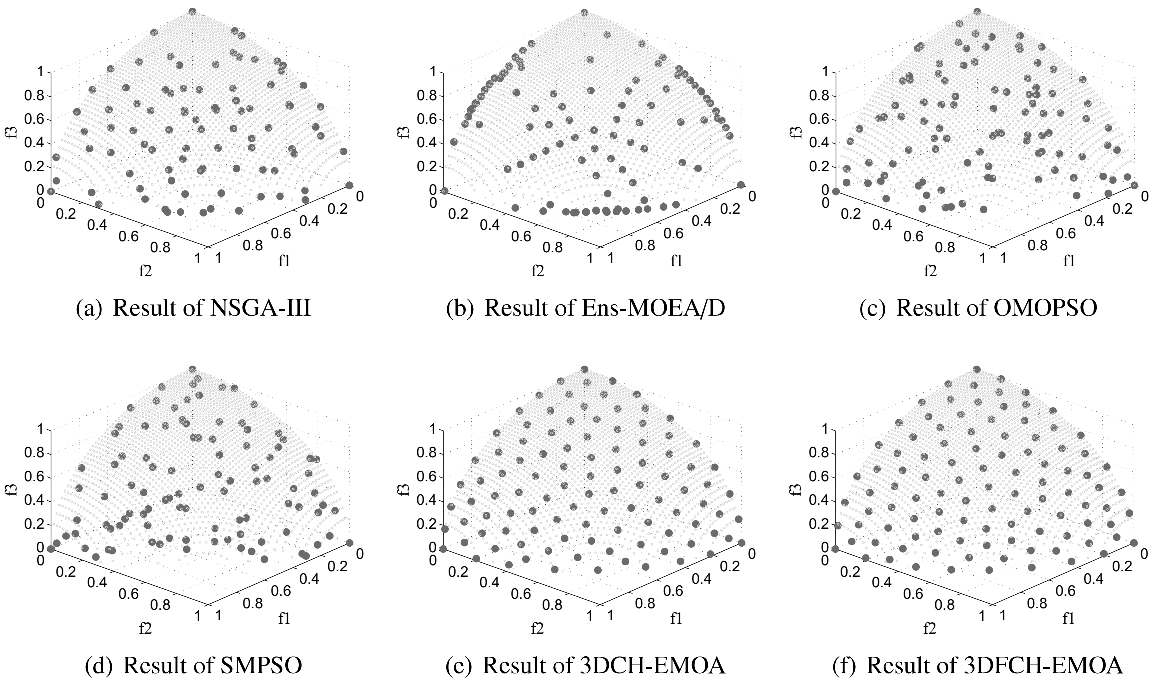
In this section, two sets of domain-specific test functions, ZED and ZEJD, are used to test the performance of 3DFCH-EMOA and several EMOAs, such as NSGA-III, Ens-MOEA/D and 3DCH-EMOA. Two PSO-based methods (i.e., OMOPSO and SMPSO) were also applied to deal with ZED and ZEJD test functions. ZED test functions were designed in [31] to evaluate the performance of 3D ROCCH maximization for three-class classification problems. ZEJD test functions are proposed in [9], which are simulations of augmented DET for parsimony binary classifiers.

Most of these experiments were performed in jMetal [52, 53], which is an optimization framework for the development of multiobjective metaheuristics in Java. The experiment for Ens-MOEA/D was performed in Matlab. All experiments were run on a desktop PC with an i5 3.2GHz processor and 4GB memory under Ubuntu 14.04LTS. For each mentioned algorithm, 30 independent trials are conducted on ZED and ZEJD test problems. For algorithms performance comparison, three groups of different experiments were carried out:

- Comparison of 3DCH-EMOA and 3DFCH-EMOA to other EMOAs, including NSGA-III, Ens-MOEA/D, OMOPSO and SMPSO on ZED and ZEJD test functions.
- Comparison of 3DFCH-EMOA and 3DCH-EMOA for runs with different population sizes (i.e., 100, 200, 300, 400, 500, 1000) on ZED test functions.
- Comparison of age-based selection to random selection of individuals in non-*FS*set for 3DFCH-EMOA.

Several metrics are chosen to evaluate the performance of studied algorithms, including volume under convex hull surface (*VAS*), *Gini* coefficient [9], pure diversity (PD) [54] and execution time:

- *VAS* metric can be used to evaluate the performance of algorithms on ZED and ZEJD test functions directly. The smallest value of *VAS* is 0, the largest value of *VAS* is bounded from



(a) Result of NSGA-III (b) Result of Ens-MOEA/D (c) Result of OMOPSO

(d) Result of SMPSO (e) Result of 3DCH-EMOA (f) Result of 3DFCH-EMOA

Figure 5: Experimental results of the Pareto front (for single run) obtained by each algorithm on ZED1 test function in $f_1 - f_2 - f_3$ space

above by 5/6 for ZED test problems and the largest value of *VAS* is bounded from above by 0.5 for ZEJD test functions. Generally, the larger the value of *VAS*, the better performance of the solution set of an algorithm.

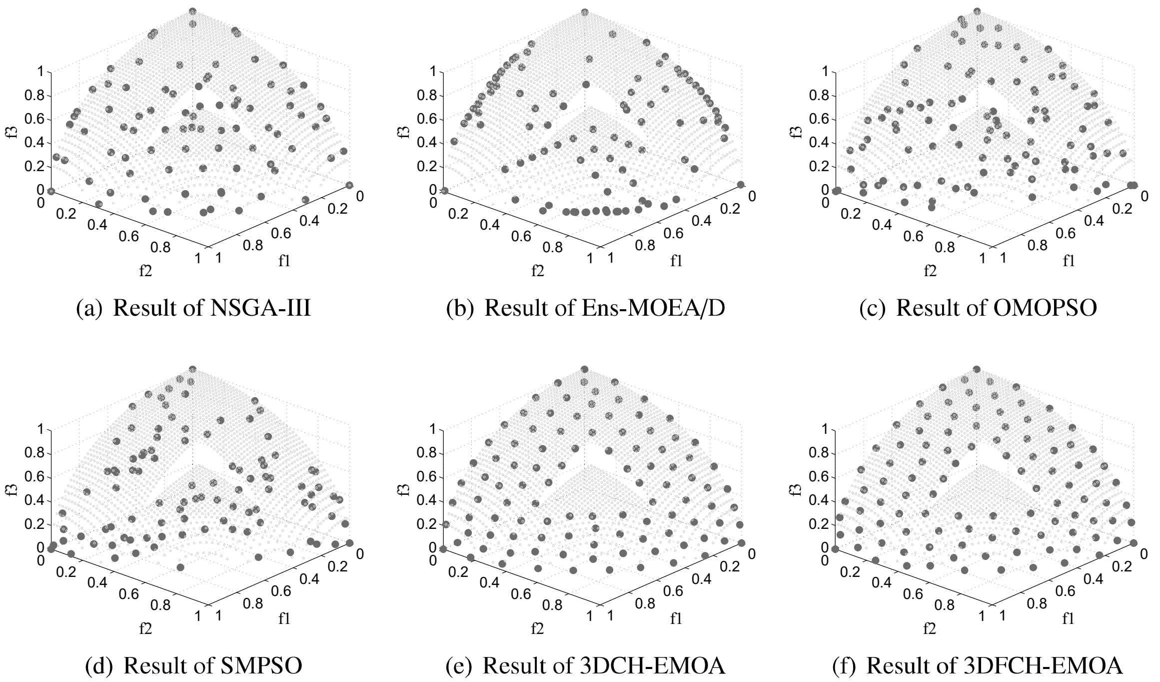
- The *Gini* coefficient was used for measuring the distribution of solutions of evolutionary algorithms in [9]. Generally, the lower the value of the *Gini* coefficient, the more evenly distributed the solution set.

- The *PD* is used as a new diversity metric in [54] to measure population diversity of evolutionary algorithms. A high population diversity leads to large value of *PD*.

- Execution time is used to measure the computational effort of all algorithms.

4.1. Comparison of 3DFCH-EMOA to other EMOAs

In this subsection performance of NSGA-III, Ens-MOEA/D, OMOPSO, SMPSO, 3DCH-EMOA and 3DFCH-EMOA is compared on ZED and ZEJD test functions.



(a) Result of NSGA-III (b) Result of Ens-MOEA/D (c) Result of OMOPSO

(d) Result of SMPSO (e) Result of 3DCH-EMOA (f) Result of 3DFCH-EMOA

Figure 6: Experimental results of the Pareto front (for single run) obtained by each algorithm on ZED2 test function in $f_1 - f_2 - f_3$ space

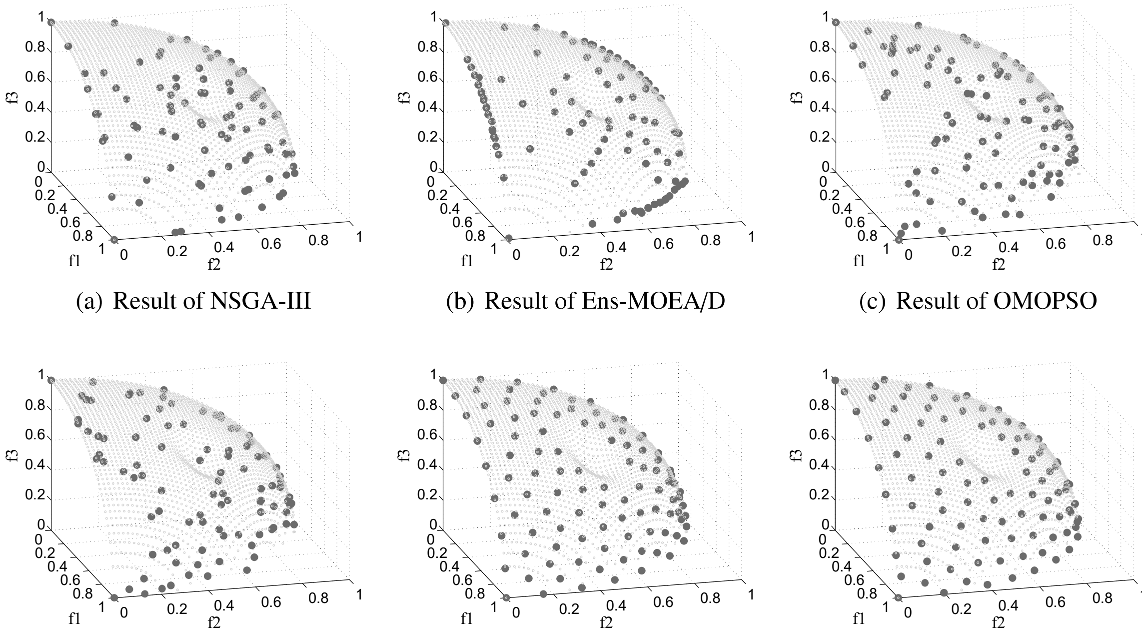
4.1.1. Parameter settings

All algorithms use a maximum of 25000 function evaluations. The simulated binary crossover (SBX), a single point crossover, and polynomial bit flip mutation operators are applied in the experiments. The crossover probability of $p_c = 0.9$ and a mutation probability of $p_m = 1/n$, where n is the population size, are chosen according to the recommendation given in [31]. In this part, the population size is set to 100 for all algorithms.

4.1.2. Experimental results and discussions

Table 1: Mean and standard deviation of *VAS* on ZED and ZEJD test problems.

	NSGA-III	Ens-MOEA/D	OMOPSO	SMPSO	3DCH-EMOA	3DFCH-EMOA
ZED1	$3.48e-01_{5.20e-04}$	$3.44e-01_{2.16e-04}$	$3.37e-01_{2.66e-03}$	$3.38e-01_{2.34e-03}$	$3.53e-01_{1.70e-05}$	$3.53e-01_{2.80e-05}$
ZED2	$3.44e-01_{8.20e-04}$	$3.42e-01_{2.76e-04}$	$3.34e-01_{2.75e-03}$	$3.35e-01_{2.46e-03}$	$3.52e-01_{2.07e-05}$	$3.52e-01_{2.12e-05}$
ZED3	$3.46e-01_{5.99e-04}$	$3.42e-01_{2.29e-04}$	$3.35e-01_{2.63e-03}$	$3.36e-01_{2.29e-03}$	$3.51e-01_{1.77e-05}$	$3.51e-01_{3.22e-05}$
ZEJD1	$4.65e-01_{4.24e-06}$	$4.63e-01_{1.20e-04}$	$4.62e-01_{6.02e-04}$	$4.63e-01_{4.30e-04}$	$4.65e-01_{2.04e-06}$	$4.65e-01_{2.08e-06}$
ZEJD2	$4.64e-01_{1.66e-05}$	$4.63e-01_{1.06e-04}$	$4.62e-01_{7.06e-04}$	$4.62e-01_{4.92e-04}$	$4.65e-01_{2.09e-06}$	$4.65e-01_{2.16e-06}$
ZEJD3	$4.64e-01_{1.52e-05}$	$4.62e-01_{1.29e-04}$	$4.61e-01_{5.79e-04}$	$4.62e-01_{4.19e-04}$	$4.64e-01_{1.59e-06}$	$4.64e-01_{1.54e-06}$



(a) Result of NSGA-III (b) Result of Ens-MOEA/D (c) Result of OMOPSO

(d) Result of SMPSO (e) Result of 3DCH-EMOA (f) Result of 3DFCH-EMOA

Figure 7: Experimental results of the Pareto front (for single run) obtained by each algorithm on ZED3 test function in $f_1 - f_2 - f_3$ space

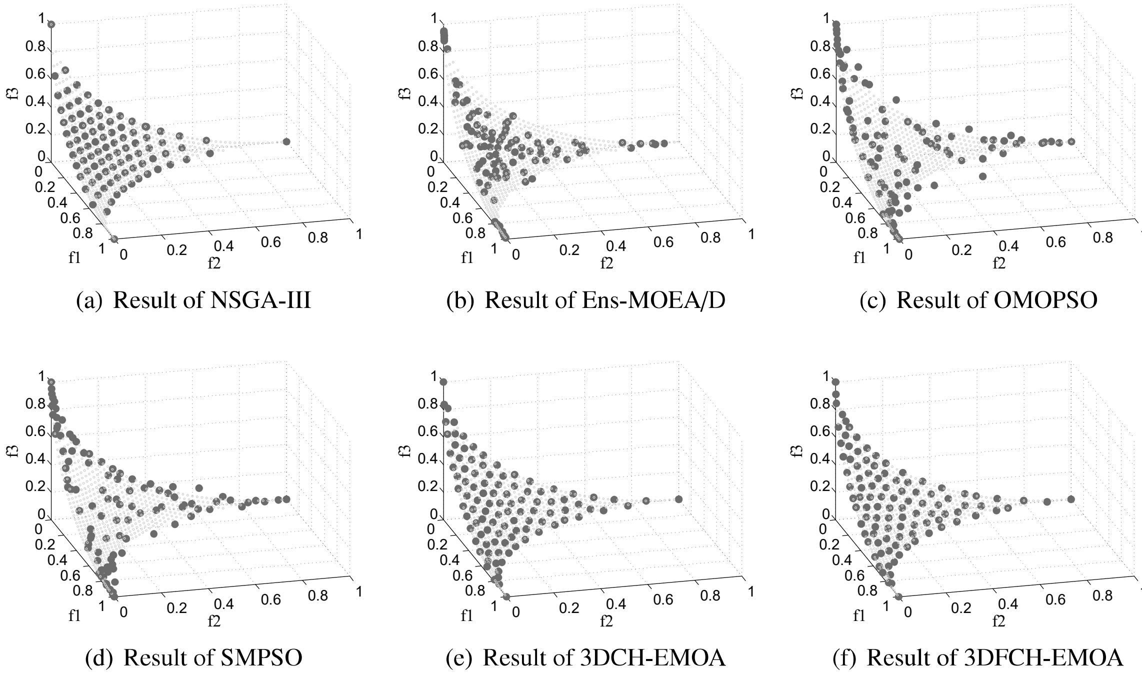
Table 2: Mean and standard deviation of *Gini* Coefficient on ZED and ZEJD test problems.

	NSGA-III	Ens-MOEA/D	OMOPSO	SMPSO	3DCH-EMOA	3DFCH-EMOA
ZED1	$3.70e-01_{4.44e-02}$	$4.24e-01_{1.40e-02}$	$2.46e-01_{2.68e-02}$	$2.64e-01_{1.99e-02}$	$4.44e-02_{3.69e-03}$	$4.75e-02_{4.61e-03}$
ZED2	$3.65e-01_{3.64e-02}$	$4.40e-01_{1.51e-02}$	$2.47e-01_{1.89e-02}$	$2.58e-01_{2.32e-02}$	$5.22e-02_{6.14e-03}$	$5.70e-02_{6.72e-03}$
ZED3	$3.65e-01_{4.02e-02}$	$4.27e-01_{1.38e-02}$	$2.48e-01_{2.62e-02}$	$2.59e-01_{1.93e-02}$	$5.90e-02_{6.12e-03}$	$6.75e-02_{1.09e-02}$
ZEJD1	$1.57e-01_{6.99e-03}$	$2.63e-01_{1.71e-02}$	$2.86e-01_{2.01e-02}$	$2.59e-01_{1.95e-02}$	$7.30e-02_{1.13e-02}$	$7.41e-02_{1.05e-02}$
ZEJD2	$1.67e-01_{6.56e-03}$	$2.75e-01_{1.95e-02}$	$2.87e-01_{2.16e-02}$	$2.64e-01_{2.14e-02}$	$7.84e-02_{1.09e-02}$	$7.69e-02_{1.01e-02}$
ZEJD3	$1.71e-01_{1.17e-02}$	$3.01e-01_{1.91e-02}$	$2.82e-01_{2.21e-02}$	$2.66e-01_{2.50e-02}$	$1.16e-01_{1.24e-02}$	$1.21e-01_{1.09e-02}$

Table 3: Mean and standard deviation of *PD* on ZED and ZEJD test problems.

	NSGA-III	Ens-MOEA/D	OMOPSO	SMPSO	3DCH-EMOA	3DFCH-EMOA
ZED1	$9.94e+04_{1.31e+04}$	$1.70e+05_{6.35e+03}$	$2.13e+05_{1.36e+04}$	$2.09e+05_{9.66e+03}$	$1.97e+05_{8.23e+03}$	$1.97e+05_{1.09e+04}$
ZED2	$1.06e+05_{1.20e+04}$	$1.61e+05_{5.80e+03}$	$2.10e+05_{8.73e+03}$	$2.10e+05_{7.33e+03}$	$1.80e+05_{8.66e+03}$	$1.80e+05_{8.12e+03}$
ZED3	$1.12e+05_{9.92e+03}$	$1.69e+05_{5.67e+03}$	$2.09e+05_{9.39e+03}$	$2.10e+05_{1.13e+04}$	$1.88e+05_{8.51e+03}$	$1.88e+05_{8.06e+03}$
ZEJD1	$4.51e+04_{3.34e+03}$	$9.65e+04_{3.12e+03}$	$8.12e+04_{4.77e+03}$	$7.95e+04_{5.07e+03}$	$1.04e+05_{6.58e+03}$	$1.04e+05_{4.62e+03}$
ZEJD2	$4.53e+04_{1.97e+03}$	$9.38e+04_{4.23e+03}$	$8.24e+04_{5.25e+03}$	$8.32e+04_{6.68e+03}$	$9.58e+04_{4.66e+03}$	$9.67e+04_{4.84e+03}$
ZEJD3	$6.03e+04_{4.33e+03}$	$9.31e+04_{5.81e+03}$	$8.25e+04_{5.50e+03}$	$8.56e+04_{6.63e+03}$	$8.81e+04_{5.32e+03}$	$8.80e+04_{4.11e+03}$

Firstly, the Pareto fronts (for one run) of ZED and ZEJD are shown in Fig. 5-10. By analyzing the Pareto fronts of ZED1 function obtained by the algorithms we can make some conclusions: 1) OMOPSO and SMPSO have the worst ability to deal with ZED1 test function not only for



(a) Result of NSGA-III (b) Result of Ens-MOEA/D (c) Result of OMOPSO

(d) Result of SMPSO (e) Result of 3DCH-EMOA (f) Result of 3DFCH-EMOA

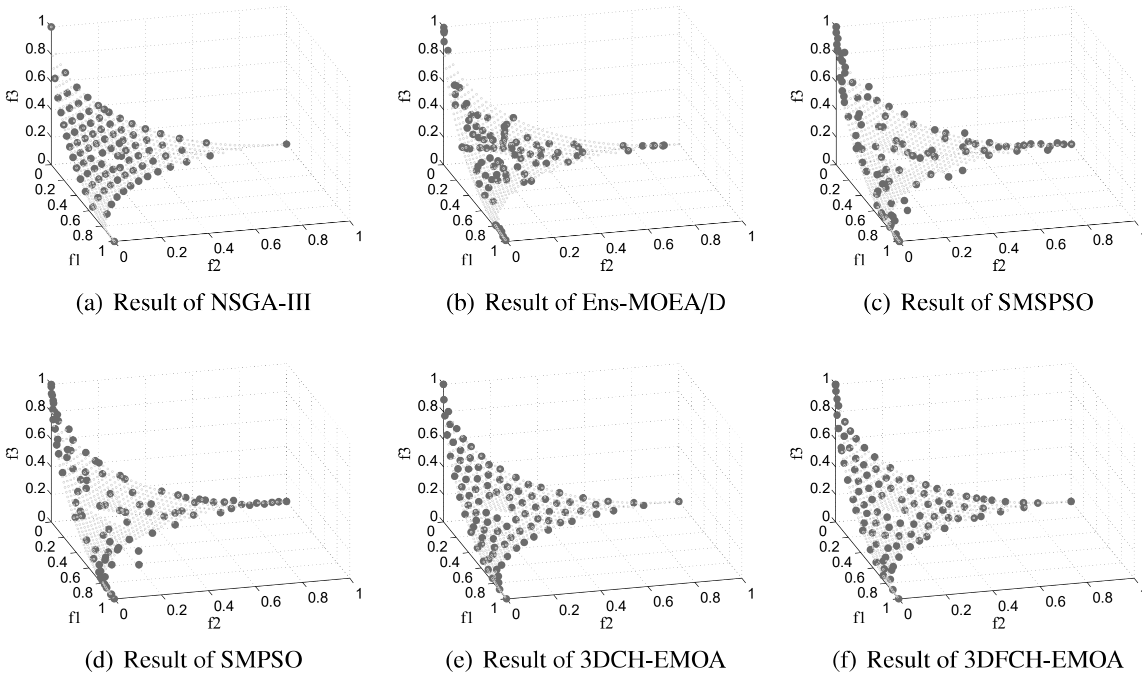
Figure 8: Experimental results of the Pareto front (for single run) obtained by each algorithm on ZEJD1 test function in $f_1 - f_2 - f_3$ space

Table 4: Mean and standard deviation of execution time (ms) on ZED and ZEJD test problems.

	NSGA-III	Ens-MOEA/D	OMOPSO	SMPSO	3DCH-EMOA	3DFCH-EMOA
ZED1	$3.18e+05_{3.22e+03}$	$6.94e+04_{1.13e+03}$	$2.17e+03_{9.20e+01}$	$4.02e+02_{5.71e+01}$	$4.68e+05_{3.70e+04}$	$6.33e+03_{5.68e+02}$
ZED2	$3.16e+05_{3.21e+03}$	$6.90e+04_{9.85e+02}$	$2.12e+03_{9.00e+01}$	$4.02e+02_{5.40e+01}$	$4.40e+05_{3.53e+04}$	$4.79e+03_{3.11e+02}$
ZED3	$3.17e+05_{2.19e+03}$	$6.96e+04_{1.63e+03}$	$2.09e+03_{9.29e+01}$	$4.03e+02_{7.15e+01}$	$4.53e+05_{3.47e+04}$	$5.35e+03_{7.06e+02}$
ZEJD1	$2.95e+05_{3.11e+03}$	$6.88e+04_{9.99e+02}$	$3.73e+02_{1.66e+01}$	$2.14e+02_{1.16e+01}$	$2.28e+05_{4.45e+03}$	$5.70e+03_{8.04e+02}$
ZEJD2	$2.94e+05_{3.10e+03}$	$6.88e+04_{1.20e+03}$	$3.70e+02_{1.22e+01}$	$2.14e+02_{6.33e+00}$	$2.11e+05_{4.54e+03}$	$5.22e+03_{6.84e+02}$
ZEJD3	$2.91e+05_{3.12e+03}$	$6.88e+04_{1.56e+03}$	$3.61e+02_{1.21e+01}$	$2.15e+02_{8.17e+00}$	$1.85e+05_{3.71e+03}$	$5.53e+03_{8.58e+02}$

convergence but also for uniformity metrics; 2) Ens-MOEA/D does not have a good uniformity of the Pareto front distribution, as it found too many solutions on the edges of objective space; 3) NSGA-III performs better than OMOPSO, SMPSO and Ens-MOEA/D; 4) 3DCH-EMOA and 3DFCH-EMOA perform better than the others not only for convergence but also for uniformity metrics.

By comparing the Pareto fronts of ZED2 and ZED3 we can draw the same conclusions as for ZED1. Besides, we can see that only 3DCH-EMOA and 3DFCH-EMOA can avoid sampling solutions in the dent area, which is better because these regions contain only redundant solutions



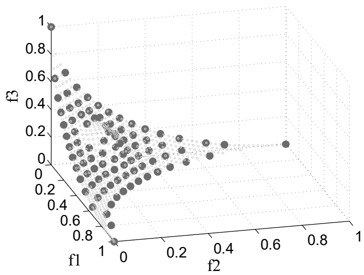
(a) Result of NSGA-III (b) Result of Ens-MOEA/D (c) Result of SMSPSO

(d) Result of SMPSO (e) Result of 3DCH-EMOA (f) Result of 3DFCH-EMOA

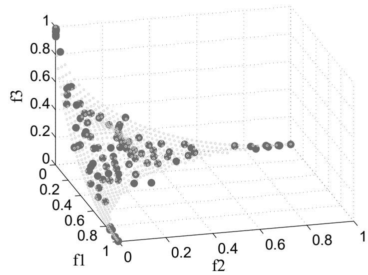
Figure 9: Experimental results of the Pareto front (for single run) obtained by algorithms on ZEJD2 test function in $f_1 - f_2 - f_3$ space

since the goal is to represent the convex hull. Moreover, it performs well on problems with discontinuities (ZED2 test problem) and continuous (ZED3 test problem) objective space.

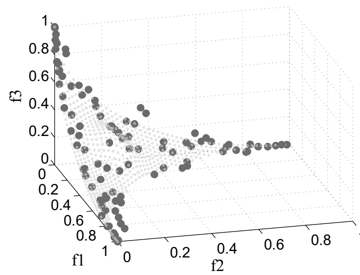
By analyzing the Pareto fronts of ZEJD1 obtained by the algorithms we can make some conclusions: 1) OMOPSO and SMPSO have the worst ability to deal with ZEJD1 test function, not only for convergence but also for uniformity metrics; 2) Ens-MOEA/D has better performance in terms of convergence than OMOPSO and SMPSO, but it does not have the ability to find solutions with good distribution on the Pareto front; 3) NSGA-III, 3DCH-EMOA and 3DFCH-EMOA can find solutions with good uniformity and convergence metrics. NSGA-III has the most uniformly distributed solutions on the Pareto front on ZEJD1 test function. As it is pointed in [54] that a solution set with good uniformity does not necessarily mean that it also has good diversity. Solutions with good uniformity should have the same dissimilarity with their neighbors, however, solutions with good diversity can provide decision makers the maximum amount of information. The diversity of all the results will be discussed later.



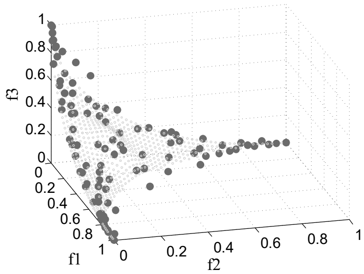
(a) Result of NSGA-III



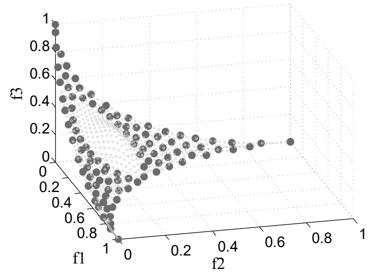
(b) Result of Ens-MOEA/D



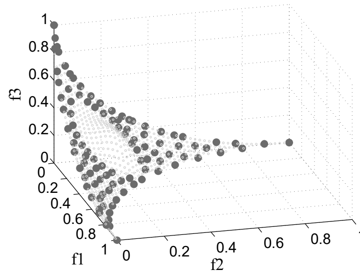
(c) Result of OMOPSO



(d) Result of SMPSO

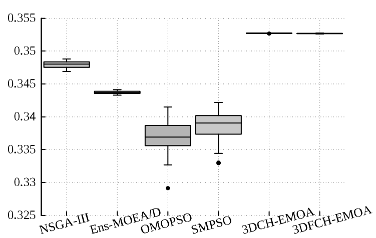


(e) Result of 3DCH-EMOA

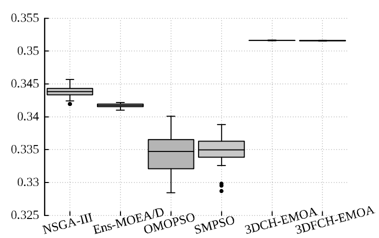


(f) Result of 3DFCH-EMOA

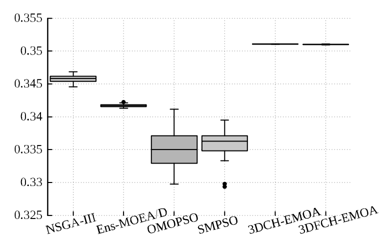
Figure 10: Experimental results of the Pareto front (for single run) obtained by each algorithm on ZEJD3 test function in $f_1 - f_2 - f_3$ space



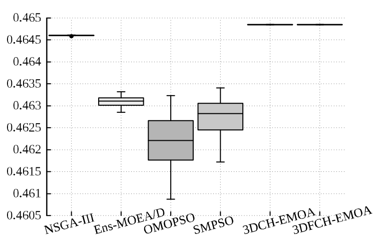
(a) Box-plot for ZED1 test problem



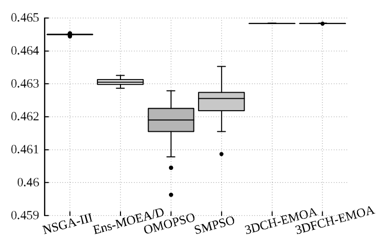
(b) Box-plot for ZED2 test problem



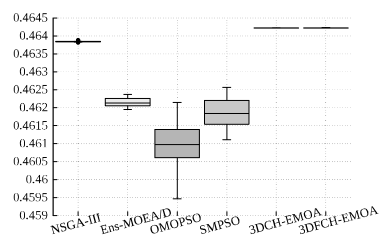
(c) Box-plot for ZED3 test problem



(d) Box-plot for ZEJD1 test problem



(e) Box-plot for ZEJD2 test problem



(f) Box-plot for ZEJD3 test problem

Figure 11: Box-plots of *VAS* for all algorithms on ZED and ZEJD test functions, each box-plot is generated by running 30 independent trials

By comparing the Pareto fronts of ZEJD2 and ZEJD3 we can draw the same conclusions as for ZEJD1. NSGA-III can obtain solutions with good uniformity on the surface of Pareto fronts, but in the area of machine learning, for problems such as ADCH maximization problem, the solutions in the concave area make no contribution for classification [9]. In addition, we can see that only 3DCH-EMOA and 3DFCH-EMOA omit the concave area of ZEJD2 and ZEJD3, which allows them to find more solutions in parts of the convex hull that are relevant for maximizing *VAS*. The solutions, which are on the Pareto front but not on the convex hull surface, do not contribute to *VAS*.

By comparing the Pareto fronts of all the algorithms we can conclude that 3DFCH-EMOA obtains results as good as 3DCH-EMOA. NSGA-III can capture the Pareto fronts of ZEJD test functions very well, but it can not avoid sampling solutions in the dent areas of ZEJD2 and ZEJD3 test problems. Solutions in the dent area, i.e., solutions on the Pareto front but not on the convex hull surface, do not provide better performance of classifiers when compared to those on the convex hull surface [9].

The statistical results (means and standard variances) of the *VAS* are shown in Table 1. *VAS* is the most important indicator in this study as it measures the size of the objective space that is either dominated by a point in the population or by a linear combination of such points [9]. *VAS* box-plots are shown in Fig. 11. Fig. 11(a) shows the results of all algorithms' performance on ZED1 test function. In the figure we can see that 3DFCH-EMOA can obtain as good results as 3DCH-EMOA, and outperform other algorithms not only on the average *VAS* but also when considering standard deviations. NSGA-III outperforms Ens-MOEA/D, OMOPSO and SMPSO algorithms. By comparing algorithms performance on other test problems, we can also see that 3DFCH-EMOA and 3DCH-EMOA can obtain the best result on *VAS* metric. This confirms that 3DFCH-EMOA has successfully inherited the good performance of 3DCH-EMOA. From the table we can see that 3DFCH-EMOA can obtain as good results as 3DCH-EMOA for the *VAS* evaluation. 3DFCH-EMOA and 3DCH-EMOA outperform the other EMOAs on all ZED test problems. When dealing with ZEJD test problems, NSGA-III, 3DCH-EMOA and 3DFCH-EMOA perform better than the other EMOAs.

The statistical results of *Gini* coefficient are shown in Table 2. From the table we can see that

3DCH-EMOA and 3DFCH-EMOA outperform the other algorithms for most of the test problems. 3DCH-EMOA is slightly better than 3DFCH-EMOA for most of the test problems. NSGA-III performs better than the other algorithms except 3DCH-EMOA and 3DFCH-EMOA on ZED and ZEJD test functions.

The statistical results of *PD* diversity metric are shown in Table 3. OMOPSO and SMOSO have good diversity metric results, but not very good convergence metric results as discussed above. 3DFCH-EMOA performs as good as 3DCH-EMOA for the diversity metric. 3DFCH-EMOA and 3DCH-EMOA can obtain higher values of *PD* than NSGA-III.

The statistical results on the execution times are shown in Table 4. SMPSO has always the lowest execution time and OMOPSO performs better than the other algorithms except of SMPSO. 3DFCH-EMOA outperforms the other algorithms except of SMPSO and OMOPSO. NSGA-III spends slightly more time than 3DCH-EMOA on ZEJD test functions. 3DCH-EMOA uses more than 30 times as much computational time as 3DFCH-EMOA algorithm, that is to confirm that the new algorithm speeds up 3DCH-EMOA about more than 30 times with the population size 100.

4.2. Comparison of 3DFCH-EMOA to 3DCH-EMOA

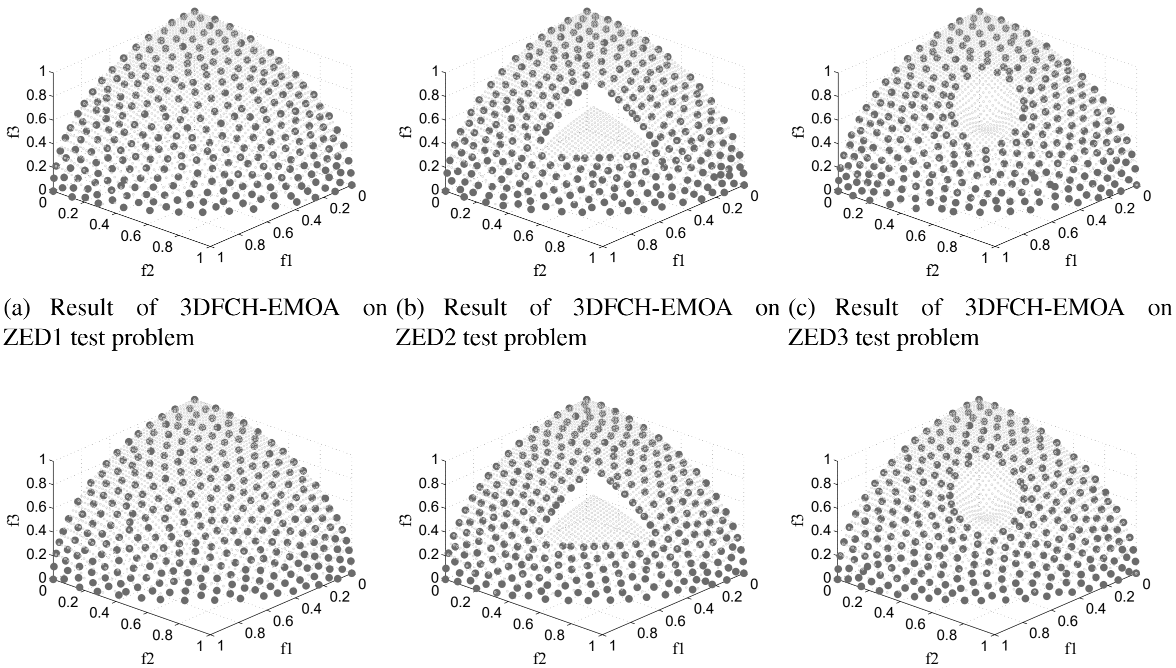
We tested 3DFCH-EMOA and 3DCH-EMOA on ZED test functions with different population sizes (100, 200, 300, 400, 500, 1000). For each mentioned parameter, 30 independent trials were run.

4.2.1. Parameter settings

All algorithms run for 25000 function evaluations. The simulated binary crossover (SBX) operator and the polynomial mutation are applied in all experiments. The crossover probability of $p_c = 0.9$ and a mutation probability of $p_m = 1/n$, where n is the number of decision variables, were used as recommended in [31]. The population size was set to 100, 200, 300, 400, 500, 1000 for all the test problems.

4.2.2. Experimental results and discussions

The Pareto fronts obtained by 3DCH-EMOA and 3DFCH-EMOA with population size 300 are shown in Fig. 12. From the figures we can observe that 3DFCH-EMOA can obtain as good Pareto



(a) Result of 3DFCH-EMOA on ZED1 test problem

(b) Result of 3DFCH-EMOA on ZED2 test problem

(c) Result of 3DFCH-EMOA on ZED3 test problem

(d) Result of 3DCH-EMOA on ZED1 test problem

(e) Result of 3DCH-EMOA on ZED2 test problem

(f) Result of 3DCH-EMOA on ZED3 test problem

Figure 12: Experimental results of the Pareto front (for single run) obtained by 3DFCH-EMOA and 3DCH-EMOA algorithms on ZED test functions for population size of 300 in $f_1 - f_2 - f_3$ space.

fronts as 3DCH-EMOA.

The results of *VAS* mean are listed in Table 5. By comparing the values of *VAS* we can see that 3DFCH-EMOA can obtain the same values of *VAS* as 3DCH-EMOA. We can conclude that 3DFCH-EMOA inherits from 3DCH-EMOA the good performance of 3D ROCCH maximization.

The results of mean execution time of 3DFCH-EMOA and 3DCH-EMOA on ZED test functions are listed in Table 6. Execution time analysis for several population sizes for ZED1 function is shown in Fig. 13. By comparing the results we can see that execution time increases with the increase of population size. The execution time of 3DCH-EMOA increases faster than 3DFCH-EMOA with the increase of population size. Besides, the 3DCH-EMOA is computationally more expensive than 3DFCH-EMOA for the same population size.

Table 5: The mean of *VAS* on ZED test problems.

test function	population size	compared methods	
		3DFCH-EMOA	3DCH-EMOA
ZED1	100	$3.53e-01$	$3.53e-01$
	200	$3.55e-01$	$3.55e-01$
	300	$3.56e-01$	$3.56e-01$
	400	$3.56e-01$	$3.56e-01$
	500	$3.56e-01$	$3.56e-01$
	1000	$3.56e-01$	$3.56e-01$
ZED2	100	$3.52e-01$	$3.52e-01$
	200	$3.54e-01$	$3.54e-01$
	300	$3.54e-01$	$3.54e-01$
	400	$3.54e-01$	$3.54e-01$
	500	$3.55e-01$	$3.55e-01$
	1000	$3.55e-01$	$3.55e-01$
ZED3	100	$3.51e-01$	$3.51e-01$
	200	$3.53e-01$	$3.53e-01$
	300	$3.54e-01$	$3.54e-01$
	400	$3.54e-01$	$3.54e-01$
	500	$3.54e-01$	$3.54e-01$
	1000	$3.55e-01$	$3.55e-01$

Table 6: The mean of execution time (ms) on ZED test functions.

test function	population size	compared methods	
		3DFCH-EMOA	3DCH-EMOA
ZED1	100	$6.33e+03$	$4.68e+05$
	200	$4.72e+04$	$1.40e+06$
	300	$1.04e+05$	$3.42e+06$
	400	$1.79e+05$	$6.48e+06$
	500	$2.87e+05$	$1.07e+07$
	1000	$9.87e+05$	$4.84e+07$
ZED2	100	$4.79e+03$	$4.40e+05$
	200	$4.50e+04$	$1.32e+06$
	300	$9.77e+04$	$3.22e+06$
	400	$1.69e+05$	$6.11e+06$
	500	$2.70e+05$	$1.00e+07$
	1000	$8.95e+05$	$4.50e+07$
ZED3	100	$5.35e+03$	$4.53e+05$
	200	$4.64e+04$	$1.35e+06$
	300	$1.00e+05$	$3.29e+06$
	400	$1.74e+05$	$6.26e+06$
	500	$2.78e+05$	$1.03e+07$
	1000	$9.28e+05$	$4.62e+07$

4.3. Comparison of age-based selection with random selection of 3DFCH-EMOA

In this subsection, we evaluate and analyze the strategies of age-based selection and random selection of individuals in non-*FS* set. We run age-based selection and random selection on ZED1 test function for 30 independent runs and recorded the values of *VAS* in every generation. The average *VAS* over generations in 30 independent runs is shown in Fig. 14. We found that the age-based selection has a slightly faster convergence rate than the random selection strategy. Simply adopting the age-based strategy cannot improve the performance of the algorithm significantly. To

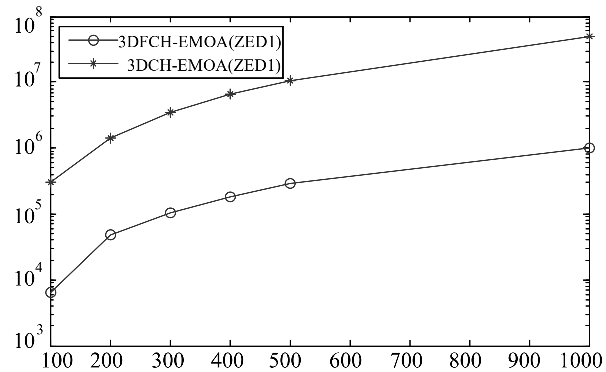


Figure 13: Comparison of execution times of 3DFCH-EMOA and 3DCH-EMOA on ZED1 test function obtained by 30 independent trials with different population sizes

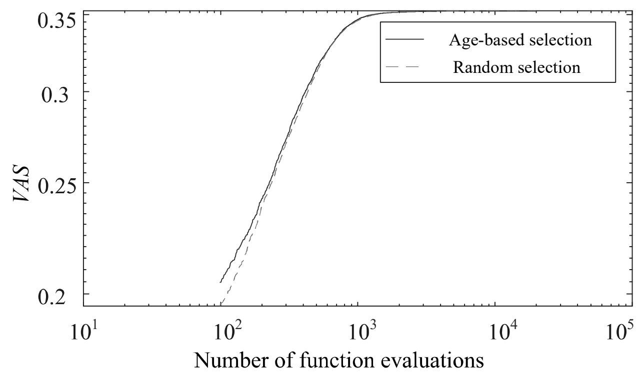


Figure 14: Average *VAS* of age-based selection and random selection of 3DFCH-EMOA on ZED1 test function

make the proposed algorithm more efficient, the age-based strategy must be applied in combination with other strategies.

5. Conclusions

In this paper, we proposed 3DFCH-EMOA, a fast version of 3DCH-EMOA, by adopting the incremental convex hull algorithm and several other evolutionary strategies. To reduce the computational time complexity of an iteration, individuals are only ranked into two levels, one is convex hull level and the other one is non-convex hull level, where age is used as a selection criterion. Besides a fast incremental computation of the contribution of each vertex to the convex hull volume is used. In total the average time complexity of 3DCH-EMOA in each generation is reduced from $O(n^2 \log n)$ to $O(n \log n)$. Six test function problems were used to test the performance of the proposed method. Experimental results show that the 3DFCH-EMOA can speed up 3DCH-EMOA for about 30 times with the size of the population 100, without reducing the performance of the method. Moreover, the benchmark was extended by modern algorithms, such as NSGA-III and MOPSO.

Alternatively, the computational time complexity of iteratively computing the convex hull, which is $O(n^{(\lfloor d/2 \rfloor + 2)})$ [55] for d dimensions, could be achieved by iteratively using the gift-wrapping algorithm. However, also here, incremental algorithms might prove to be useful for obtaining a better average computational time complexity. In the future it would be interesting to derive fast algorithms for more than 3 dimensions. Besides, since Particle Swarm Optimization (PSO) methods show a relatively fast convergence, its search operators might be combined with indicator based selection of 3DFCH-EMOA in the future.

Acknowledgments

This work was partially supported by the National Key Research and Development Plan (No. 2016YFC0600908), the National Natural Science Foundation of China (No. U1610124 and 61473215), the National Basic Research Program (973 Program) of China (No. 2013CB329402).

References

[1] T. Fawcett, An introduction to ROC analysis, Pattern recognition letters 27 (8) (2006) 861–874.

[2] A. F. Martin, G. R. Doddington, T. Kamm, M. Ordowski, M. A. Przybocki, The det curve in assessment of decision task performance, in: European Conference on Speech Communication and Technology, Eurospeech 1997, Rhodes, Greece, September, 1997, pp. 1895–1898.

[3] J. A. Hanley, Receiver operating characteristic (ROC) methodology: the state of the art., Critical Reviews in Computed Tomography 29 (3) (1989) 307–335.

[4] T. Fawcett, Using rule sets to maximize ROC performance, in: IEEE International Conference on Data Mining, 2001, pp. 131–138. doi:10.1109/ICDM.2001.989510.

[5] M. Barreno, A. A. Cárdenas, J. D. Tygar, Optimal ROC curve for a combination of classifiers, in: J. C. Platt, D. Koller, Y. Singer, S. T. Roweis (Eds.), Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December, Curran Associates, Inc., 2008, pp. 57–64.

[6] T. Fawcett, PRIE: a system for generating rulelists to maximize ROC performance, Data Mining and Knowledge Discovery 17 (2) (2008) 207–224.

[7] P. Wang, K. Tang, T. Weise, E. Tsang, X. Yao, Multiobjective genetic programming for maximizing ROC performance, Neurocomputing 125 (2014) 102–118.

[8] P. Wang, M. Emmerich, R. Li, K. Tang, T. Bäck, X. Yao, Convex hull-based multi-objective genetic programming for maximizing receiver operator characteristic performance, IEEE Transactions on Evolutionary Computation 19 (2) (2015) 188–200. `doi:10.1109/TEVC.2014.2305671`.

[9] J. Zhao, V. Basto Fernandes, L. Jiao, I. Yevseyeva, A. Maulana, R. Li, T. Bäck, K. Tang, M. T.M. Emmerich, Multiobjective optimization of classifiers by means of 3D convex-hull-based evolutionary algorithms, Information Sciences 367–368 (2016) 80–104.

[10] W. Hong, K. Tang, Convex hull-based multi-objective evolutionary computation for maximizing receiver operating characteristics performance, Memetic Computing 8 (1) (2016) 35–44.

[11] W. Hong, G. Lu, P. Yang, Y. Wang, K. Tang, A new evolutionary multi-objective algorithm for convex hull maximization, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 931–938.

[12] K. Li, S. Kwong, Q. Zhang, K. Deb, Interrelationship-based selection for decomposition multiobjective optimization, IEEE Transactions on Cybernetics 45 (10) (2015) 2076–2088.

[13] T. Liu, L. Jiao, W. Ma, J. Ma, R. Shang, Cultural quantum-behaved particle swarm optimization for environmental/economic dispatch , Applied Soft Computing 48 (2016) 597–611.

[14] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. Coello, Survey of multiobjective evolutionary algorithms for data mining: Part II, IEEE Transactions on Evolutionary Computation 18 (1) (2014) 20–35. `doi:10.1109/TEVC.2013.2290082`.

[15] S. Wang, L. L. Minku, X. Yao, A multi-objective ensemble method for online class imbalance learning, in: 2014 International Joint Conference on Neural Networks (IJCNN), 2014, pp. 3311–3318. `doi:10.1109/IJCNN.2014.6889545`.

[16] L. Li, X. Yao, R. Stolkin, M. Gong, S. He, An evolutionary multiobjective approach to sparse reconstruction, IEEE Transactions on Evolutionary Computation 18 (6) (2014) 827–845. `doi:10.1109/TEVC.2013.2287153`.

[17] H. Li, M. Gong, Q. Wang, J. Liu, L. Su, A multiobjective fuzzy clustering method for change detection in sar images, Applied Soft Computing 46 (C) (2016) 767–777.

[18] I. Yevseyeva, V. Basto-Fernandes, J. R. Méndez, Survey on anti-spam single and multi-objective optimization, in: ENTERprise Information Systems, Springer, 2011, pp. 120–129.

[19] V. Basto-Fernandes, I. Yevseyeva, J. R.Méndez, Anti-spam multiobjective genetic algorithms optimization analysis, International Resource Management Journal 26 (2012) 54–67.

[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[21] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[22] Z. Wang, Q. Zhang, A. Zhou, M. Gong, L. Jiao, Adaptive replacement strategies for MOEA/D, IEEE Transactions on Cybernetics 46 (2) (2016) 474–486.

[23] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.

[24] I. Hupkens, M. Emmerich, Logarithmic-time updates in SMS-EMOA and hypervolume-based archiving, in: EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation, 2013, pp. 155–169.

[25] K. Bringmann, T. Friedrich, F. Neumann, M. Wagner, Approximation-guided evolutionary multi-objective optimization., Proceedings of the twenty-second international joint conference on artifical intellignece (2011) 1846–1853.

[26] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, Lecture Notes in Computer Science (2004) 832–842.

[27] R. Ariew, Ockham's razor: A historical and philosophical analysis of Ockham's principle of parsimony.

[28] V. Basto-Fernandes, I. Yevseyeva, J. R. Méndez, J. Zhao, F. Fdez-Riverola, M. T.M. Emmerich, A SPAM filtering multi-objective optimization study covering parsimony maximization and three-way classification, Applied Soft Computing (2016) 111–123.

[29] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005, Edinburgh, UK, 2-4 September 2005), Vol. 1, IEEE Press, 2005, pp. 443–450.

[30] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland (2001).

[31] J. Zhao, V. Basto-Fernandes, L. Jiao, I. Yevseyeva, A. Maulana, R. Li, T. Bäck, M. Emmerich, Multiobjective optimization of classifiers by means of 3-D convex hull based evolutionary algorithms, arXiv preprint arXiv:1412.5710.

[32] S. Yitzhaki, Relative deprivation and the gini coefficient, Quarterly Journal of Economics 93 (2) (1979) 321–24.

[33] I. Zelinka, A survey on evolutionary algorithms dynamics and its complexity mutual relations, past, present and future, Swarm and Evolutionary Computation 25 (2015) 2–14.

[34] J. Moore, R. Chapman, Application of particle swarm to multiobjective optimization, in: International Conference on Computer Science and Software Engineering, 2003.

[35] M. R. Sierra, C. A. Coello Coello, Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance, in: in Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization, EMO, 2005, pp. 505–519.

[36] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, SMPSO: A new PSO-based metaheuristic for multi-objective optimization, in: Computational intelligence in multi-criteria decision-making, 2009. IEEE symposium on, 2009, pp. 66–73.

[37] C. Lücken, B. Barán, C. Brizuela, A survey on multi-objective evolutionary algorithms for many-objective problems, Computational Optimization and Applications 58 (3) (2014) 707–756.

[38] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601.

[39] S. Z. Zhao, P. N. Suganthan, Q. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Transactions on Evolutionary Computation 16 (3) (2012) 442–446.

[40] J. O'Rourke, Computational Geometry in C, Cambridge University Press, 1998.

[41] D. R. Chand, S. S. Kapur, An algorithm for convex polytopes, Journal of the Association for Computing Machinery 17 (1) (1970) 78–86.

[42] G. S. Brodal, R. Jacob, Dynamic planar convex hull, in: The 43rd Annual IEEE Symposium on Proceeding of Foundations of Computer Science, 2002, pp. 617–626.

[43] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software (TOMS) 22 (4) (1996) 469–483.

[44] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. M. A. D. Heide, H. Rohnert, R. E. Tarjan, Dynamic perfect hashing: upper and lower bounds, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 1988, pp. 524–531.

[45] F. P. Preparata, S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, Communications of the ACM 20 (2) (1977) 87–93.

[46] K. L. Clarkson, P. W. Shor, Applications of random sampling in computational geometry, II, Discrete and Computational Geometry 4 (5) (1989) 387–421.

[47] K. Clarkson, Kenneth L.and Mehlhorn, R. Seidel, Four results on randomized incremental constructions, Computational Geometry: Theory and Applications 3 (1993) 185–212. doi:10.1016/0925-7721(93)90009-U.

[48] G. Rote, K. Buchin, K. Bringmann, S. Cabello, M. Emmerich, Selecting K points that maximize the convex hull volume, in: The 19th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG3 2016), 2016, pp. 58–60.

[49] A. Ghosh, S. Tsutsui, H. Tanaka, Individual aging in genetic algorithms, in: Proceedings of Australian and New Zealand Conference on Intelligent Information Systems, 1996, pp. 276–279.

[50] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM Computing Surveys 45 (3) (2013) 533–545.

[51] W. N. Chen, J. Zhang, Y. Lin, N. Chen, Z. H. Zhan, S. H. Chung, Y. Li, Y. H. Shi, Particle swarm optimization

with an aging leader and challengers, IEEE Transactions on Evolutionary Computation 17 (2) (2013) 241–258.

[52] J. J. Durillo, A. J. Nebro, jMetal: A Java framework for multi-objective optimization, Advances in Engineering Software 42 (2011) 760–771.

[53] A. J. Nebro, J. J. Durillo, M. Vergne, Redesigning the jMetal multi-objective optimization framework, in: Genetic and Evolutionary Computation Conference (GECCO), 2015, pp. 1093–1100.

[54] H. Wang, Y. Jin, X. Yao, Diversity assessment in many-objective optimization, IEEE Transactions on Cybernetics (99) (2016) 1–13. `doi:10.1109/TCYB.2016.2550502`.

[55] R. Seidel, Convex hull computations, CRC Press, Inc., 1997.