Applied Soft Computing 67 (2018) 322–336

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

3D fast convex-hull-based evolutionary multiobjective optimization algorithm

Jiaqi Zhao[a,*], 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, Leicester LE1 9BH, UK
[f] Multicriteria Optimization, Design, and Analytics Group, LIACS, Leiden University, Niels Bohrweg 1, 2333-CA Leiden, The Netherlands

ARTICLE INFO

Article history:
Received 8 August 2017
Received in revised form 19 February 2018
Accepted 2 March 2018
Available online 11 March 2018

Keywords:
Convex hull
Area under ROC
Indicator-based evolutionary algorithm
Multiobjective optimization
ROC analysis

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, RVEA, etc., 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 300 without reducing the performance of the method. Besides, the proposed algorithm is applied for neural networks pruning, and several UCI datasets are used to test the performance.

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 [7–11]. 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], includ-

* Corresponding author.
E-mail addresses: jiaqizhao88@126.com (J. Zhao), xiasx@cumt.edu.cn (S. Xia).

https://doi.org/10.1016/j.asoc.2018.03.005
1568-4946/

ing 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 Algorithm (AG-EMOA) [25]. However, these methods do not consider 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 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. *CCR* is defined to describe the complexity of a classifier: $CCR \triangleq \emptyset$, which for instance can be measured by the number of rules used by the classifier and it determines the average cost (in time) a classifier requires [29]. 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. In practice, to find a classifier having better performance than that of classifiers in concave parts there is no need to linearly combine two classifiers on the ADCH, as we can select a classifier having good performance on the ADCH by using the iso-performance theory [1]. Experimental results in [9] show that 3DCH-EMOA outperforms NSGA-II [20], GDE3 (the third evolution step of Generalized Differential Evolution) [30], SPEA2 (Strength Pareto Evolutionary Algorithm 2) [31], MOEA/D [21] and SMS-EMOA [23] on the volume above surface (*VAS*) [32] performance indicator and in the *Gini* coefficient [33,9] 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.

3DCH-EMOA has good performance on both benchmark functions and many real-world classification problems. However, 3DCH-EMOA performs worse than several compared methods in terms of computational time, which 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 to fast the implementation of 3DCH-EMOA 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. More recently, several studies focused on solving many-objective optimization problems, i.e., problems having four or more objectives [34]. Generally, most of these many-optimization algorithms have better performance than multiobjective optimization algorithms while dealing with tri-objective optimization problems, as many-optimization algorithms have taken the complexity distribution of solutions in high dimensional into consideration. In the experimental section, several state-of-the-art many-objective optimization algorithms are applied to solve multi-objective ADCH maximization problems, including the two-archive algorithm (Two_Arch2) [35] which focuses on convergence and diversity separately, the decomposition based algorithms such as NSGA-III [36], the evolutionary algorithms based on both dominance and decomposition (MOEA/DD) [37], and the reference vector guided evolutionary algorithm (RVEA) [38], an indicator based evolutionary algorithm with reference point adaptation (AR-MOEA) [39], and a multi-objective particle swarm optimization algorithm based on decomposition (MPSO/D) [40].

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 and neural networks pruning problems. 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).

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

where $\mathbf{x}$ represents the parameters for a classifier to be optimized, and f_1, f_2, f_3 represent *FPR* (false positive rate), *FNR* (false negative rate) and *CCR* (classifier complexity ratio) [9], as defined by Eq. (2).

$$\begin{cases} f_1(\mathbf{x}) \triangleq FPR \\ f_2(\mathbf{x}) \triangleq FNR \\ f_3(\mathbf{x}) \triangleq CCR \end{cases} \tag{2}$$

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 [41–43]. The 3D convex hull CH of a finite set $A \subset \mathbb{R}^3$ is given by Eq. (3),

$$CH(A) \triangleq \left\{ x : x = \sum_{i=1}^{|A|} \mathbf{a}_i \lambda_i, \sum \lambda_i = 1, 0 \le \lambda_i \le 1 \right\}, \tag{3}$$

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 [44]. 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, we can obtain its facets, edges and vertices.

Several convex hull construction algorithms have been developed in the computational geometry community [41,45]. The gift-wrapping algorithm presented in [42] 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 [46], however, it is difficult to implement [41]. The randomized incremental convex hull algorithm was proposed in [47], 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 [48]. 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 [44]. 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) $R \subset \mathbb{R}^3$. To select the set of reference points for a new or real-world application we should analyze the distribution of solutions in advance. Any effective solutions can construct a convex hull with the reference point together. 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. (4).

$$FS(P) \triangleq \{p : p \in CH(P \cup R), p \in P\} \tag{4}$$

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. (5).

$$\text{non-}FS(P) \triangleq PFS(P) \tag{5}$$

The volume above DET convex hull surface (*VAS*) is defined as the volume of convex hull *CH*, and is denoted by Eq. (6).

$$\begin{aligned} VAS(P) &\triangleq Volume(CH(P \cup R)) \\ &= Volume(CH(FS(P) \cup R)) \end{aligned} \tag{6}$$

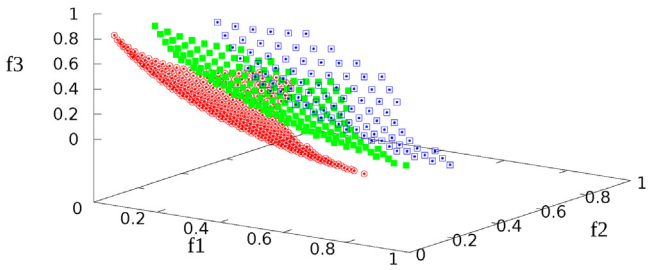


Fig. 1. Ranking of population into three different levels with 3D convex-hull-based ranking without redundancy scheme in 3DCH-EMOA; the individuals in different levels are marked in different colors. The first level of solutions are marked in red, the second level of solutions are marked in green and the third level solutions are marked in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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

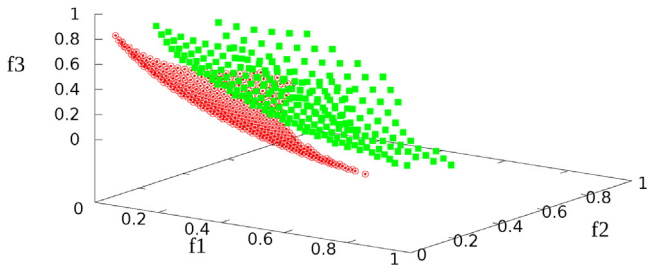


Fig. 2. Ranking of the population into two different levels with 3DICH-based sorting in 3DFCH-EMOA. The individuals in different levels are marked in different colors. The first level of solutions is marked in red and the rest of solutions is marked in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 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 irrelevant 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.

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

Require:
$P \neq \emptyset$, ,
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.
1: *CH* ⟵ building convex hull with points $P \cup R$.
2: *FSset* ⟵ *FS(P)*
3: non-*FSset* ⟵ *P FSset*
4: RS_0 ⟵ *FSset* , .
5: **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 and a solution with bad performance will be deleted 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 to maximize the convex hull volume is a NP complete problem [49]. This is why the $(\mu + 1)$ strategy is favored over a more general $(\mu + \lambda)(\lambda > 1)$ 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 [50,51]. 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 [52].

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 $\neq \emptyset$.
Ensure: *AgingQueue* and non-*FS*set are updated.
1: **if** non-*FS*set $\neq \emptyset$ **then**
2: q ⟵ the first element in *AgingQueue*.
3: Remove the first element in *AgingQueue*.
4: non-*FS*set ⟵ non-*FS*set q.
5: **end if**
6: **return** *AgingQueue*, non-*FS*set

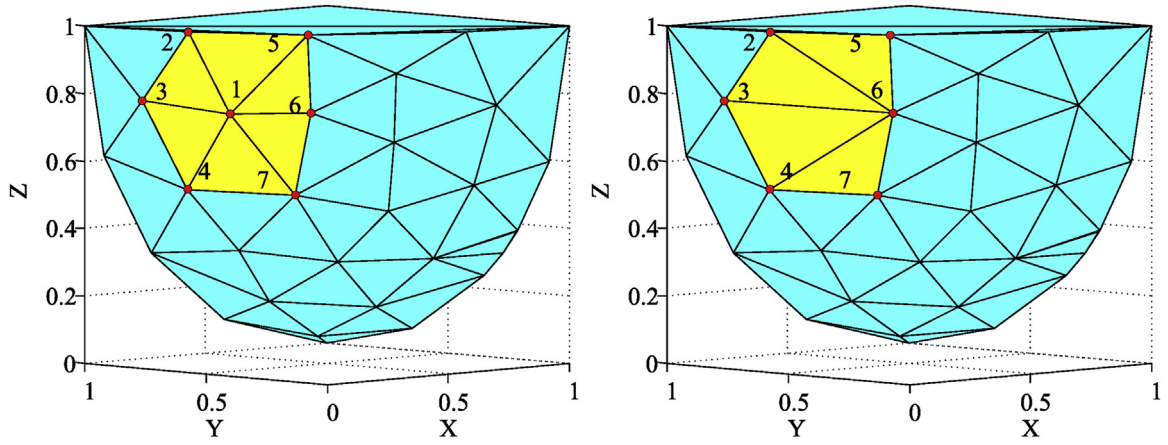
3.3. Fast calculation of VAS contribution

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 *VAS* contribution of each solution should be calculated.

The theory of random incremental convex hulls [48] 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. (7).

$$RV(p) \triangleq \{q : p \in F_i, q \in F_i, p \neq q, F_i \in CH\} \tag{7}$$



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

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

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.

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.
$RV \longleftarrow \emptyset$
for $i \longleftarrow 1 : N_F$ **do**
  if $q \in F_i$ **then**
    for all $p \in F_i$ **do**
      if $p \neq q$ and $p \notin RV$ **then**
$RV \longleftarrow RV \cup \{p\}$
      end if
    end for
  end if
end for
return *RV*

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 [41].

The importance of individuals in the convex hull is evaluated by their contribution to *VAS*, which is denoted as ΔVAS. In [32], 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 initial convex hull that includes p. The contribution of solution p is calculated by Eq. (8).

$$\Delta VAS(p) = VAS(P) - VAS(P\{p\}). \tag{8}$$

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 vertices and a reference vertex r. The reference vertex r acts as a vertex of the partial convex hull with related vertices together. Generally, to select a reference point r we should analyze the distribution of solutions first. The fast way to compute the contribution of vertex p is described in Eq. (9).

$$\Delta VAS_f(p) = Volume(CH(RV(p) \cup \{p\} \cup \{r\})) - Volume(CH(RV(p) \cup \{r\})) \tag{9}$$

where r is the reference vertex (r is point (1, 1, 1) in the context of *VAS*). In the implementation of Eq. (9) 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 $O(\log\ n)$.

Algorithm 4. Fast ΔVAS *(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.
$RV \longleftarrow$ Finding related vertices(CH, q).
$VAS_0 \longleftarrow Volume(CH(RV \cup \{q\} \cup \{r\}))$.
$VAS_1 \longleftarrow Volume(CH(RV \cup \{r\}))$.
$\Delta VAS \longleftarrow VAS_0 - VAS_1$.
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. The $(\mu + 1)$ selection strategy is employed in this algorithm. According to this steady state strategy only a 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*, i.e., q is out of *CH*, it will be added to *CH* as a new vertex and be stored in the *FS*set. If q is inside *CH*, it will be stored in the non-*FS*set.

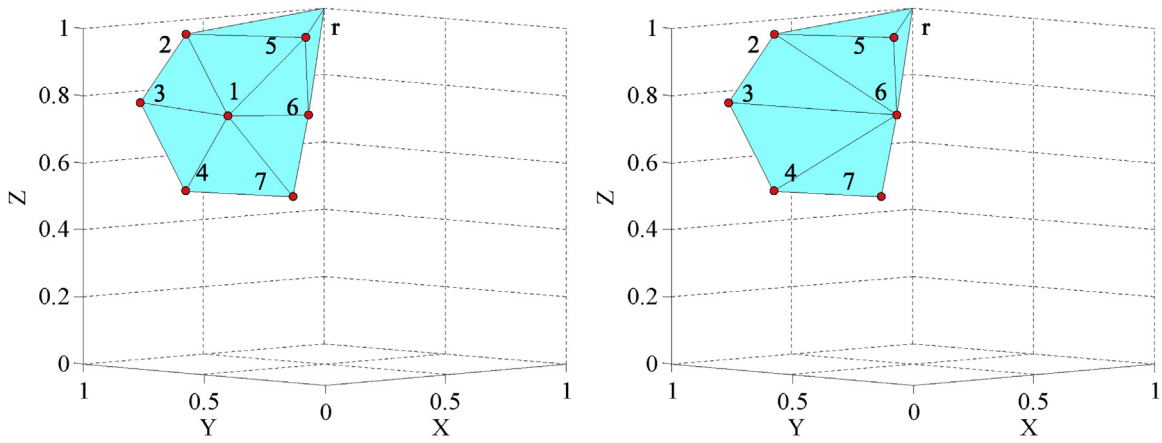


Fig. 4. An example of calculating the *VAS* contribution of each vertex to the convex hull in 3DFCH-EMOA.

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 of the convex hull structure 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 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 $O(\log\ n)$.

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

Require:
CH is the convex hull,
*FS*set $\neq \emptyset$,
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*.
*FS*set ⟵ *FS*set ∪{q}
for all $p \in$ *FS*set **do**
  if p is not a vertex of *CH* **then**
    Add p to the end of *AgingQueue*
    non-*FS*set ⟵ non-*FS*set ∪{p}
    *FS*set ⟵ *FS*set {p}
  end if
end for
RV ⟵ Finding related vertices(*CH*,q).
CH.q.contribution ⟵Fast Δ*VAS(CH,q,r)*.
for all $p \in$ *RV* **do**
  CH.p.contribution ⟵Fast Δ*VAS(CH,p,r)*.
end for
return *CH*, *FS*set, non-*FS*set and *AgingQueue*.

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 $O(n\ \log\ n)$.

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

Require:
CH is the convex hull,
*FS*set $\neq \emptyset$,
q is a solution that will be deleted.
Ensure: The contribution to *CH* and *FS*set are updated.
*FS*set ⟵ *FS*set {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 $p \in$ *RV* **do**
  CH.p.contribution ⟵Fast Δ*VAS(CH, p, r)* computation.
end for
return *CH, FS*set

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. Thus, their computational time complexity can be described by considering one iteration of the entire algorithm. In this section, we consider the population to be of size n. In 3DCH-EMOA, the computational time complexity of variation operation for generating a new offspring is equal to $O(d)$, where d is the length of decision variables. 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 $O(n^2\ \log\ n)$.

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 ⟵ init()
RS, *CH* ⟵ 3DICH-based sorting (, R)
*FS*set ⟵ RS_0, non-*FS*set ⟵ 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*
$t \longleftarrow n$
while $t <$ *MEs* **do**

10: $t \longleftarrow t+1$, Mutate (Recombine (FSset non-FSset))
11: **if** q_t is inside the convex hull *(CH)* **then**
12: non-*FS*set $\longleftarrow$ non-*FS*set $\cup\{q_t\}$
13: Add q_t to the end of *AgingQueue*
14: **else**
15: *CH*, *FS*set, non-*FS*set, *AgingQueue* $\longleftarrow$ Adding a point to *CH (CH, FS*set, non-*FS*set, q_t)
16: **end if**
17: **if** *AgingQueue* $\neq \emptyset$ **then**
18: *AgingQueue*, non-*FS*set $\longleftarrow$ Age-based selection (*AgingQueue*, non-*FS*set)
19: **else**
20: Finding the least contribution vertex p
21: *CH*, *FS*set $\longleftarrow$ Deleting a point from *CH (CH, p*)
22: **end if**
23: **end while**
24: **return** *FS*set

In 3DFCH-EMOA, the computational time complexity of variation operation for generating a new offspring is equal to $O(d)$, where d is the length of the decision variables. 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 $O(\log\ n)$ 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, including Two_Arch2 [35], NSGA-III [36], MOEA/DD [37], RVEA [38], AR-MOEA [39], MPSO/D [40] and 3DCH-EMOA [9]. ZED test functions were designed in [32] 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.

All of these experiments were performed in PlatEMO [53], which is a MATLAB platform for evolutionary multi-objective optimization algorithms. All experiments were run on a desktop PC with an E5-2630 2.3GHz processor and 16GB memory under Ubuntu 16.04LTS. 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 Two_Arch2, NSGA-III, MOEA/DD, RVEA, AR-MOEA, and MPSO/D 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 ZED1 test function.
- 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], inverted generational distance (*IGD*) [53], 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 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 *IGD* metric is able to measure both diversity and convergence of solutions obtained by EMOAs, and a smaller *IGD* value indicates a better performance.
- 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.

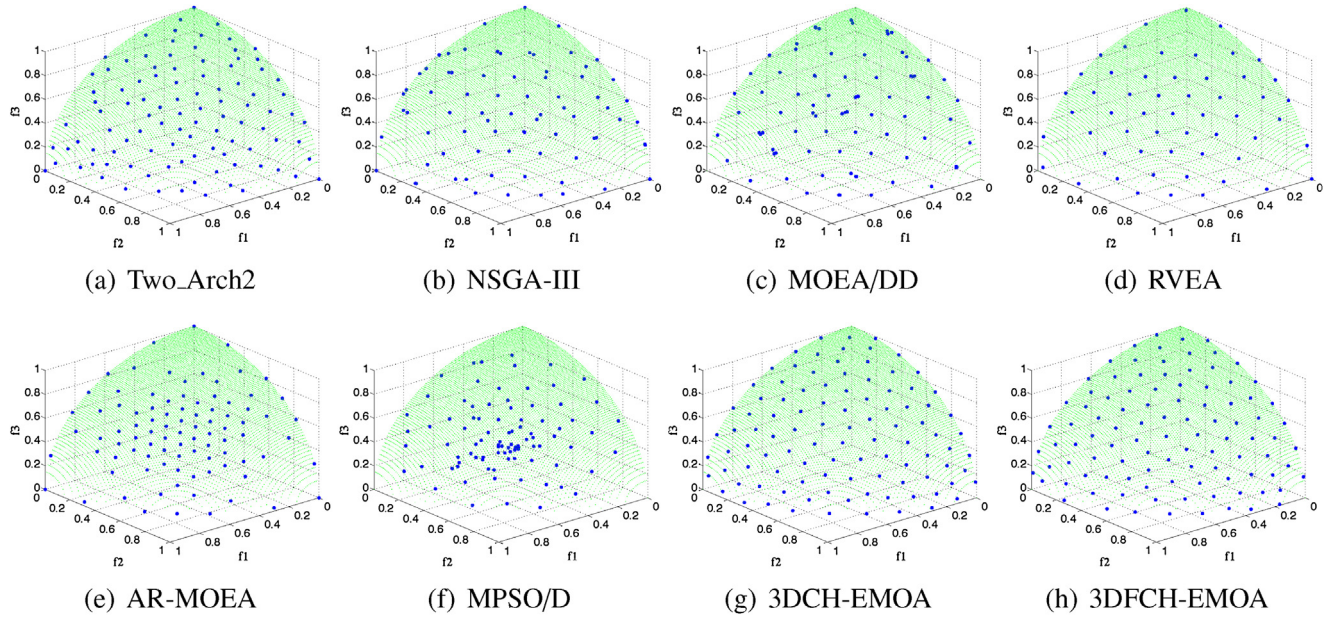


Fig. 5. Experimental results of the frontal solutions (for single run) obtained by each algorithm on ZED1 test function in $f_1 - f_2 - f_3$ space.

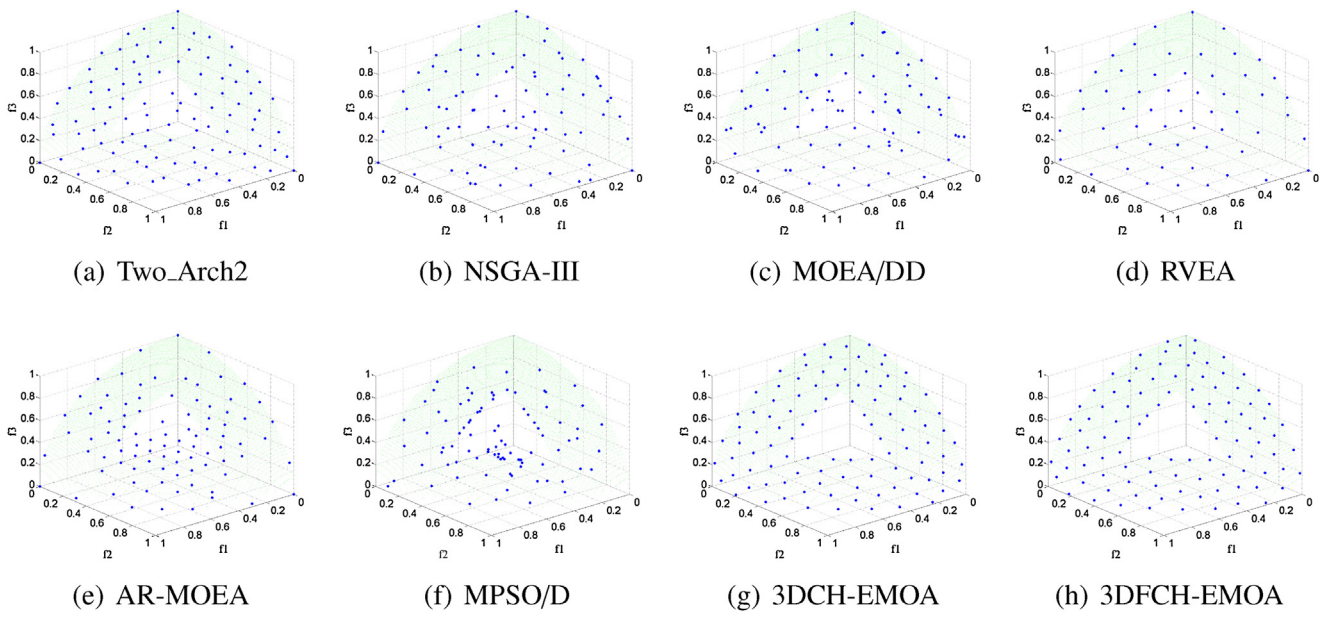


Fig. 6. Experimental results of the frontal solutions (for single run) obtained by each algorithm on ZED2 test function in $f_1 - f_2 - f_3$ space.

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.

4.1.1. Parameter settings

All algorithms use a maximum of 30,000 function evaluations, and the population size is set to 100 for all algorithms. The remaining parameters are set as the default suggestion by PlatEMO for all algorithms.

4.1.2. Experimental results and discussions

Firstly, the frontal solutions (for one run) of ZED and ZEJD are shown in Figs. 5–10. In these figures, the true frontal solutions are marked in light colour, and the solutions obtained by each algorithm are marked in deep colour.

The solutions obtained by each algorithm of ZED1 are shown in Fig. 5. By analyzing the frontal solutions of the ZED1 function obtained by the algorithms we can make some conclusions: (1) All algorithms can convergence to the true frontal solutions; (2) NSGA-III, MOEA/DD, and MPSO/D perform worse than others on the uniformity; (3) The solutions obtained by AR-MOEA are gathered in the central area of the frontal solutions; (4) Two_Arch2 performs better on the uniformity metric than others except for 3DCH-EMOA and 3DFCH-EMOA, however, it can not obtain solutions on the boundary of frontal solutions; (5) RVEA has good ability on convergence and uniformity, however, the number of solutions obtained by it is less than other algorithms; (6) 3DCH-EMOA and 3DFCH-EMOA perform better than other algorithms on uniformity.

The solutions obtained by each algorithm of ZED2 are shown in Fig. 6. The surface of frontal solutions of ZED2 is not continuous, and there is an area of concave on the frontal solutions, which is designed to test whether the algorithms can avoid the dent areas [32]. By analyzing the frontal solutions of ZED2 function obtained by the algorithms we can make some conclusions: (1) The distribution of solutions obtained by each algorithm is similar with ZED1 test problems; (2) MPSO/D perform worse than others on the uniformity; (3) The solutions obtained by AR-MOEA are gathered in the concave area of the frontal solutions; (4) Most of the algorithms except for 3DCH-EMOA and 3DFCH-EMOA can find solutions in the concave area, which makes no sense for ADCH maximization problems; (5) 3DCH-EMOA and 3DFCH-EMOA can avoid sampling solutions in the dent area, which is better because these regions contain only redundant solutions since the goal of ADCH maximization is to find solutions that lie on the convex hull.

The solutions obtained by each algorithm of ZED3 are shown in Fig. 7. The surface of frontal solutions of ZED3 is continuous, and there is an area of concave on the frontal solutions. By analyzing the frontal solutions of ZED3 function obtained by the algorithms we can make some conclusions: (1) MPSO/D perform worse than others on the uniformity; (2) The solutions obtained by AR-MOEA are gathered in the concave area of the frontal solutions; (3) Most of the algorithms except for 3DCH-EMOA and 3DFCH-EMOA can find solutions in the concave area.

The solutions obtained by each algorithm of ZEJD1 are shown in Fig. 8. By analyzing the frontal solutions of ZEJD1 function obtained by the algorithms we can make some conclusions: (1) All of the algorithms can almost convergence to the true frontal solutions; (2) Two_Arch2 and RVEA perform worse than other algorithms, as some solutions obtained by it are not on the frontal solutions; (3) NSGA-III, MOEA/DD, AR-MOEA, MPSO/D can obtain solutions with good uniformity, however, as it is pointed in [54], a solution set with good uniformity does not necessarily mean that it also has good diversity; (4) 3DCH-EMOA and 3DFCH-EMOA can obtain solutions not only with good uniformity but also with good convergence.

The solutions obtained by each algorithm of ZEJD2 are shown in Fig. 9. The surface of frontal solutions of ZEJD2 is not continuous, and there is an area of concave on the frontal solutions [9]. By analyzing the frontal solutions of ZEJD2 function obtained by the algorithms we can make some conclusions: (1) Two_Arch2 performs worse than others on the convergence and uniformity; (2) 3DCH-EMOA and 3DFCH-EMOA can avoid sampling solutions in the dent area.

The solutions obtained by each algorithm of ZEJD3 are shown in Fig. 10. The surface of frontal solutions of ZEJD3 is continuous, and there is an area of concave on the frontal solutions [9]. By analyzing the frontal solutions of ZEJD3 function obtained by the algorithms we can make some conclusions: (1) RVEA performs worse than others on the convergence and uniformity; (2) 3DCH-EMOA and 3DFCH-EMOA can avoid sampling solutions in the dent area.

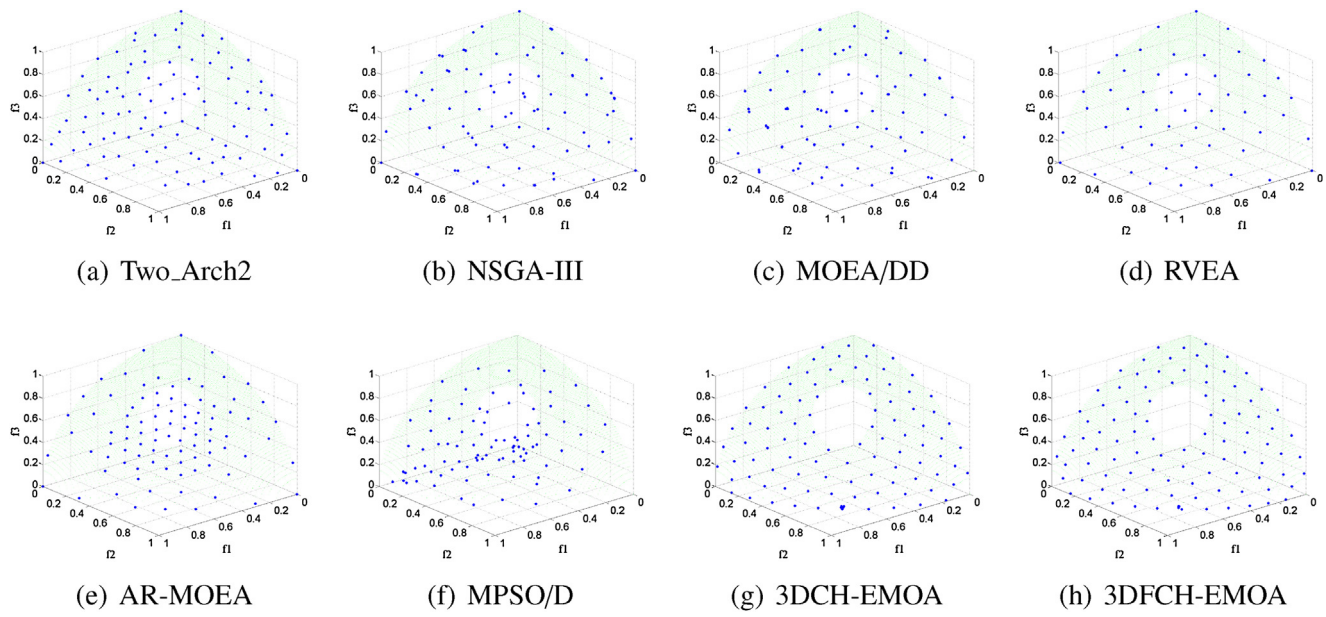


Fig. 7. Experimental results of the frontal solutions (for single run) obtained by each algorithm on ZED3 test function in $f_1 - f_2 - f_3$ space.

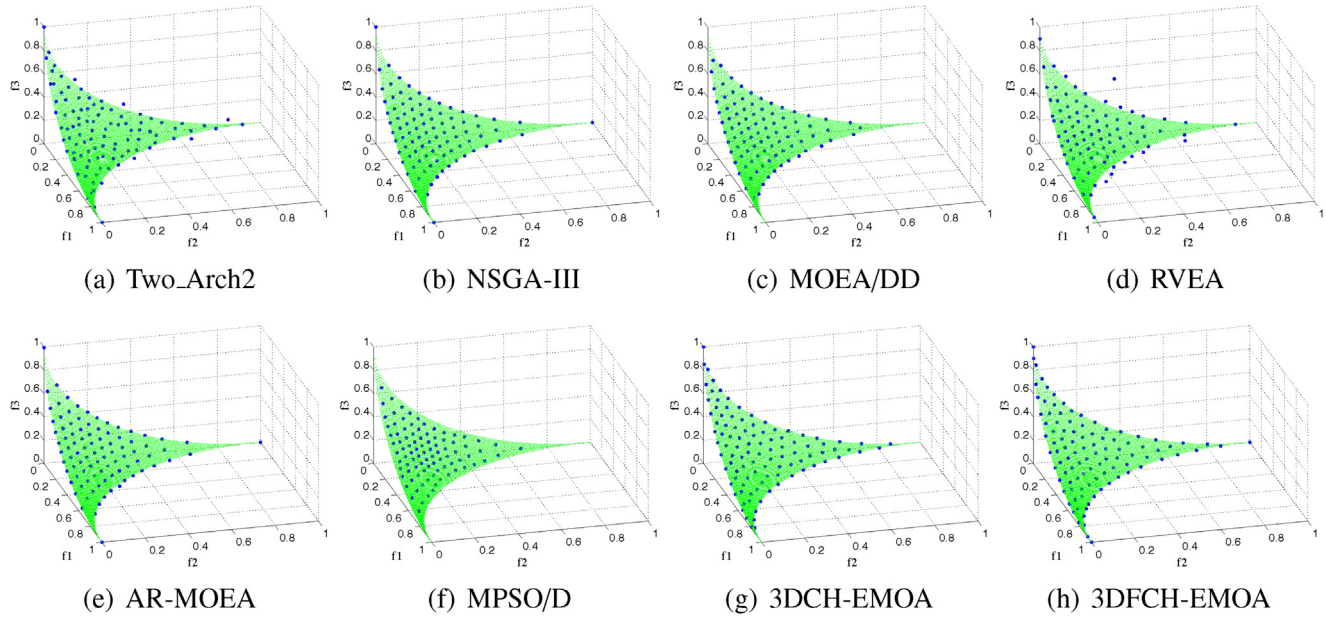


Fig. 8. Experimental results of the frontal solutions(for single run) obtained by each algorithm on ZEJD1 test function in $f_1 - f_2 - f_3$ space.

By comparing the frontal solutions of ZED and ZEJD test functions we can conclude that 3DFCH-EMOA obtains results as good as 3DCH-EMOA. Besides, only 3DCH-EMOA and 3DFCH-EMOA can omit the solutions in concave areas, 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 of several metrics are listed in the following tables. In these tables the best results obtained are marked in light grey and the second best results are marked in dark grey. 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].

By comparing the results on the table, 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. Two_Arch2, NSGA-III and AR-MOEA outperforms other algorithms on the metric of *VAS* except of 3DCH-EMOA and 3DFCH-EMOA. MPSO/D performs the worst over all these methods, which is similar with the conclusion made by comparing the frontal solutions above. This confirms that 3DFCH-EMOA has successfully inherited the good performance of 3DCH-EMOA.

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 all of ZED test problems. RVEA performs better than the other algorithms except of 3DCH-EMOA and 3DFCH-EMOA on ZED test

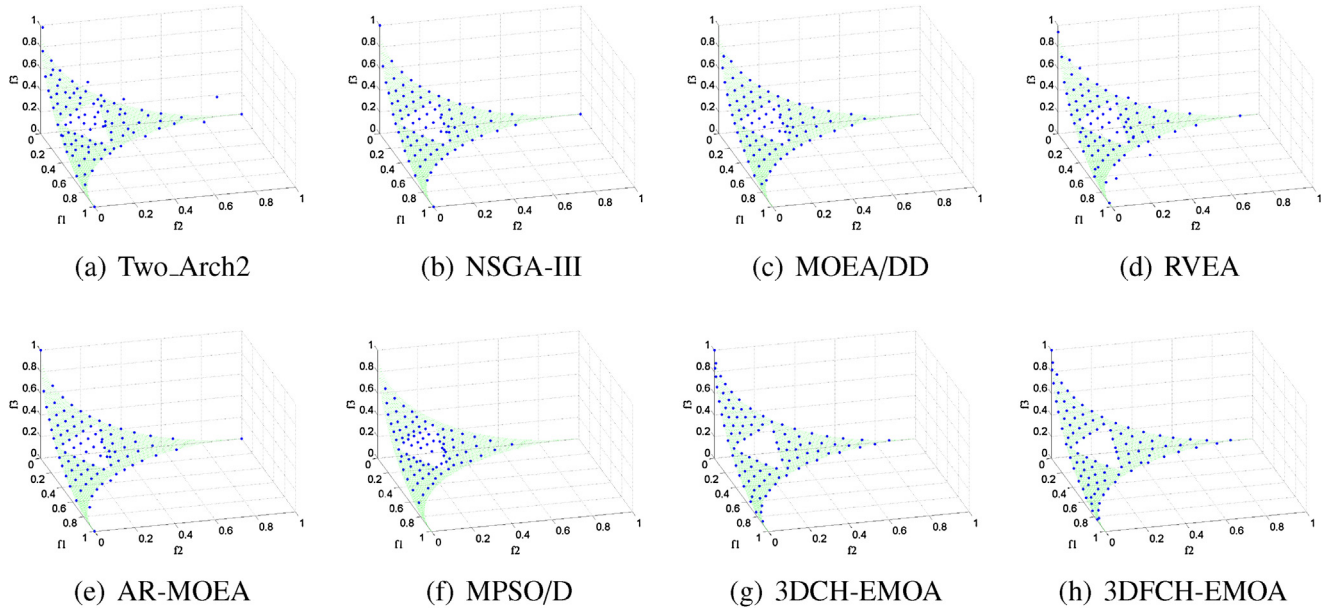


Fig. 9. Experimental results of the frontal solutions (for single run) obtained by algorithms on ZEJD2 test function in $f_1 - f_2 - f_3$ space.

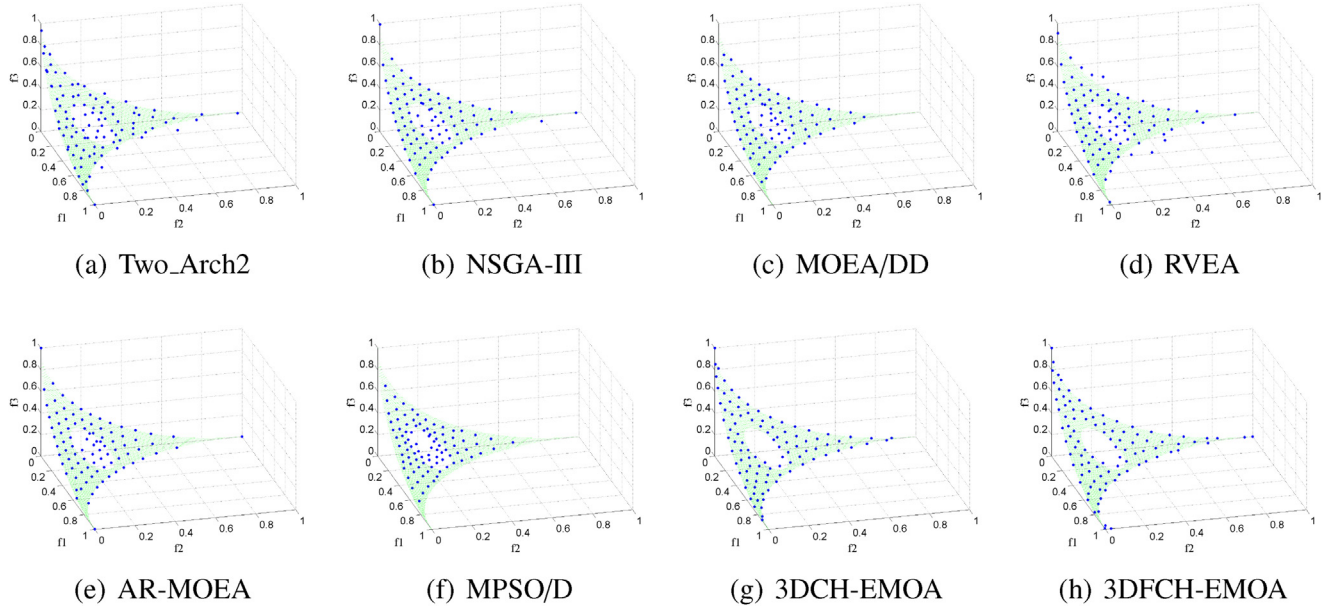


Fig. 10. Experimental results of the frontal solutions (for single run) obtained by each algorithm on ZEJD3 test function in $f_1 - f_2 - f_3$ space.

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

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
ZED1	$3.50e-01_{9.22e-04}$	$3.47e-01_{6.40e-04}$	$3.48e-01_{2.36e-04}$	$3.44e-01_{9.03e-04}$	$3.47e-01_{6.70e-05}$	$3.29e-01_{1.88e-03}$	$3.53e-01_{2.17e-05}$	$3.53e-01_{1.18e-04}$
ZED2	$3.48e-01_{9.68e-04}$	$3.43e-01_{6.27e-04}$	$3.43e-01_{5.27e-04}$	$3.39e-01_{1.36e-03}$	$3.44e-01_{2.79e-04}$	$3.27e-01_{2.07e-03}$	$3.52e-01_{1.88e-05}$	$3.52e-01_{5.94e-05}$
ZED3	$3.40e-01_{1.23e-03}$	$3.39e-01_{3.86e-03}$	$3.40e-01_{1.91e-03}$	$3.29e-01_{2.21e-03}$	$3.31e-01_{1.03e-04}$	$3.28e-01_{1.68e-03}$	$3.42e-01_{2.09e-03}$	$3.42e-01_{1.55e-03}$
ZEJD1	$4.62e-01_{4.21e-03}$	$4.64e-01_{8.14e-04}$	$4.16e-01_{2.66e-03}$	$4.50e-01_{6.58e-03}$	$4.64e-01_{8.96e-04}$	$3.91e-01_{6.83e-04}$	$4.65e-01_{1.29e-06}$	$4.65e-01_{2.68e-03}$
ZEJD2	$4.61e-01_{4.84e-03}$	$4.64e-01_{8.69e-04}$	$4.18e-01_{3.83e-03}$	$4.49e-01_{6.16e-03}$	$4.64e-01_{2.66e-04}$	$3.91e-01_{7.85e-04}$	$4.65e-01_{1.25e-06}$	$4.65e-01_{2.93e-04}$
ZEJD3	$4.62e-01_{3.92e-03}$	$4.63e-01_{1.49e-03}$	$4.16e-01_{5.58e-03}$	$4.49e-01_{7.67e-03}$	$4.64e-01_{6.23e-04}$	$3.91e-01_{8.26e-04}$	$4.65e-01_{1.64e-06}$	$4.65e-01_{2.78e-04}$

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

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
ZED1	$1.14e-01_{1.65e-02}$	$4.35e-01_{4.53e-02}$	$5.45e-01_{4.05e-02}$	$9.82e-02_{5.85e-03}$	$1.99e-01_{4.19e-03}$	$2.14e-01_{2.45e-02}$	$4.54e-02_{4.97e-03}$	$4.49e-02_{8.45e-03}$
ZED2	$1.19e-01_{1.41e-02}$	$4.19e-01_{4.30e-02}$	$5.51e-01_{5.44e-02}$	$9.48e-02_{1.19e-02}$	$1.97e-01_{7.82e-03}$	$2.71e-01_{2.11e-02}$	$5.51e-02_{6.01e-03}$	$5.49e-02_{8.08e-03}$
ZED3	$1.21e-01_{1.19e-02}$	$4.31e-01_{4.09e-02}$	$5.48e-01_{3.09e-02}$	$9.89e-02_{7.60e-03}$	$2.02e-01_{5.06e-03}$	$2.37e-01_{2.39e-02}$	$8.55e-02_{8.01e-03}$	$8.77e-02_{9.20e-03}$
ZEJD1	$1.66e-01_{1.62e-02}$	$1.47e-01_{3.59e-03}$	$3.86e-02_{2.76e-03}$	$1.39e-01_{1.49e-02}$	$2.30e-01_{5.14e-02}$	$1.13e-01_{3.93e-03}$	$7.12e-02_{8.56e-03}$	$8.12e-02_{9.64e-03}$
ZEJD2	$1.70e-01_{1.83e-02}$	$1.51e-01_{6.45e-03}$	$7.41e-02_{2.55e-02}$	$1.41e-01_{1.65e-02}$	$2.43e-01_{4.31e-02}$	$1.75e-01_{5.83e-03}$	$7.97e-02_{8.59e-03}$	$7.23e-02_{7.47e-03}$
ZEJD3	$1.69e-01_{1.53e-02}$	$1.60e-01_{9.61e-03}$	$7.17e-02_{2.39e-02}$	$1.38e-01_{1.51e-02}$	$2.41e-01_{5.24e-02}$	$1.79e-01_{4.63e-03}$	$1.26e-01_{1.15e-02}$	$1.32e-01_{1.90e-02}$

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

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
ZED1	$5.35e-02_{1.09e-03}$	$6.88e-02_{1.93e-03}$	$6.68e-02_{9.79e-04}$	$7.62e-02_{1.07e-03}$	$6.75e-02_{3.93e-04}$	$8.12e-02_{3.67e-03}$	$5.07e-02_{3.72e-04}$	$5.09e-02_{3.73e-04}$
ZED2	$5.28e-02_{1.18e-03}$	$6.86e-02_{1.93e-03}$	$6.73e-02_{1.12e-03}$	$7.74e-02_{1.65e-03}$	$6.58e-02_{5.72e-04}$	$8.54e-02_{2.93e-03}$	$5.80e-02_{3.15e-04}$	$5.82e-02_{6.03e-04}$
ZED3	$5.33e-02_{8.39e-04}$	$6.90e-02_{1.37e-03}$	$6.67e-02_{1.20e-03}$	$7.64e-02_{8.32e-04}$	$6.72e-02_{3.72e-04}$	$8.28e-02_{2.47e-03}$	$5.56e-02_{3.91e-04}$	$5.58e-02_{3.22e-04}$
ZEJD1	$2.88e-02_{5.01e-04}$	$3.21e-02_{1.81e-04}$	$3.04e-02_{1.52e-04}$	$3.26e-02_{6.82e-04}$	$3.21e-02_{1.71e-04}$	$3.14e-02_{1.10e-04}$	$2.70e-02_{1.34e-04}$	$2.75e-02_{2.56e-04}$
ZEJD2	$2.87e-02_{5.02e-04}$	$3.23e-02_{1.70e-04}$	$3.06e-02_{4.72e-04}$	$3.30e-02_{8.19e-04}$	$3.22e-02_{1.13e-04}$	$3.24e-02_{3.11e-04}$	$3.19e-02_{1.11e-04}$	$3.20e-02_{2.16e-04}$
ZEJD3	$2.99e-02_{5.71e-04}$	$3.31e-02_{2.22e-04}$	$3.16e-02_{4.05e-04}$	$3.34e-02_{7.89e-04}$	$3.31e-02_{1.61e-04}$	$3.33e-02_{1.33e-04}$	$3.16e-02_{1.46e-04}$	$3.20e-02_{2.72e-04}$

Table 4
Mean and standard deviation of *PD* on ZED and ZEJD test problems.

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
ZED1	$2.48e+05_{1.22e+04}$	$1.15e+05_{9.09e+03}$	$1.54e+05_{9.16e+03}$	$1.22e+05_{7.62e+03}$	$1.72e+05_{6.36e+03}$	$1.91e+05_{1.06e+04}$	$1.89e+05_{9.14e+03}$	$1.89e+05_{1.12e+04}$
ZED2	$2.44e+05_{1.26e+04}$	$1.17e+05_{1.60e+04}$	$1.58e+05_{1.28e+04}$	$1.26e+05_{9.53e+03}$	$1.66e+05_{9.53e+03}$	$1.83e+05_{8.00e+03}$	$1.70e+05_{6.60e+03}$	$1.67e+05_{9.46e+03}$
ZED3	$2.49e+05_{9.94e+03}$	$1.22e+05_{1.07e+04}$	$1.56e+05_{1.09e+04}$	$1.22e+05_{7.02e+03}$	$1.69e+05_{8.39e+03}$	$1.96e+05_{1.00e+04}$	$1.84e+05_{1.01e+04}$	$1.86e+05_{1.09e+04}$
ZEJD1	$1.18e+05_{5.98e+03}$	$6.25e+04_{2.23e+03}$	$7.39e+04_{3.47e+03}$	$7.14e+04_{3.78e+03}$	$6.34e+04_{8.13e+03}$	$8.36e+04_{3.68e+03}$	$1.04e+05_{5.42e+03}$	$9.66e+04_{6.76e+03}$
ZEJD2	$1.19e+05_{5.61e+03}$	$6.41e+04_{3.27e+03}$	$8.62e+04_{5.65e+03}$	$7.54e+04_{3.84e+03}$	$6.55e+04_{6.85e+03}$	$8.40e+04_{3.50e+03}$	$8.80e+04_{4.85e+03}$	$8.50e+04_{6.36e+03}$
ZEJD3	$1.18e+05_{4.54e+03}$	$6.39e+04_{2.53e+03}$	$8.20e+04_{4.92e+03}$	$7.41e+04_{4.01e+03}$	$6.37e+04_{3.02e+03}$	$8.58e+04_{2.45e+03}$	$1.08e+05_{1.04e+04}$	$9.83e+04_{7.87e+03}$

Table 5
Mean and standard deviation of execution time (ms) on ZED and ZEJD test problems.

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
ZED1	$9.19e+04_{5.34e+03}$	$2.61e+05_{1.43e+04}$	$5.52e+04_{3.17e+03}$	$2.23e+03_{1.36e+02}$	$1.12e+06_{1.51e+05}$	$1.19e+04_{3.83e+02}$	$2.64e+06_{9.98e+04}$	$3.60e+05_{1.28e+04}$
ZED2	$9.23e+04_{5.39e+03}$	$2.62e+05_{1.53e+04}$	$5.62e+04_{4.12e+03}$	$2.21e+03_{1.37e+02}$	$1.12e+06_{1.79e+05}$	$1.21e+04_{4.15e+02}$	$2.63e+06_{1.02e+05}$	$3.48e+05_{1.24e+04}$
ZED3	$9.18e+04_{4.99e+03}$	$2.62e+05_{1.32e+04}$	$5.61e+04_{3.56e+03}$	$2.18e+03_{1.29e+02}$	$1.12e+06_{1.79e+05}$	$1.20e+04_{4.40e+02}$	$2.19e+06_{9.61e+04}$	$3.41e+05_{1.20e+04}$
ZEJD1	$8.36e+04_{5.00e+03}$	$2.22e+05_{1.37e+04}$	$5.54e+04_{4.65e+03}$	$2.53e+03_{1.65e+02}$	$9.91e+05_{1.44e+05}$	$1.52e+04_{6.06e+02}$	$2.03e+06_{6.79e+04}$	$3.03e+05_{1.25e+04}$
ZEJD2	$8.36e+04_{4.79e+03}$	$2.33e+05_{1.26e+04}$	$5.47e+04_{3.97e+03}$	$2.58e+03_{1.15e+02}$	$9.88e+05_{1.40e+05}$	$1.52e+04_{5.52e+02}$	$1.78e+06_{8.11e+04}$	$2.87e+05_{1.05e+04}$
ZEJD3	$8.32e+04_{4.89e+03}$	$2.39e+05_{1.32e+04}$	$5.54e+04_{4.45e+03}$	$2.59e+03_{1.08e+02}$	$9.89e+05_{1.34e+05}$	$1.50e+04_{4.57e+02}$	$1.74e+06_{7.00e+04}$	$2.86e+05_{1.05e+04}$

Table 6
Computational complexity of compared algorithms (with a population of n individuals of three objective optimization problems).

Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
$O(n^2)$	$O(n^2)$	–	$O(n^2)$	–	–	$O(n^2 \log n)$	$O(n \log n)$

Table 7
Wilcoxon sum-rank test on ZED and ZEJD test problems: each *x-y-z* in following table means 3DFCH-EMOA wins *x* times, losses *y* times and draws *z* times.

	Two_Arch2	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA
VAS	6-0-0	6-0-0	6-0-0	6-0-0	6-0-0	6-0-0	0-0-6
Gini	6-0-0	6-0-0	3-2-1	6-0-0	6-0-0	6-0-0	0-0-6
IGD	2-4-0	5-1-0	4-2-0	6-0-0	5-1-0	5-0-1	1-1-4
PD	0-6-0	6-0-0	5-0-1	6-0-0	5-0-1	2-2-2	1-1-4
Time	0-6-0	0-6-0	0-6-0	0-6-0	6-0-0	0-6-0	6-0-0

functions. While dealing with ZEJD test functions, MOEA/DD, 3DCH-EMOA and 3DFCH-EMOA performs better than the other algorithms.

The statistical results of *IGD* are shown in Table 3. From the table we can see that Two_Arch2, 3DCH-EMOA and 3DFCH-EMOA outperform the other algorithms for most of the test problems. Two_Arch2 is slightly better than 3DCH-EMOA and 3DFCH-EMOA for most of the test functions.

The statistical results of *PD* diversity metric are shown in Table 4. Two_Arch2 has the best diversity metric results; MPSO/D, 3DCH-EMOA and 3DFCH-EMOA have better performance than others on *PD* metric except for Two_Arch2. NSGA-III performs the worst over all these method on *PD* metric.

The statistical results on the execution times are shown in Table 5. RVEA has always the lowest execution time and MPSO/D performs better than the other algorithms except of SMPSO. 3DCH-EMOA cost the most time of all algorithms. 3DFCH-EMOA performs better than 3DCH-EMOA and AR-MOEA. 3DCH-EMOA uses more than 7 times as much computational time as 3DFCH-EMOA algorithm, that is to confirm that the new algorithm speeds up 3DCH-EMOA about more than 7 times with the population size 100.

The computational complexity of several algorithms are listed in Table 6; as the complexity of some algorithms was not mentioned in the proposed paper, we mark them as "–" in the table. From the table, we can see that 3DCH-EMOA has the highest computational complexity and 3DFCH-EMOA has the lowest computational complexity. The computational complexity represents the rate at the execution time increases with the population increases. Low computational complexity does not mean low execution time; 3DFCH-EMOA costs more time than Two_Arch2 in Table 5. The execution time of 3DCH-EMOA and 3DFCH-EMOA with different population sizes will be discussed in the next part.

A more comprehensive comparison between 3DFCH-EMOA and other EMOAs is presented in Table 7, which gives the Wilcoxon sum-rank test [8] results for them. It is very clear that 3DFCH-EMOA performs very well over most of these EMOAs on *VAS*, *Gini*, *IGD*, and *PD*.

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

We tested 3DFCH-EMOA and 3DCH-EMOA on ZED1 test function with different population sizes (100, 200, 300, 400, 500, 1000). For each mentioned parameter, 30 independent trials were run. All algorithms run for 30,000 function evaluations.

4.2.1. Experimental results and discussions

The results of *VAS* mean are listed in Table 8. By comparing the values of *VAS* we can see that 3DFCH-EMOA can obtain the same

Table 8
The mean of *VAS* on ZED1 test problem.

Population size	Compared methods	
	3DFCH-EMOA	3DCH-EMOA
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

Table 9
The mean of execution time (ms) on ZED1 test functions.

population size	compared methods	
	3DFCH-EMOA	3DCH-EMOA
100	$3.60e+05$	$2.64e+06$
200	$4.97e+05$	$7.90e+06$
300	$5.53e+05$	$1.93e+07$
400	$5.82e+05$	$3.63e+07$
500	$6.52e+05$	$4.03e+07$
1000	$1.10e+06$	$2.72e+08$

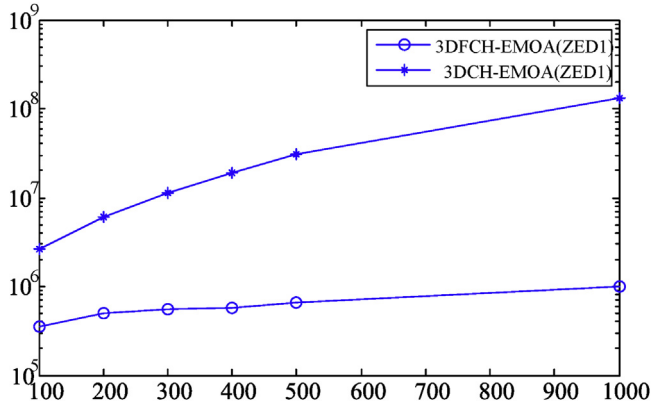


Fig. 11. Comparison of execution times of 3DFCH-EMOA and 3DCH-EMOA on ZED1 test function obtained by 30 independent trials with different population sizes.

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 ZED1 test function are listed in Table 9. Execution time analysis for several population sizes for ZED1 function is shown in Fig. 11. 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. In addition, we can find that the 3DFCH-EMOA can speed up 3DCH-EMOA for about 30 times with population size of 300.

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 ran 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. 12. 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

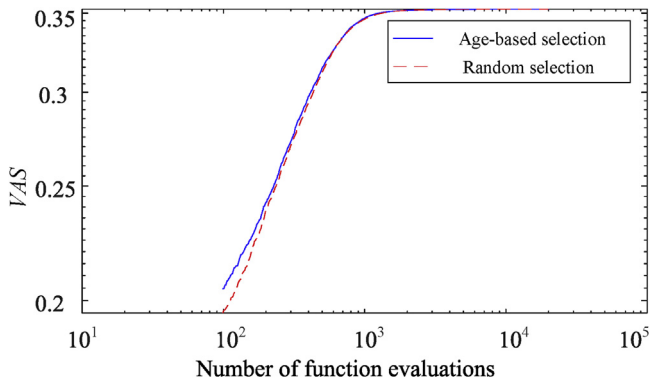


Fig. 12. Average *VAS* of age-based selection and random selection of 3DFCH-EMOA on ZED1 test function.

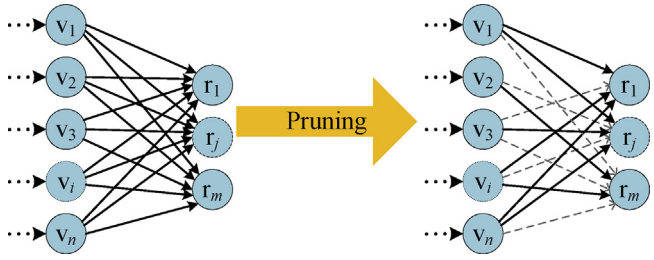


Fig. 13. The diagram of neural networks pruning.

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

4.4. Multiobjective optimization of neural networks pruning

4.4.1. Neural networks pruning problem formulation

Deep neural networks have obtained human-level performance on large-scale classification tasks, however, these deep networks typically contain a large number of parameters due to dense matrix multiplications [55]. Recently, sparse neural networks have been attracted much attention [55], and evolutionary algorithms have been proved to be good tools for neural networks optimization [9,56]. Not only can the computational complexity reduced but also the generalization of neural networks can be improved by neural networks pruning. The diagram of neural networks pruning is shown in Fig. 13.

We apply gate variables $G^s = \{g_1^s, g_2^s, \ldots, g_m^s\}$ for sparsifying weight matrices by performing element-wise multiplication of G^s with $W = \{w_1, w_2, \ldots, w_m\}$, to yield sparse weight matrices $W^s = \{w_1^s, w_2^s, \ldots, w_m^s\}$, as it is denoted by Eq. (10).

$$w^s = w_i \odot g^s, \quad i = 1, 2, \ldots, m \tag{10}$$

It represents an m-layer dense neural network architecture where w_i^s is a weight metric for the i^{th} layer. The gate metric $g_i^s = \{0, 1\}^{n_i}$ contains n_i elements for the i^{th} layer. The sparsity is defined as the complexity objective to be optimized, as it is denoted by Eq. (11),

$$CCR = \frac{\sum_{i=1}^{m} \mathbf{1}\{g_i^s = 1\}}{\sum_{i}^{m} n_i}, \tag{11}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, so that $\mathbf{1}\{\text{a true statement}\} = 1$, and $\mathbf{1}\{\text{a false statement}\} = 0$. A classifier with lower *CCR* should be preferred, as classifiers with a lower *CCR* will have a lower tendency

Table 10
14 balanced and unbalanced UCI datasets.

No.	Data set	No. features	Class distribution
1	australian	14	307:383
2	breast	9	239:444
3	diabetes	8	268:500
4	german	24	700:300
5	heart	13	139:164
6	hill	13	139:164
7	ionosphere	34	225:126
8	liverbupa	6	145:200
9	mask	166	207:269
10	sonar	60	97:111
11	spam	57	1813:2788
12	spectf	44	254:95
13	vote	16	267:168
14	wdbc	30	212:357

of overfitting [9]. In our study we also follow a tri-objective problem formulation for neural networks pruning, as it is denoted by Eq. (12).

$$\min_{G^s \in \Omega} \mathbf{F}(G^s) = \min_{G^s \in \Omega} \left(PPR(G^s), FNR(G^s), CCR(G^s) \right) \tag{12}$$

Neural networks pruning is a combinatorial optimization problem, in this part, several EMOAs are applied to seek sparse neural networks in augmented DET space.

4.4.2. UCI dataset

In this section, a total of 14 two-class datasets from the UCI repository [57] are used to evaluate the performance of EMOAs for neural networks pruning. Both balanced and unbalanced benchmark datasets are included, details are described in Table 10. For all these datasets, 1/4 of instances are randomly selected as training datasets, 1/4 of instances are randomly selected as validation datasets, and the remains instances are selected as test datasets. The training dataset is used for neural networks pre-training, the validation dataset is used for neural networks pruning, and the test dataset is used for performance evaluation.

4.4.3. Algorithms involved

Seven reference EMOAs (NSGA-III, MOEA/D, RVEA, AR-MOEA, MPSO/D, 3DCH-EMOA, and 3DFCH-EMOA) were tested for neural networks pruning. Experiments were performed with LightNet [58] and PlatEMO [53] in Matlab.

4.4.4. Parameter setting

All algorithms mentioned above are used to optimize a multi-layer feedforward network with an input layer with the size of the number of features of each dataset, two hidden layers with 10 and 6 neuron units and an output layer with 2 neuron units. The sigmoid function is selected as activation function in the neural networks. The batch size is set to 5, and 100 epochs are performed for the pre-training stage for each dataset.

Encoding: We employed a binary encoding scheme where the chromosome is constituted by an array of 0 or 1, 0 means drop the connection and 1 means keep the connection between two neuron units. The length of the chromosome is $n_f \times 10 + 10 \times 6 + 6 \times 2$, where n_f is the number of features of each dataset. In the pruning stage, only validation is performed to evaluate the performance of each chromosome.

Configuration: The seven algorithms are set with a maximum of 20,000 function evaluations as the experimental stopping criteria. The binary single-point crossover and bitwise mutation are applied in the experiments. The crossover probability of $P_c = 0.9$ and a mutation probability of $p_m = 1/\sum n_i$, where $\sum n_i$ is the number of gate variables. The population size is set to 100 for all algorithms. All of the algorithms are run 10 times independently. All of these experiments were run on an IBM X3650 server with Xeon E5-2600 2.9GHz processors and 32GB memory under Ubuntu 16.04LTS.

4.4.5. Experimental results and discussion

To evaluate the performance of these algorithms, we compare the statistical results of time cost and classification accuracy, which is defined as the partition of the correctly classified samples to all samples in the test datasets.

Table 11
Mean and standard deviation of accuracy on UCI datasets.

	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA	No Pruning
australian	$8.08e{-}01_{9.58e{-}02}$	$8.13e-01_{9.76e-02}$	$8.08e{-}01_{9.61e-02}$	$8.19e-01_{9.86e-02}$	$7.92e{-}01_{9.44e-02}$	$8.15e-01_{9.79e-02}$	$8.15e{-}01_{9.72e-02}$	$7.46e-1_{7.83e-2}$
breast	$9.34e{-}01_{1.01e-01}$	$9.66e-01_{2.35e-02}$	$9.65e{-}01_{2.18e-02}$	$9.57e-01_{2.56e-02}$	$9.66e{-}01_{1.93e-02}$	$9.73e-01_{2.18e-02}$	$9.71e{-}01_{2.17e-02}$	$9.59e-1_{2.46e-2}$
diabetes	$6.47e{-}01_{5.58e-02}$	$6.47e-01_{5.65e-02}$	$6.45e{-}01_{5.19e-02}$	$6.46e-01_{5.60e-02}$	$6.40e{-}01_{4.26e-02}$	$6.47e-01_{5.57e-02}$	$6.48e{-}01_{5.46e-02}$	$6.23e-1_{2.16e-2}$
german	$7.28e{-}01_{2.95e-02}$	$7.28e-01_{2.98e-02}$	$7.24e{-}01_{2.74e-02}$	$7.26e-01_{2.89e-02}$	$7.16e{-}01_{1.73e-02}$	$7.26e-01_{2.66e-02}$	$7.25e{-}01_{2.69e-02}$	$7.13e-1_{1.58e-2}$
heart	$7.82e{-}01_{8.35e-02}$	$7.92e-01_{8.65e-02}$	$7.89e{-}01_{8.68e-02}$	$7.80e-01_{8.40e-02}$	$8.21e{-}01_{1.97e-02}$	$7.80e-01_{8.31e-02}$	$7.89e{-}01_{8.82e-02}$	$7.91e-1_{3.94e-2}$
hill	$7.72e{-}01_{8.02e-02}$	$7.78e-01_{8.27e-02}$	$7.67e{-}01_{7.84e-02}$	$7.67e-01_{7.90e-02}$	$7.73e{-}01_{8.25e-02}$	$7.79e-01_{8.57e-02}$	$8.00e{-}01_{8.29e-02}$	$7.72e-1_{8.07e-2}$
ionosphere	$8.71e{-}01_{4.57e-02}$	$8.74e-01_{2.75e-02}$	$8.70e{-}01_{4.00e-02}$	$8.70e-01_{3.21e-02}$	$8.62e{-}01_{3.41e-02}$	$8.83e-01_{3.21e-02}$	$8.81e{-}01_{3.08e-02}$	$8.45e-1_{2.45e-2}$
liverbupa	$6.15e{-}01_{6.69e-02}$	$6.15e-01_{6.86e-02}$	$6.08e{-}01_{6.53e-02}$	$6.17e-01_{7.02e-02}$	$5.95e{-}01_{5.63e-02}$	$6.20e-01_{7.28e-02}$	$6.20e{-}01_{7.25e-02}$	$5.67e-1_{2.39e-2}$
mask	$7.85e{-}01_{2.30e-02}$	$7.86e-01_{2.41e-02}$	$7.92e{-}01_{2.50e-02}$	$7.81e-01_{3.73e-02}$	$7.73e{-}01_{2.28e-02}$	$7.94e-01_{2.95e-02}$	$7.92e{-}01_{2.66e-02}$	$7.60e-1_{3.56e-2}$
sonar	$6.34e{-}01_{8.73e-02}$	$6.38e-01_{9.85e-02}$	$6.35e{-}01_{9.06e-02}$	$6.33e-01_{8.67e-02}$	$6.54e{-}01_{9.65e-02}$	$6.38e-01_{9.58e-02}$	$6.44e{-}01_{1.05e-01}$	$6.25e-1_{7.66e-2}$
spam	$9.09e{-}01_{1.43e-02}$	$9.16e-01_{1.17e-02}$	$9.08e{-}01_{1.20e-02}$	$9.18e-01_{1.22e-02}$	$8.96e{-}01_{1.13e-02}$	$9.24e-01_{1.17e-02}$	$9.22e{-}01_{1.03e-02}$	$9.20e-1_{5.00e-3}$
spectf	$7.30e{-}01_{2.41e-03}$	$7.29e-01_{7.23e-03}$	$7.31e{-}01_{1.81e-03}$	$7.31e-01_{1.81e-03}$	$7.27e{-}01_{1.88e-02}$	$7.34e-01_{7.23e-03}$	$7.35e{-}01_{1.08e-02}$	$7.31e-1_{1.80e-3}$
vote	$9.14e{-}01_{9.49e-02}$	$9.25e-01_{9.82e-02}$	$9.13e{-}01_{9.48e-02}$	$9.10e-01_{9.47e-02}$	$9.43e{-}01_{1.70e-02}$	$9.58e-01_{1.30e-02}$	$9.26e{-}01_{9.85e-02}$	$9.15e-1_{2.31e-2}$
wdbc	$8.49e{-}01_{1.36e-01}$	$8.50e-01_{1.37e-01}$	$8.47e{-}01_{1.34e-01}$	$8.49e-01_{1.36e-01}$	$8.37e{-}01_{1.35e-01}$	$8.51e-01_{1.37e-01}$	$8.50e{-}01_{1.36e-01}$	$8.18e-1_{1.42e-1}$
Average	0.7841	0.7898	0.7859	0.7860	0.7854	0.7944	0.7941	0.7622

Table 12
Mean and standard deviation of execution time (s) on UCI datasets.

	NSGA-III	MOEA/DD	RVEA	AR-MOEA	MPSO/D	3DCH-EMOA	3DFCH-EMOA
australian	$2.87e+04_{2.77e+03}$	$6.50e+02_{7.72e+01}$	$3.30e+02_{6.21e+01}$	$3.30e+04_{6.28e+03}$	$3.05e+02_{3.20e+01}$	$3.09e+04_{3.24e+03}$	$3.13e+02_{3.72e+01}$
breast	$2.90e+04_{2.90e+03}$	$6.41e+02_{6.79e+01}$	$3.24e+02_{5.37e+01}$	$3.07e+04_{3.44e+03}$	$3.09e+02_{3.50e+01}$	$3.12e+04_{3.76e+03}$	$3.07e+02_{3.00e+01}$
diabetes	$3.01e+04_{2.65e+03}$	$7.35e+02_{7.99e+01}$	$3.40e+02_{5.07e+01}$	$3.38e+04_{5.04e+03}$	$3.23e+02_{2.05e+01}$	$3.33e+04_{3.02e+03}$	$3.28e+02_{2.56e+01}$
german	$4.12e+04_{4.21e+03}$	$9.00e+02_{7.43e+01}$	$4.66e+02_{6.65e+01}$	$4.57e+04_{8.28e+03}$	$4.38e+02_{4.22e+01}$	$4.41e+04_{3.92e+03}$	$4.46e+02_{4.98e+01}$
heart	$1.49e+04_{6.87e+02}$	$3.52e+02_{2.21e+01}$	$1.83e+02_{8.33e+00}$	$1.64e+04_{5.12e+02}$	$1.53e+02_{1.17e+01}$	$1.59e+04_{1.46e+03}$	$1.60e+02_{6.36e+00}$
hill	$1.47e+04_{4.06e+02}$	$3.51e+02_{2.24e+01}$	$1.77e+02_{1.68e+01}$	$1.62e+04_{2.63e+02}$	$1.57e+02_{9.57e+00}$	$1.62e+04_{1.26e+03}$	$1.56e+02_{1.01e+01}$
ionosphere	$1.67e+04_{3.97e+02}$	$3.72e+02_{2.74e+01}$	$2.03e+02_{2.49e+01}$	$1.77e+04_{1.49e+03}$	$1.84e+02_{2.59e+01}$	$1.78e+04_{1.82e+03}$	$1.81e+02_{1.77e+01}$
liverbupa	$1.64e+04_{1.74e+02}$	$4.59e+02_{1.51e+02}$	$1.84e+02_{2.56e+01}$	$1.79e+04_{2.13e+03}$	$1.72e+02_{1.97e+01}$	$1.75e+04_{2.00e+03}$	$1.74e+02_{1.76e+01}$
mask	$1.97e+04_{1.99e+03}$	$4.69e+02_{6.63e+01}$	$2.27e+02_{3.64e+01}$	$2.31e+04_{4.26e+03}$	$2.20e+02_{2.64e+01}$	$2.20e+04_{3.93e+03}$	$2.25e+02_{3.31e+01}$
sonar	$1.04e+04_{5.52e+02}$	$2.44e+02_{2.08e+01}$	$1.16e+02_{1.69e+01}$	$1.09e+04_{9.23e+02}$	$1.12e+02_{1.62e+01}$	$1.19e+04_{1.54e+03}$	$1.16e+02_{1.81e+01}$
spam	$1.66e+04_{8.41e+01}$	$3.99e+02_{2.42e+00}$	$1.84e+02_{1.42e+00}$	$1.91e+04_{1.38e+02}$	$1.85e+02_{4.66e+00}$	$3.51e+04_{4.06e+03}$	$1.88e+02_{4.18e+00}$
spectf	$1.52e+04_{1.67e+03}$	$3.50e+02_{4.05e+01}$	$1.75e+02_{3.12e+01}$	$1.67e+04_{2.16e+03}$	$1.64e+02_{2.21e+01}$	$1.69e+04_{2.42e+03}$	$1.61e+02_{1.39e+01}$
vote	$1.80e+04_{1.79e+03}$	$4.06e+02_{4.06e+01}$	$2.12e+02_{4.05e+01}$	$1.91e+04_{2.26e+03}$	$1.91e+02_{2.09e+01}$	$1.93e+04_{2.15e+03}$	$1.93e+02_{2.20e+01}$
wdbc	$2.33e+04_{2.39e+03}$	$5.23e+02_{5.17e+01}$	$2.67e+02_{3.87e+01}$	$2.52e+04_{2.44e+03}$	$2.42e+02_{1.36e+01}$	$2.47e+04_{1.84e+03}$	$2.52e+02_{3.43e+01}$

Table 11 shows the mean and standard deviation of accuracy obtained by these EMOAs for UCI datasets. In this part, the highest accuracy in the population is listed in the table. To compare the results, the average accuracy of these UCI datasets is listed in the bottom of the table. From the table, we can make some conclusions: (1) EMOAs can obtain higher accuracy than neural networks without pruning on most UCI datasets; (2) 3DCH-EMOA and 3DFCH-EMOA outperforms other EMOAs on most UCI datasets; (3) 3DFCH-EMOA performs as good as 3DCH-EMOA on most of these UCI datasets.

Table 12 shows the mean and standard deviation of time cost for UCI datasets. In the table, time cost is computed for neural networks pruning stage only. In the pruning stage, neural networks performance validation is executed for the chromosome evaluation; as the time cost for validation is not time-consuming, the time cost most depends on the computational complexity of each algorithm. From the table, we can see that the new proposed algorithm cost less time than 3DCH-EMOA, as in the neural networks pruning problems 3DFCH-EMOA cost far less computation than 3DCH-EMOA while dealing with solutions in non-*FS*set by applying age-based selection strategy.

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 computation of the contribution of each vertex to the convex hull volume is proposed. 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 and neural networks pruning 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 300, without reducing the performance of the method. Moreover, the benchmark was extended by modern algorithms, such as NSGA-III and MPSO/D.

Alternatively, the computational time complexity of iteratively computing the convex hull, which is $O(n^{(\lfloor d/2 \rfloor+2)})$ [59] 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.

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, 61572505 and 61772530), and the National Natural Science Foundation of Jiangsu Province (No. BK20171192). We also thank the authors of PlatEMO [53] and LightNet [58].

References

[1] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett. 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, Crit. Rev. Comput. Tomogr. 29 (3) (1989) 307–335.
[4] T. Fawcett, Using rule sets to maximize ROC performance, IEEE International Conference on Data Mining (2001) 131–138, http://dx.doi.org/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 Knowl. Discov. 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 Trans. Evol. Comput. 19 (2) (2015) 188–200, http://dx.doi.org/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, Inform. Sci. 367–368 (2016) 80–104.
[10] W. Hong, K. Tang, Convex hull-based multi-objective evolutionary computation for maximizing receiver operating characteristics performance, Memetic Comput. 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 Trans. Cybern. 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? Appl. Soft Comput. 48 (2016) 597–611.
[14] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. Coello, Survey of multiobjective evolutionary algorithms for data mining: Part II, IEEE Trans. Evol. Comput. 18 (1) (2014) 20–35, http://dx.doi.org/10.1109/TEVC.2013.2290082.
[15] S. Wang, L.L. Minku, X. Yao, A multi-objective ensemble method for online class imbalance learning, 2014 International Joint Conference on Neural Networks (IJCNN) (2014) 3311–3318, http://dx.doi.org/10.1109/IJCNN.2014.6889545.
[16] L. Li, X. Yao, R. Stolkin, M. Gong, S. He, An evolutionary multiobjective approach to sparse reconstruction, IEEE Trans. Evol. Comput. 18 (6) (2014) 827–845, http://dx.doi.org/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, Appl. Soft Comput. 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, Int. Resour. Manage. J. 26 (2012) 54–67.
[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[21] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[22] Z. Wang, Q. Zhang, A. Zhou, M. Gong, L. Jiao, Adaptive replacement strategies for MOEA/D, IEEE Trans. Cybern. 46 (2) (2016) 474–486.
[23] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, Eur. J. Oper. Res. 181 (3) (2007) 1653–1669.
[24] I. Hupkens, M. Emmerich, Logarithmic-time updates in SMS-EMOA and hypervolume-based archiving, EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation (2013) 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, Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign-Urbana, 1976.
[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, Appl. Soft Comput. (2016) 111–123.
[29] V. Basto-Fernandes, I. Yevseyeva, D. Ruano-Ordás, J. Zhao, F. Fdez-Riverola, J.R. Méndez, M.T.M. Emmerich, Quadcriteria Optimization of Binary Classifiers: Error Rates, Coverage, and Complexity, Springer International Publishing, Cham, 2018, pp. 37–49.
[30] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, CEC 2005, Edinburgh, UK, 2–4 September 2005, IEEE Press, 2005, pp. 443–450.
[31] 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.

[32] 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.
[33] S. Yitzhaki, Relative deprivation and the gini coefficient, Q. J. Econ. 93 (2) (1979) 321–324.
[34] C. Lücken, B. Barán, C. Brizuela, A survey on multi-objective evolutionary algorithms for many-objective problems, Comput. Optim. Appl. 58 (3) (2014) 707–756.
[35] H. Wang, L. Jiao, X. Yao, Two_Arch2: An improved two-archive algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (4) (2015) 524–541.
[36] 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 Trans. Evol. Comput. 18 (4) (2014) 577–601.
[37] K. Li, K. Deb, Q. Zhang, S. Kwong, An evolutionary many-objective optimization algorithm based on dominance and decomposition, IEEE Trans. Evol. Comput. 19 (5) (2015) 694–716.
[38] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 773–791.
[39] Y. Tian, R. Cheng, X. Zhang, F. Cheng, Y. Jin, An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility, IEEE Trans. Evol. Comput. PP (99) (2017), 1-1.
[40] C. Dai, Y. Wang, M. Ye, A new multi-objective particle swarm optimization algorithm based on decomposition, Inform. Sci. 325 (C) (2015) 541–557.
[41] J. O'Rourke, Computational Geometry in C, Cambridge University Press, 1998.
[42] D.R. Chand, S.S. Kapur, An algorithm for convex polytopes, J. Assoc. Comput. Mach. 17 (1) (1970) 78–86.
[43] G.S. Brodal, R. Jacob, Dynamic planar convex hull, The 43rd Annual IEEE Symposium on Proceeding of Foundations of Computer Science (2002) 617–626.
[44] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Softw. (TOMS) 22 (4) (1996) 469–483.
[45] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F.M.A.D. Heide, H. Rohnert, R.E. Tarjan, Dynamic perfect hashing: upper and lower bounds, 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (1988) 524–531.
[46] F.P. Preparata, S.J. Hong, Convex hulls of finite sets of points in two and three dimensions, Commun. ACM 20 (2) (1977) 87–93.
[47] K.L. Clarkson, P.W. Shor, Applications of random sampling in computational geometry, II, Discrete Comput. Geom. 4 (5) (1989) 387–421.
[48] K. Clarkson, L. Kenneth, R. Mehlhorn, Seidel, Four results on randomized incremental constructions, Comput. Geom. Theory Appl. 3 (1993) 185–212, http://dx.doi.org/10.1016/0925-7721(93)90009-U.
[49] G. Rote, K. Buchin, K. Bringmann, S. Cabello, M. Emmerich, Selecting K points that maximize the convex hull volume, The 19th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG3 2016) (2016) 58–60.
[50] A. Ghosh, S. Tsutsui, H. Tanaka, Individual aging in genetic algorithms, Proceedings of Australian and New Zealand Conference on Intelligent Information Systems (1996) 276–279.
[51] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surveys 45 (3) (2013) 533–545.
[52] 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 Trans. Evol. Comput. 17 (2) (2013) 241–258.
[53] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum], IEEE Comput. Intell. Mag. 12 (4) (2017) 73–87.
[54] H. Wang, Y. Jin, X. Yao, Diversity assessment in many-objective optimization, IEEE Trans. Cybern. 47 (6) (2017) 1510–1522, http://dx.doi.org/10.1109/TCYB.2016.2550502.
[55] S. Srinivas, A. Subramanya, R.V. Babu, Training sparse neural networks, Computer Vision and Pattern Recognition Workshops (2017) 455–462.
[56] A.L. Rincon, A. Tonda, M. Elati, O. Schwander, B. Piwowarski, P. Gallinari, Evolutionary optimization of convolutional neural networks for cancer miRNA biomarkers classification, Appl. Soft Comput. 65 (2018) 91–100.
[57] K. Bache, M. Lichman, UCI machine learning repository.
[58] C. Ye, C. Zhao, Y. Yang, Y. Aloimonos, Lightnet: A versatile, standalone Matlab-based environment for deep learning, ACM on Multimedia Conference (2016) 1156–1159.
[59] R. Seidel, Convex Hull Computations, CRC Press, Inc., 1997.