Applied Intelligence (2021) 51:1118–1132
https://doi.org/10.1007/s10489-020-01791-4

HFADE-FMD: a hybrid approach of fireworks algorithm and differential evolution strategies for functional module detection in protein-protein interaction networks

Junzhong Ji[1,2] · **Hanghang Xiao**[1,2] · **Cuicui Yang**[1,2]

Published online: 16 September 2020

Abstract

Functional module detection in protein-protein interaction (PPI) network is one important content of the proteomics research in the post-genomic era. Nowadays the swarm intelligence and evolutionary based approaches have become effective ways for detecting functional modules. This paper proposes a novel hybrid approach of fireworks algorithm and differential evolution strategies for functional module detection in PPI networks (called HFADE-FMD). HFADE-FMD first initializes each firework individual into a candidate functional module partition based on label propagation according to the topological and functional information between protein nodes. Then HFADE-FMD uses the explosion operator of firework algorithm, and mutation, crossover and selection strategies of differential evolution algorithm to iteratively search for better functional module partitions. To verify the performance of HFADE-FMD, this paper compared it with ten competitive methods on four public PPI datasets. The experimental results show that HFADE-FMD achieves prominent performance with respective to Recall, Sn, PPV, and ACC metrics while performing well in terms of Precision and F-measure metrics. Thus, it is able to more accurately detect functional modules and help biologists to find some novel biological insights.

Keywords Protein-protein interaction network · Functional module detection · Fireworks algorithm · Explosion operation · Differential evolution strategies

1 Introduction

With the completion of human genome project, proteomics has become the frontier of life science and natural science research in the post-genome time. Proteins are the main undertakers of the life activity where they rarely perform a function in an individual way, but do it by interacting with other proteins [1]. All the interactions between proteins in an organism form a kind of molecular networks called protein-protein interaction (PPI) network. The set of proteins that interact with each other to perform a specific biological function in a PPI network is a functional module. Identifying functional modules in PPI networks is an important part of proteomics, which not only helps people understand the mechanism of life at the molecular level, but also has great significance for the diagnosis of diseases and the development of new drugs [2].

The biological experiments, as the original methods of detecting functional modules in PPI networks, often take a lot of time, manpower and material resources, which appears to be quite helpless when facing the explosive growth of PPI data and can not meet the urgent need of life science research in the post-genome time [3]. Fortunately, computational approaches based on clustering obtain a rapid development due to the advantages of short time and low cost, and have become more important means of detecting functional modules in recent years. In general, these computational approaches based on clustering first represent a PPI network by an undirected graph $G(V, E)$,

✉ Cuicui Yang
yangcc@bjut.edu.cn

Junzhong Ji
jjz01@bjut.edu.cn

Hanghang Xiao
xiaohh70@qq.com

[1] Beijing College of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing, China

[2] Municipal Key Laboratory of Multimedia and Intelligent Software Technology, Beijing Artificial Intelligence Institute, Beijing, China

where V is the set of nodes with each node denotes a protein and E is the set of edges with each edge denotes an interaction (link). Then they uses various clustering technologies to divide the network and get corresponding functional modules [2]. So far there have been many kinds of clustering approaches which adopt different ideas and schemes to detect functional modules such as density based clustering approaches [4–6], hierarchy based clustering approaches [7, 8], partition based clustering approaches [9–11], flow simulation approaches [12, 13], spectral clustering based approaches [14, 15], core attachment approaches [16, 17], and supervised approaches [18, 19]. Besides swarm intelligence and evolutionary algorithms which simulate certain behavioral activities and evolutionary mechanisms of natural organisms have been widely applied into clustering analysis of different fields [20–26]. It is noted that although functional module detection is a clustering problem in essence, different application fields have their own unique characteristics. It is just for the unique characteristics which make the implementation of swarm intelligence and evolutionary algorithms different. To release potential of swarm intelligence and evolutionary algorithms for functional module detection, some researchers study the individual solution representation and realization of optimization mechanisms. Now swarm intelligence and evolutionary algorithms have also become the new spot in detecting functional modules from PPI networks. For example, Sallim et al. introduced ant colony optimization (ACO) into functional module detection by taking it as the optimization problem of solving traveling salesman problem, which started a new page for detecting functional modules using swarm intelligence and evolutionary based methods [27]. Pizzuti et al. made use of genetic algorithm to mine functional modules (called as GA-FMD) [28]. Ji et al. combined ACO with multi-agent evolution to identify functional modules [29]. Subsequently, they again presented an algorithm based on ant colony clustering for functional module detection (called as ACC-FMD) [30]. Recently, Yang et al. developed an method based on bacterial foraging optimization for detecting functional modules (called as BFO-FMD) [31]. Although many effective swarm intelligence and evolutionary based approaches (as listed above) have been proposed for detecting functional modules, this kind of approaches are still evolving with efforts aimed at detecting functional modules more effectively in face of the explosive growth of PPI data nowadays.

Inspired by explosion of fireworks, Tan and Zhu proposed a new swarm intelligence algorithm called fireworks algorithm (FA) in 2010 [32]. This algorithm takes each firework as a candidate solution in the search space. Each firework would explode and generate a set of sparks. The fireworks and sparks with better fitness values are selected to form the fireworks of the next generation. This explosion process continues until a desired solution is found, or the stopping criterion is met. Although FA does not come early, it have been widely used in many fields due to being able to self-tune local search and global search [33–40]. This good problem-solving capability inspire us to develop its potential to detect functional modules from PPI networks. However, references [41–44] have found that FA lacks of information communication among firework individuals which easily cause the algorithm to fall into local optima. At the same time, these studies find differential evolution (DE) algorithm can help to enhance information communication among firework individuals. Among them, references [41–43] used the hybrid method of FA and DE to solve function optimization problems while reference [44] applied the hybrid method of FA and DE into the design of fuzzy classification system. That is, a hybrid of FA and DE algorithm can enhance their optimization ability on different application areas. In fact, many other studies also agree that a hybrid of different swarm intelligence and evolutionary algorithms can learn from each other and have the advantages that single method do not have [45–52]. Thus, to find a more effective way to detect functional modules, this paper propose a novel hybrid approach of firework algorithm and differential evolution algorithm for functional module detection in PPI networks (called HFADE-FMD). The main contributions of HFADE-FMD are summarized as follows: (1) This paper presents a novel approach which takes advantage of FA and DE for functional module detection. (2) Aimed at the problem of detecting functional modules, HFADE-FMD uses a new individual initialization methods and new implementation method of optimization mechanisms. (3) Systematic experiments have not been only conducted on the Saccharomyces cerevisiae datasets but also on the Homo sapiens datasets. The experimental results show that HFADE-FMD has prominent performance in term of Recall, Sn, PPV, and ACC metrics while performing well on Precision and F-measure metrics. Therefore, this method is highly competent to detect functional modules in PPI networks.

This paper is structured as follows: Section 2 presents the proposed HFADE-FMD algorithm in detail. Section 3 carries out comparative experiments to validate the performance of HFADE-FMD. Finally, Section 4 concludes this paper and outlines some future research directions.

2 The proposed HFADE-FMD algorithm

2.1 Basic idea

The proposed HFADE-FMD algorithm takes functional module detection (i.e., clustering on protein nodes) as an

optimization problem. To solve this optimization problem, HFADE-FMD first initialized each firework individual into a candidate functional module partition based on label propagation. Then it optimizes iteratively each firework individual by the explosion operation (the core optimization mechanism of FA), mutation, crossover, and selection operations (three DE strategies). During the optimization process, each firework individual (a firework or a spark) is evaluated according to the modularity density (the fitness function) that is commonly used in the partition of a network into modules [53], and is given below:

$$f(X) = \sum_{i=1}^{m} \frac{2l_i - \bar{l}_i}{n_i}, \tag{1}$$

where X is a candidate module partition represented by a individual solution (a firework or a spark), m is the number of modules, l_i is the number of edges between nodes in module i, $\bar{l}_i$ is the number of edges between one node in module i and the other node outside it, and n_i is the number of nodes in module i. The higher the modularity density, the better the detected functional module partition (i.e., the better the corresponding individual). In the following, the key parts including the explosion operator of FA, and three differential evolution strategies will be explained in detail.

2.2 Initialization of firework individuals

Each firework individual corresponds to a candidate functional module partition, and is encoded as a string of length n where n is the number of protein nodes in a PPI network and the character at each location is the label of the corresponding protein node. Each firework individual is initialized based on label propagation. First, HFADE-FMD numbers all the protein nodes in a PPI network and builds a neighbor order list for each protein node. Then, each firework individual probabilistically selects a neighbor node for each protein node. Now the authors assume a firework individual selects a neighbor node h for a protein node k from its neighbor nodes, then two nodes k and h will be assigned same label according to the following three rules of label propagation:

(1) If both protein nodes k and h are not assigned labels yet, then assign them a same and new label.
(2) If one of protein nodes k and h is assigned a label, then propagate this label to the other node.
(3) If both protein nodes k and h are assigned labels, then do nothing in case of that they have same labels, and otherwise, propagate the label of node k to those nodes that have the same label with node h.

In the above initialization process, the probability that each firework individual selects a neighbor node h for a node k is given as follows:

$$P_k^h = \frac{S_{kh}}{\sum_{r \in \Gamma(k)} S_{kr}}, \tag{2}$$

where $\Gamma(k)$ is the neighbor node set of node k, S_{kh} is the similarity between nodes k and h. Here, S_{kh} combines the structural similarity and functional similarity between nodes k and h, and is defined as below:

$$S_{kh} = \frac{s_{kh} + f_{kh}}{2}, \tag{3}$$

where s_{kh} and f_{kh} are the structural similarity and functional similarity between nodes k and h, respectively.

Given two protein nodes k and h, the structural similarity formula is given as follows [54]:

$$s_{kh} = \frac{|\Gamma(k) \bigcap \Gamma(h)|}{\sqrt{|\Gamma(k)||\Gamma(h)|}}, \tag{4}$$

where $|\Gamma(k)|$ is the size of the set $\Gamma(k)$.

Based on the annotation information of gene ontology (GO), the function similarity is expressed as follow [55]:

$$f_{kh} = \frac{|g^k \bigcap g^h|}{|g^k \bigcup g^h|}, \tag{5}$$

where g^k and g^h are GO term sets of nodes k and h, respectively.

To clearly show the solution representation and initialization process, Fig. 1 illustrates an example in a simplified PPI network with 8 nodes. Figure 1a is a PPI network containing 8 nodes marked from 1 to 8. Figure 1b gives the neighbor order lists of 8 nodes. Figure 1c shows a firework individual solution obtained by the above label propagation process. Figure 1d is the functional modules corresponding to the firework individual shown in Fig. 1c.

2.3 Explosion operator

The explosion operator is the core mechanism of FA. Each firework explodes and produces a certain number of sparks. One explosion can be viewed as a search around a firework. The explosion operator is able to self-tune local search and global search. The better the firework individual, the more the sparks and the smaller the amplitude of explosion. The more sparks can make a careful search around the corresponding firework individual in a smaller range, which reflects the local search process. The worse the firework individual, the less the sparks and the bigger the amplitude of explosion. The less sparks will make a extensive exploration in a bigger range, which embodies the

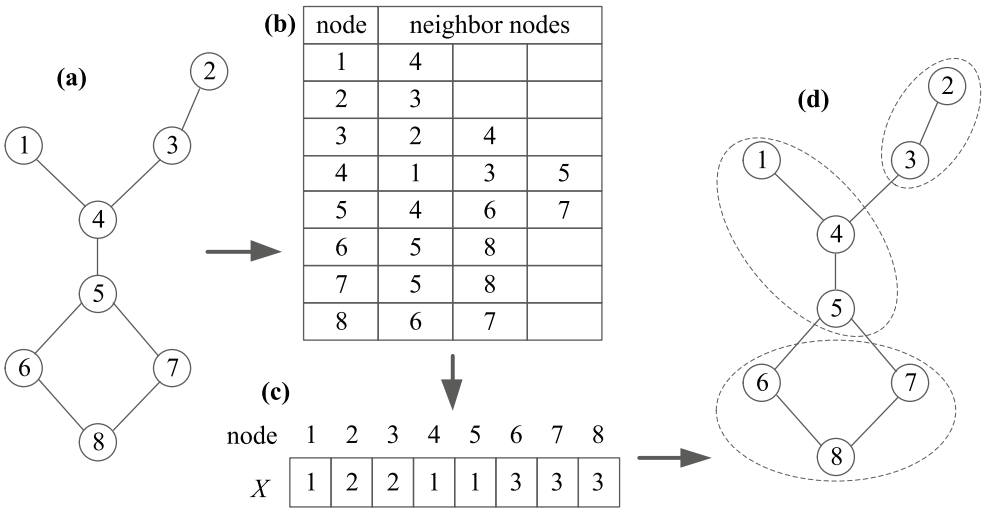


Fig. 1 Firework individual representation and initialization process: **a** PPI network, **b** neighbor order lists, **c** firework individual solution, and **d** a functional module partition

global search process. In general, the number of sparks and the amplitude of explosion for firework individual X_i are respectively defined as below:

$$B_i = \check{B} \times \frac{f(X_i) - f_{\min} + \varepsilon}{\sum_{i=1}^{N}(f(X_i) - f_{\min}) + \varepsilon}, \tag{6}$$

$$R_i = \check{R} \times \frac{f(X_i) - f_{\max} + \varepsilon}{\sum_{i=1}^{N}(f(X_i) - f_{\max}) + \varepsilon}, \tag{7}$$

where N is the size of the firework population, $f_{\min}$ and $f_{\max}$ are respectively the highest and lowest modularity density among all the firework individuals, $\check{R}$ and $\check{B}$ are two parameters that respectively control the explosion amplitude and the number of sparks generated, and ε is a small constant to avoid zero-division-error.

For each firework individual X_i, the method of generating a spark individual X_{sp} is described as follows. First, produce a random number q uniformly distributed in (0, 1) for each protein node k. If this random number is not less than $sigmoid(R_i)$, the label of node k in the spark individual X_{sp} is the same as that in the firework individual X_i. Otherwise, the label of node k in the spark individual X_{sp} is the same as that of the neighbor node which has the largest influence on node k.

The influence that a neighbor node h on node k is defined as follows:

$$I_k^h = \alpha \times \frac{num_k}{|\Gamma(k)|} + (1-\alpha) \times \frac{S_{kh}}{\sum_{j \in \Gamma(k)} S_{kj}}, \tag{8}$$

where num_k is the number of neighbor nodes which have the same label with node k, α is a parameter that controls the importance of similarity nodes and nodes with the same labels. From this definition, the explosion operation not only considers the number of nodes with same labels, but also the similarity between nodes, which makes full use of interrelationships between nodes in PPI networks.

After each firework individual X_i explodes and produces B_i spark individuals at each iteration, all the firework individuals and the spark individuals form a big population. The best individual (whether firework or spark) with the highest modularity density is chosen to be a firework individual of the next generation. Then $N-1$ firework individuals are selected from the remaining big population using the roulette strategy. In this way, HFADE-FMD realizes the explosion operator and produces a new generation of firework population.

2.4 Differential evolution operators

Although the explosion operator combines local search and global search, there is little communication and cooperation among firework individuals, which is very detrimental to find a global optimal functional module partition. Introduced by Storn and Price, DE algorithm has much interaction among individual solutions [56]. Thus, to compensate the disadvantage of explosion operator in FA, three DE operators are used to help firework individuals to search better module partitions. The specific implementation of three DE strategies are as follows.

2.4.1 Mutation operator

According to the mutation probability P_m, each firework individual X_i mutates into a variation individual V_i by the random single point variation in HFADE-FMD. The concrete steps are as follows. First, randomly select a node k for firework individual X_i. Then, probabilistically choose a neighbor node h from neighbor nodes of node k according to (2), and pass the label of node h to node k based on the mutation probability P_m. After the above process is repeated n times (n is the number of nodes in the PPI network), the mutation operator is completed. Figure 2 shows the method of updating the label for one node in the mutation operator. In Fig. 2, the firework individual X_i chooses neighbor node 3 for node 4 according to probability formula (2). The label of node 3 is 2. If a random number in (0,1) is smaller than the mutation probability P_m, pass the label 2 to node 4.

2.4.2 Crossover operator

HFADE-FMD makes use of one-way mergence crossover operator to produce a trial individual U_i for the variation individual V_i of firework individual X_i. The detailed procedures are given below. Firstly, randomly select a firework individual as one source individual Y_1 and take V_i as the other source individual. Secondly, randomly choose a crossover position k. Let $D_{Y_1}(k)$ denotes the node set of the functional module that node k belongs to in source individual Y_1, and $D_{Y_2}(k)$ represents the node set of the functional module that node k belongs to in source individual Y_2. At last, change the labels of nodes that are in $D_{Y_1}(k)$ and do not belong to $D_{Y_2}(k)$ into the label of node k in source individual Y_2, and generate a trail individual U_i.

To be more clear, Fig. 3 gives the sketch map of one-way mergence crossover operator. As shown in Fig. 3, randomly select a crossover position on node 7. $D_{Y_1}(k)$, i.e., the node set of functional module that node 7 belongs to in source individual Y_1, is $\{7, 8\}$. $D_{Y_2}(k)$, i.e., the node

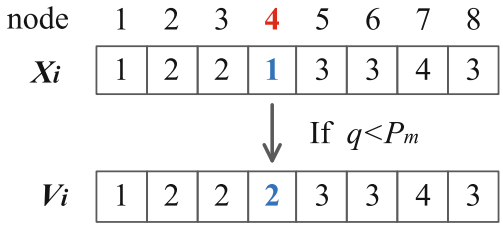


Fig. 2 The method of updating the label for one node in mutation operator

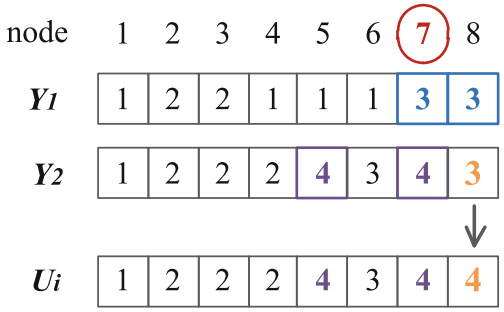


Fig. 3 The sketch map of the crossover operator

set of functional module that node 7 belongs to in source individual Y_2, is $\{5, 7\}$. The node 8 is in $D_{Y_1}(k)$ and do not belong to $D_{Y_2}(k)$. So change the label of node 8 into the label of node 7 in source individual Y_2, and obtain a trial individual U_i.

2.4.3 Selection operator

After performing the crossover operator, HFADE-FMD makes evaluation on firework individual X_i and its trial individual U_i, and keeps the one with higher modularity density according to the following formula:

$$X_i = \begin{cases} U_i, \ if \ f(U_i > f(X_i) \\ X_i, \ otherwise \end{cases}. \tag{9}$$

2.5 Algorithm description

According to the previously detailed introduction, the pseudocode of HFADE-FMD is given in Algorithm 1. This algorithm first starts with an initial firework population, each of which represents a module partition and is constructed based on label propagation. Next, it iteratively executes explosion, mutation, crossover, and selection operators to search better modularity partitions with higher modularity density.

Based on the description of Algorithm 1, the complexity of HFADE-FMD can be analyzed simply as follows. Let the maximum number of the node degree be $d_{\max}$. The time complexity of the initialization process is $O(N \times n \times d_{\max})$. The time complexity of the explosion operator is $O(T \times N \times n \times d_{\max} \times \check{B})$. The time complexity of three differential evolution operators is $O(T \times N \times d_{\max} \times (N + d_{\max}))$. Therefore, the time complexity of the proposed HFADE-FMD algorithm is $O(T \times N \times d_{\max} \times (\check{B} \times n + n + d_{\max}))$.

Algorithm 1 The proposed HFADE-FMD algorithm.

Input: a PPI network $G(V, E)$
Output: the set of modules M
1 **Initialization:**
2 **a) Set parameters:** N, T, $\check{B}$, $\check{R}$, α, P_m, P_c.
N: the size of the firework swarm,
T: the maximum number of iterations,
$\check{B}$: the parameter that controls the number of sparks,
$\check{R}$: the parameter that controls the explosion amplitude,
α: the parameter that relates to the node influence,
P_m: the mutation probability,
P_c: the crossover probability.
3 **b) Initialize the firework population**:
For $i = 1$ to N:
do the initialization process as what described in Section 2.2 and get
ith candidate solution represented by firework i.
4 **The iteration process:**
5 For $t = 1$ to T
6 For $i = 1$ to N
7 firework i performs the explosion operator.
8 firework i performs the mutation operator.
9 firework i performs the crossover operator.
10 firework i performs the selection operator.
11 **Find the module partition with the highest modularity density**.
12 **Return the the set of modules** M.

3 Evaluation

In this section, extensive experiments will be taken to evaluate the performance of the proposed HFADE-FMD algorithm. The experimental platform is a PC with Intel Core i5-6500 3.20GHz CPU, 12.0GB RAM and Windows 8.1, and HFADE-FMD is implemented by Java language.

3.1 Datasets

Four public PPI datasets including two Saccharomyces cerevisiae datasets and two Homo sapiens datasets are used in the experiments. Table 1 shows a summary of the datasets, where the 3th and 4th columns provide the number of proteins and the number of interactions respectively. To evaluate the functional modules discovered by a computational method, two set of gold standard functional modules for each species are used as the benchmark. Two benchmarks of Saccharomyces cerevisiae are SGD [57] and CYC2008 [58], while two benchmarks of Homo sapiens are PCDq [59] and CORUM [60]. Table 2 gives a summary of the four benchmarks, where the 3th and 4th columns provide the number of functional modules and the web links respectively.

3.2 Evaluation metrics

Two types of popular evaluation metrics are used to evaluate the quality of the detected modules [3].

3.2.1 Precision, Recall, F-measure

Precision, Recall, and F-measure are three common evaluation metrics in information retrieval and machine learning. Using the three metrics, it is necessary to define how well a detected module $h = (V_h, E_h)$ matches a standard module $s = (V_s, E_s)$. Many researches use a Neighborhood Affinity score (NA) to assess the matching degree, which is given as follows:

$$NA(h, s) = \frac{|V_h \bigcap V_s|^2}{|V_h| \times |V_s|}. \tag{10}$$

If $NA(h, s) \geq \omega$, two modules h and s are considered to be matched (generally, $\omega = 0.2$). Let H be the set of detected modules and S be the set of gold standard functional modules. The number of the detected modules in H which at least matches one standard module in S is denoted by $N_{ch} = |\{h|h \in H, \exists s \in S, NA(h, s) \geq \omega\}|$, while the number of the standard modules in S which at least matches one detected module in H is indicated by $N_{cs} = |\{s|s \in S, \exists h \in H, NA(h, s) \geq \omega\}|$. Thus, Precision and Recall is given below:

$$Precision = \frac{N_{ch}}{|H|}, \tag{11}$$

Table 1 Datasets used in experiments

Species	Datasets	Number of proteins	Number of interactions
Saccharomyces cerevisiae	Gavin	1855	7669
	KroganCore	2701	7118
Homo sapiens	DIPCore	4356	6586
	DIPFull	4561	6990

Table 2 Set of gold standard functional modules used in experiments

Species	Set of gold standard functional modules	Number of functional modules	Http address
Saccharomyces cerevisiae	SGD	372	http://www.yeastgenome.org/
	CYC 2008	408	http://wodaklab.org/cyc2008/
Homo sapiens	PCDq	1261	http://h-invitational.jp/hinv/pcdq/
	CORUM	1738	https://mips.helmholtz-muenchen.de/

and

$$Recall = \frac{N_{cs}}{|S|}. \tag{12}$$

F-measure is a harmonic mean of Precision and Recall. So it can be used to evaluate the overall performance, and is expressed as:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{13}$$

3.2.2 Sensitivity, positive predictive value and accuracy

Sensitivity (Sn), Positive Predictive Value (PPV) and Accuracy (Acc) are three usual metrics to evaluate the performance of a detection method. Let T_{ij} be the number of the common proteins in ith standard module and jth detected module, then Sn and PPV are defined as:

$$Sn = \frac{\sum_{i=1}^{|S|} \max_j \{T_{ij}\}}{\sum_{i=1}^{|S|} N_i}, \tag{14}$$

and

$$PPV = \frac{\sum_{j=1}^{|H|} \max_i \{T_{ij}\}}{\sum_{j=1}^{|H|} T_{\cdot j}}, \tag{15}$$

where N_i is the number of the proteins in the ith standard module, and $T_{\cdot j} = \sum_{i}^{|S|} T_{ij}$.

In general, high Sn means that the detected results have a good coverage of the proteins in the gold standard modules, while high PPV represents the detected modules are likely to be the standard modules. Acc is a comprehensive metric, and is defined as the geometric mean of Sn and PPV:

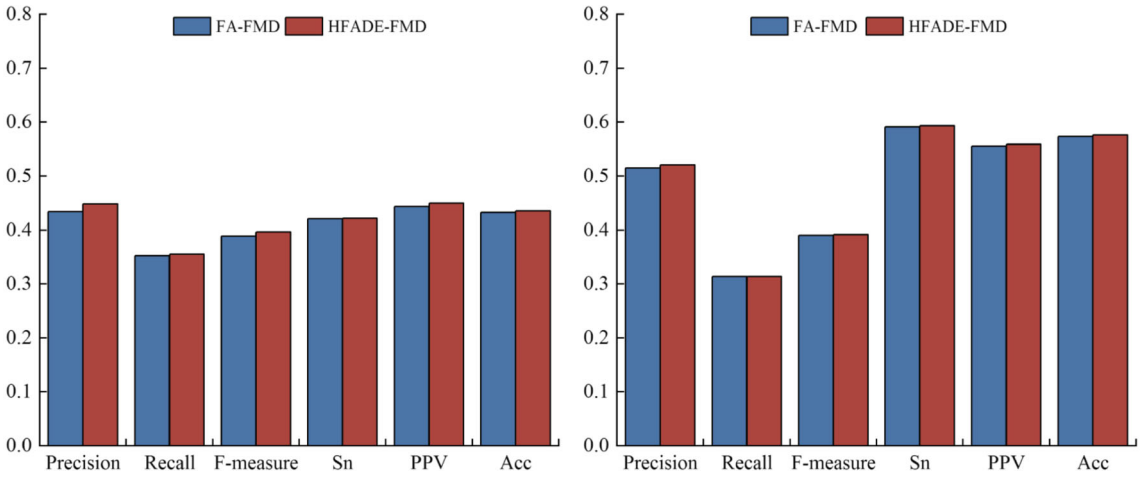
$$Acc = (Sn \times PPV)^{1/2}. \tag{16}$$

3.3 Effects of DE strategies

In this section, the authors investigated the effects of DE strategies by comparing HFADE-FMD with FA-FMD (without using the DE strategies). In the experiments, the authors chose a set of parameters after doing some preliminary experimentations, and set $N = 40$, $T = 100$, $\check{R} = 90$, $\check{B} = 50$, $\alpha = 0.4$, $P_m = 0.1$, $P_c = 0.8$. Table 3 provides the basic information of the detection results for the two algorithms on the four datasets. For each algorithm, the authors have listed the number of detected modules (Number of modules), the average number of proteins in each module (Average size of modules), the number of detected modules which match at least one gold standard module (N_{ch}) and the number of gold standard modules that match at least one detected module (N_{cs}). Taking HFADE-FMD on Gavin dataset as an example, it has detected 194 modules, and the average size of the

Table 3 The basic results of FA-FMD and HFADE-FMD on four datasets

Dataset	Algorithm	Number	Average size	SGD		CYC2008	
		of modules	of modules	$N_{ch} \geq 0.2$	$N_{cs} \geq 0.2$	$N_{ch} \geq 0.2$	$N_{cs} \geq 0.2$
Gavin	FA-FMD	196	9.46	85	131	101	128
	HFADE-FMD	194	9.56	86	134	101	132
KroganCore	FA-FMD	341	7.92	120	162	146	173
	HFADE-FMD	315	8.57	114	151	139	162
DIPCore	FA-FMD	675	6.45	91	91	123	241
	HFADE-FMD	671	6.49	92	93	121	245
DIPFull	FA-FMD	695	6.56	92	92	121	237
	HFADE-FMD	689	6.62	93	94	120	238



(a) using SGD as gold standard functional modules (b) using CYC2008 as gold standard functional modules

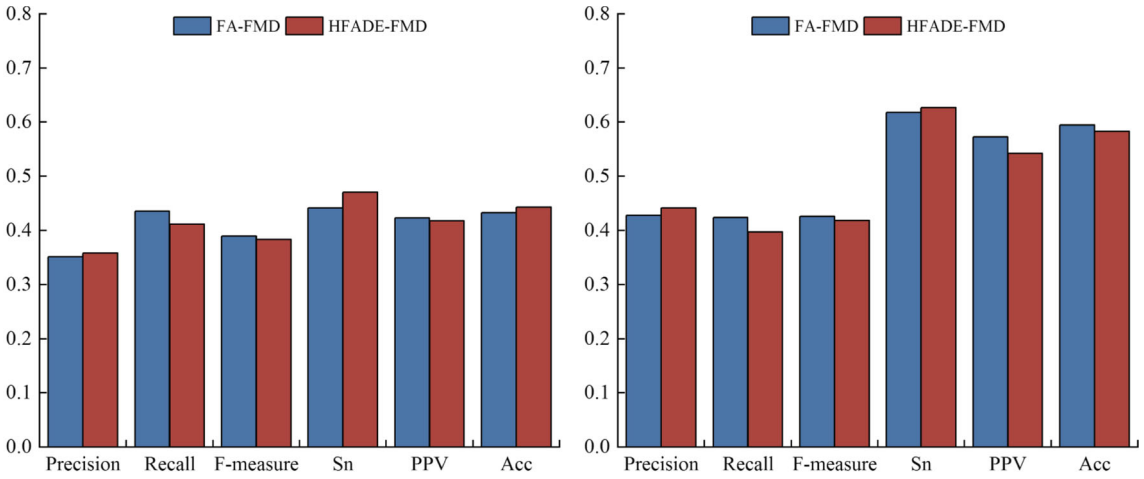
Fig. 4 Comparative results of HFADE-FMD and FA-FMD in terms of various evaluation metrics on Gavin dataset

194 detected modules is about nine proteins. Among the 194 detected modules, there are 86 modules matching 134 standard modules using SGD as benchmark, while there are 101 detected modules matching 131 standard modules using CYC2008 as benchmark.

Figures 4, 5, 6, and 7 show the comparison results of HFADE-FMD and FA-FMD on the four datasets in terms of various evaluation metrics including Precision, Recall, F-measure, Sensitivity, PPV, and Accuracy. From these figures, it is easy to see that no matter which benchmark is used, HFADE-FMD performs better than FA-FMD in terms of nearly all metrics on three out of four datasets (Gavin, DIPCore and DIPFull). As for the one remaining dataset (KroganCore), HFADE-FMD and FA-FMD play equally well using SGD as benchmark, and they both get the best results on three metrics. Only in the case of using CYC2008 as benchmark for KroganCore dataset, HFADE-FMD is slightly worse than FA-FMD. HFADE-FMD obtains the best results on two out of six metrics while FA-FMD gets the best results on four out of six metrics. On the whole, the above experimental results fully proved that the DE strategies are helpful for searching better functional module partitions.

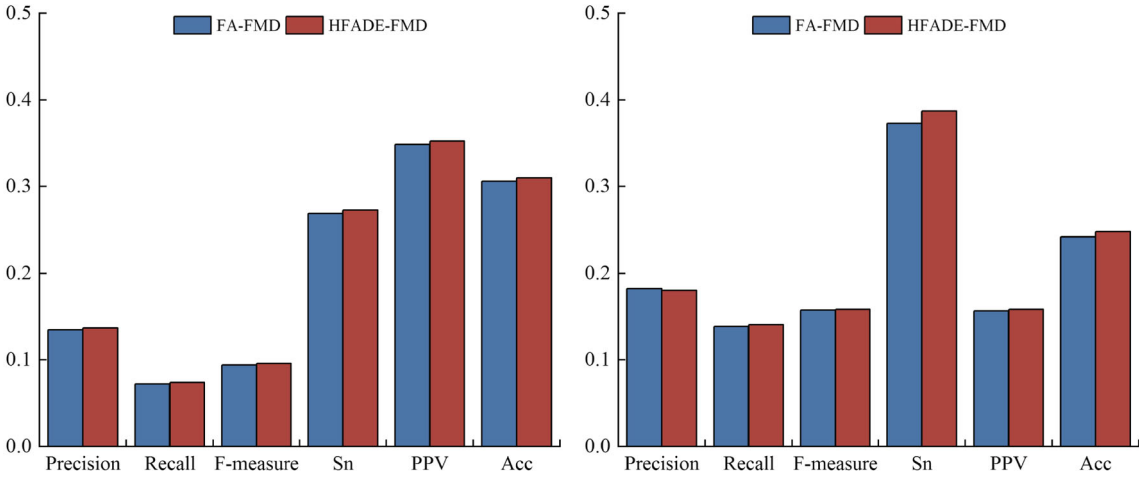
3.4 Comparing with competitive methods

To demonstrate the performance of HFADE-FMD, the authors compared it with ten competitive methods: MCODE [4], CFinder [5], DCAFP [6], NeMo [7], FAG-EC [9], MTGO [11], GA-PPI [28], ClusterEPs [18], ACC-FMD [30] and BFO-FMD [31]. Among them, GA-PPI, ACC-FMD and BFO-FMD are three swarm intelligence and



(a) using SGD as gold standard functional modules (b) using CYC2008 as gold standard functional modules

Fig. 5 Comparative results of HFADE-FMD and FA-FMD in terms of various evaluation metrics on KroganCore dataset

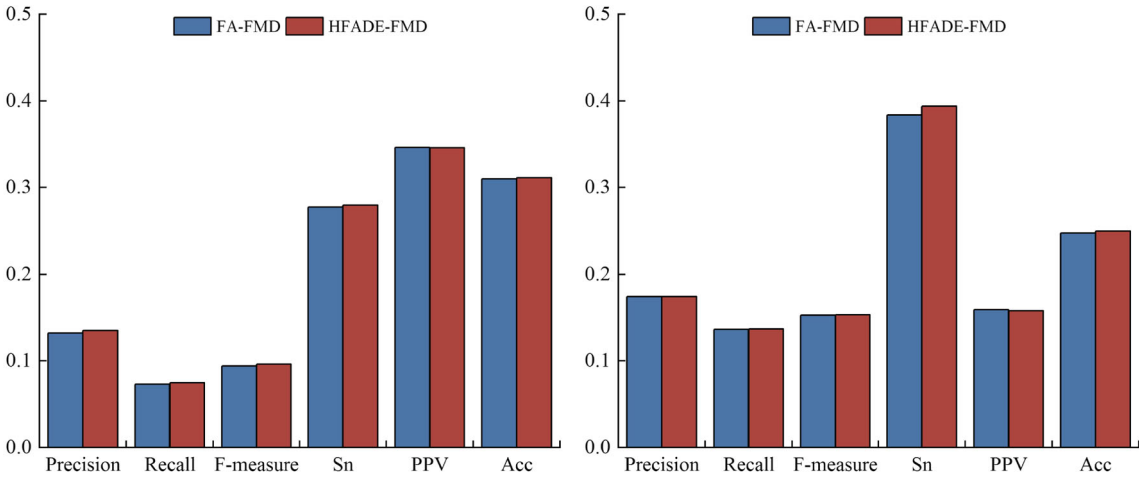


(a) using PCDq as gold standard functional modules (b) using CORUM as gold standard functional modules

Fig. 6 Comparative results of HFADE-FMD and FA-FMD in terms of various evaluation metrics on DIPCore dataset

evolution based algorithms. MCODE, CFinder and DCAFP are three density-based algorithms. According to the node's neighbor local density, MCODE fist picks out seed nodes for initial clusters, and then further augments these clusters to form the final clusters. CFinder first identifies the k-cliques using clique percolation, and then combines the adjacent k-cliques to get the functional modules. DCAFP integrates functional preferences of proteins and the graph topology of PPI network for detecting functional modules. NeMO combines a unique neighbor sharing score with the hierarchical agglomerative clustering to identify functional modules. FAG-EC is a fast agglomerate algorithm based on the edge clustering coefficients. MTGO directly exploits GO terms during the module assembling process, and labels each module with its best fit Go term. ClusterEPs is different from the above methods and is a supervised technology. Its key idea is to discover emerging patterns, which can clearly distinguish true complexes from random subgraphs in a PPI network. In the experiments, GA-PPI, ACC-FMD and BFO-FMD adopted the same parameters as [28, 30] and [31], respectively. For MCODE, CFinder, DCAFP, NeMo, FAG-EC, MTGO, and ClsterEPs, the authors obtained their software implementations, and used the default values for their parameters. The parameters of the proposed HFADE-FMD algorithm is given in the last section.

Table 4 provides the basic information of the detection results for the eleven algorithms on the four datasets. From the table, it is clear that whichever set of gold standard functional modules using, MCODE always obtains the least number of modules (Number of modules), and ACC-FMD and BFO-FMD get the biggest module and the smallest

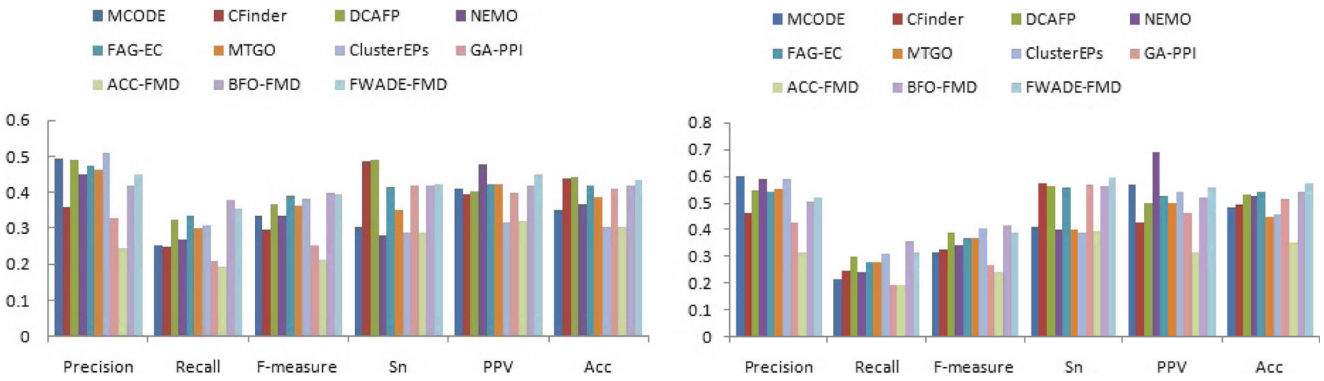


(a) using PCDq as gold standard functional modules (b) using CORUM as gold standard functional modules

Fig. 7 Comparative results of HFADE-FMD and FA-FMD in terms of various evaluation metrics on DIPFull dataset

Table 4 The basic results of eight algorithms on four datasets

Dataset	Algorithm	Number of modules	Average size of modules	SGD		CYC2008	
				$N_{ch} \geq 0.2$	$N_{cs} \geq 0.2$	$N_{ch} \geq 0.2$	$N_{cs} \geq 0.2$
Gavin	MCODE	122	8.14	60	94	73	87
	CFinder	184	9.16	66	93	85	102
	DCAFP	204	9.08	100	120	112	122
	NeMo	259	8.64	117	100	152	99
	FAG-EC	158	10.39	75	124	86	114
	MTGO	205	9.12	95	111	113	112
	ClusterEPs	286	7.58	146	114	168	126
	GA-PPI	155	11.98	51	77	66	80
	ACC-FMD	644	12.78	157	71	205	80
	BFO-FMD	213	7.16	89	141	108	145
	HFADE-FMD	194	9.56	87	132	101	128
KroganCore	MCODE	77	7.23	54	83	58	69
	CFinder	115	10.91	60	86	71	83
	DCAFP	186	11.78	101	157	92	156
	NeMO	302	7.99	104	86	117	77
	FAG-EC	245	8.65	83	117	99	122
	MTGO	192	8.12	66	112	85	121
	ClusterEPs	398	7.45	238	93	248	96
	GA-PPI	237	11.41	54	68	66	76
	ACC-FMD	328	20.89	155	102	178	95
	BFO-FMD	230	7.22	95	146	121	157
	HFADE-FMD	315	8.57	113	153	139	162
DIPCore	MCODE	90	5.32	29	32	35	101
	CFinder	243	5.93	61	63	88	214
	DCAFP	289	6.31	90	53	108	179
	NeMo	409	10.31	55	51	71	158
	FAG-EC	427	8.33	57	60	73	159
	MTGO	504	7.93	94	91	90	182
	ClusterEPs	582	7.17	48	40	127	108
	GA-PPI	570	7.64	42	45	74	131
	ACC-FMD	427	23.86	54	29	135	151
	BFO-FMD	554	5.98	70	69	110	263
	HFADE-FMD	671	6.49	92	93	121	245
DIPFull	MCODE	91	5.35	28	31	35	100
	CFinder	241	6.04	61	63	89	212
	DCAFP	296	6.34	81	30	106	215
	NeMO	431	9.38	54	50	69	155
	FAG-EC	440	8.41	60	63	75	165
	MTGO	402	7.96	65	44	84	158
	ClusterEPs	612	5.43	112	40	135	103
	GA-PPI	586	7.78	44	46	74	137
	ACC-FMD	417	24.62	59	36	129	160
	BFO-FMD	580	5.67	87	89	111	249
	HFADE-FMD	671	6.49	92	93	121	245



(a) using SGD as gold standard functional modules

(b) using CYC2008 as gold standard functional modules

Fig. 8 Comparative results of eleven methods in terms of various evaluation metrics on Gavin dataset

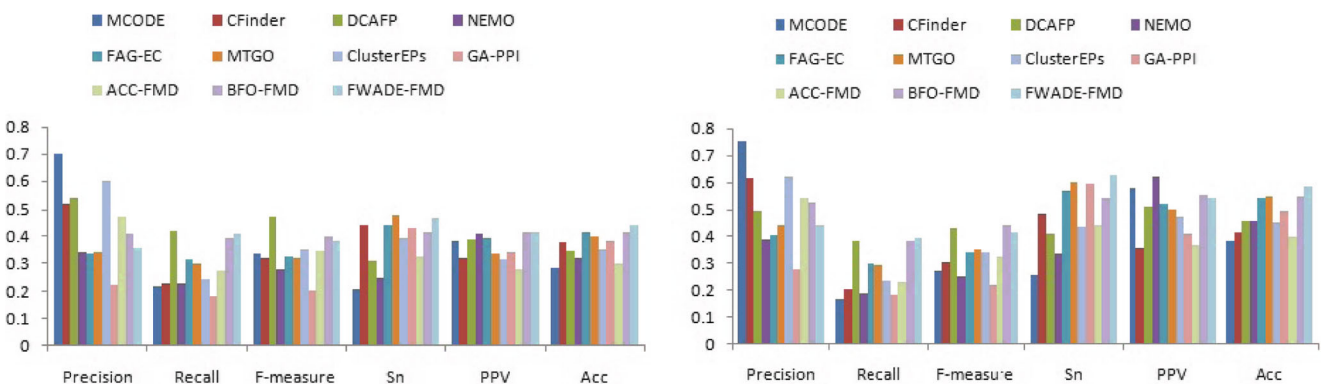
module (Average Size of modules), respectively. HFADE-FMD usually generates more and smaller modules. Besides, the authors find that the number of gold standard modules matching at least one detected module (N_{cs}) is more than all the other algorithms in most cases, and the number of detected module matching at least one gold standard module (N_{ch}) stands the top two ranks in most situations.

Figures 8, 9, 10, and 11 show the overall comparison results of eleven algorithms on four datasets in terms of six evaluation metrics including Precision, Recall, F-measure, Sensitivity, PPV, and Accuracy. For the Gavin dataset shown in Fig. 8, it is can be seen that according to the SGD benchmark, HFADE-FMD ranks second in terms of Recall, F-measure, and PPV, takes third places for Sn and ACC, and is in the fifth place on the Precision metric. Using CYC2008 as benchmark, HFADE-FMD achieves the best results on the Sn and ACC statistics, respectively takes the third and fourth places on the Recall and PPV statistics, and obtains the fourth positions on the Precision and F-measure statistics.

On KroganCore dataset in Fig. 9, One can easily finds that when using SGD as benchmark, HFADE-FMD has the best performance in terms of PPV and ACC, has the second-highest top level in term of Recall, F-measure and Sn, and ranks seventh with respect to the precision metric. Using CYC2008 as benchmark, HFADE-FMD is in the first place for Sn and ACC, and respectively ranks second, third, sixth, eighth place in terms of Recall, F-measure, PPV and Precision.

Figure 10 displays the results of the DIPCore dataset. Taking PCDq as the benchmark, HFADE-FMD is first in terms of three out of six metrics: Recall, Sn and ACC, receives the second places with respect to PPV and F-measure measures, and takes fifth place for the Precision metric. Taking CORUM as benchmark, HFADE-FMD ranks number one in regard to ACC metric, ranks second in terms of Recall, Sn and PPV, and respectively takes the third and seventh places on F-measure and Precision metrics.

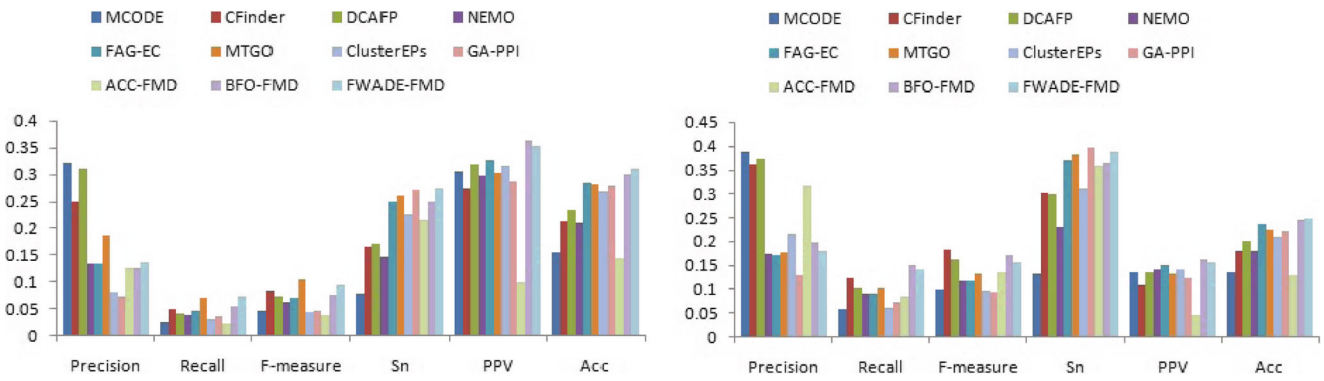
As for the remaining one dataset in Fig. 11, using PCDq as benchmark, HFADE-FMD stands first on half of metrics: Recall, F-measure and Sn, gets the next best results on PPV and ACC statistics, and ranks ninth with respect to the Precision metric. Using CORUM as benchmark, HFADE-FMD obtains the best performance on ACC metric,



(a) using SGD as gold standard functional modules

(b) using CYC2008 as gold standard functional modules

Fig. 9 Comparative results of eleven methods in terms of various evaluation metrics on KroganCore dataset



(a) using PCDq as gold standard functional modules

(b) using CORUM as gold standard functional modules

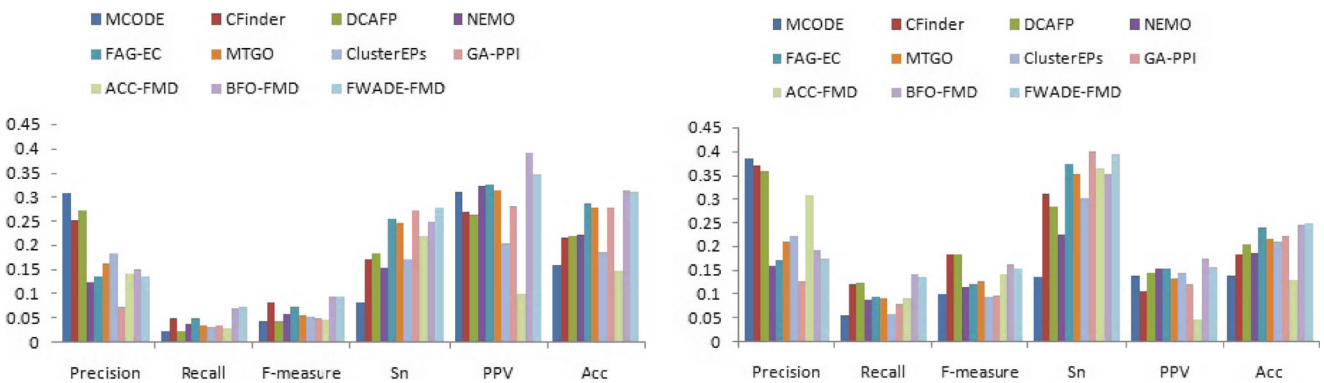
Fig. 10 Comparative results of eleven methods in terms of various evaluation metrics on DIPCore dataset

receives the second place in terms of Recall, Sn and PPV, and respectively ranks third and eighth in regard to the F-measure and Precision metrics.

According to the above analysis, no matter what PPI datasets and benchmark being used, HFADE-FMD always obtains the superior results in most evaluation metrics. Only on the Precision metric, HFADE-FMD usually does not perform well. However, as previously mentioned, the number of detected modules matching at least one gold standard module (N_{ch}) stands the top two ranks in most situations. So the reason for the bad performance with respect to Precision metric is because HFADE-FMD usually generates more functional modules according to formula (11). Besides, regardless of which PPI datasets and benchmark being used, HFADE-FMD plays better than the other three swarm intelligence and evolution based algorithms (GA-PPI, ACC-FMD and BFO-FMD) on overwhelming majority of metrics. Therefore, these experimental results fully demonstrate that HFADE-FMD is a very promising swarm intelligence and evolution based approach for functional module detection in PPI networks.

4 Conclusions

The identification of functional modules in PPI networks is important for biological knowledge discovery since many important biological processes in the cell are carried out through the formation of protein modules. Nowadays, swarm intelligence and evolutionary based approaches have been a kind of effective way to detect functional modules. In this paper, a novel hybrid approach of FA and differential evolution strategies is proposed for detecting functional modules in PPI networks (called HFADE-FMD). HFADE-FMD uses a new individual initialization method and new implementation methods of optimization mechanisms. To demonstrate the performance of HFADE-FMD, the authors have carried out a series of experiments on four datasets in terms of various evaluation metrics. The empirical results illustrate that HFADE-FMD achieves prominent performance with respective to Recall, Sn, PPV, and ACC while performing well on other metrics. Therefore, the proposed HFADE-FMD algorithm can effectively detect functional modules from PPI networks and can be



(a) using PCDq as gold standard functional modules

(b) using CORUM as gold standard functional modules

Fig. 11 Comparative results of eleven methods in terms of various evaluation metrics on DIPFull dataset

considered as a complement to help biologists to discover some biological insights.

In spite of HFADE-FMD having superior performance, there are the following two serious limitations that require further study in the future: (1) HFADE-FMD as a swarm intelligence algorithm is an iterative algorithm based on population evolution, which is time consuming. Due to parallel mechanisms are effective in reducing time complexity, thus the systematical research of the parallel algorithms will be a major task to enhance the efficiency of detecting functional modules. (2) At present most studies including this paper consider PPI networks as static graphs and overlook the inherent dynamics of PPI networks. So it it very essential to make clustering analysis on dynamics PPI networks.

Acknowledgments This work is partly supported by the NSFC Research Program (61672065, 61906010), Beijing Municipal Education Research Plan Project (KM202010005032), China Postdoctoral Science Foundation funded project (71007011201801), Beijing Postdoctoral Research Foundation (2017-ZZ-024), and Chaoyang Postdoctoral Research Foundation (2018ZZ-01-05).

References

1. Eisenberg D, Marcotte EM, Xenarios I et al (2000) Protein function in the post-genomic era. Nature 405(6788):823–826
2. Ji J, Zhang A, Liu C et al (2014) Survey: functional module detection from protein-protein interaction networks. IEEE Trans Knowl Data Eng 26(2):261–277
3. Li X, Wu M, Kwoh CK, Ng SK (2010) Computational approaches for detecting protein complexes from protein interaction networks: a survey. BMC Genom 11(1):S3
4. Bader GD, Hogue CWV (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinform 4(1):1
5. Adamcsek B, Palla G, Farkas IJ, Derényi I, Vicsek T (2006) CFinder: locating cliques and overlapping modules in biological networks. Bioinformatics 22(8):1021–1023
6. Hu L, Chan KCC (2015) A density-based clustering approach for identifying overlapping protein complexes with functional preferences. BMC Bioinform 16:174
7. Rivera CG, Vakil R, Bader JS (2010) NeMo: network module identification in cytoscape. BMC Bioinform 11(Suppl 1):S61
8. Li M, Wang J, Chen J (2008) A fast agglomerate algorithm for mining functional modules in protein interaction networks. In: Proceedings of the 1st international conference on biomedical engineering and informatics, pp 3–7
9. King AD, Pržulj N, Jurisica I (2004) Protein complex prediction via cost-based clustering. Bioinformatics 20(17):3013–3020
10. Abdullah A, Deris S, Hashim SZM, Jamil HM (2009) Graph partitioning method for functional module detections of protein interaction network. In: Proceedings of the international conference on computer technology and development (ICCTD'09), pp 230–234
11. Vella D, Marini S, Vitali F, Silvestre DD, Mauri G, Bellazzi R (2018) MTGO: PPI network analysis via topological and functional module identification. Sci Rep 8(1):5499
12. Cho YR, Hwang W, Ramanathan M, Zhang A (2007) Semantic integration to identify overlapping functional modules in protein interaction networks. BMC Bioinform 8(1):265
13. Feng J, Jiang R, Jiang T (2011) A max-flow-based approach to the identification of protein complexes using protein interaction and microarray data. IEEE/ACM Trans Comput Biol Bioinform 8(3):621–634
14. Qin G, Gao L (2010) Spectral clustering for detecting protein complexes in protein-protein interaction (PPI) networks. Math Comput Model 52(11):2066–2074
15. Inoue K, Li W, Kurata H (2010) Diffusion model based spectral clustering for protein-protein interaction networks. Plos One 5(9):e12623
16. Wu M, Li X, Kwoh CK, Ng SK (2009) A core-attachment based method to detect protein complexes in PPI networks. BMC Bioinform 10(1):169
17. Ma X, Gao L (2012) Predicting protein complexes in protein interaction networks using a core-attachment algorithm based on graph communicability. Inf Sci 189:233–254
18. Liu Q, Song J, Li J (1223) Using contrast patterns between true complexes and random subgraphs in PPI networks to predict unknown protein complexes. Sci Rep 6(2):2016
19. Yu FY, Yang ZH, Tang N, Lin HF, Wang J, Yang ZW (2014) Predicting protein complex in protein interaction network-a supervised learning based method. BMC Syst Biol 8(3):S4
20. Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Studies in Computational Intelligence. https://doi.org/10.1007/978-3-030-10674-4
21. Fahy C, Yang S, Gongora M (2019) Ant colony stream clustering: a fast density clustering algorithm for dynamic data streams. IEEE Trans Cybern 49(6):2215–2228
22. Abualigah L, Khader A, Hanandeh E (2018) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48(11):4047–4071
23. Maulik U, Saha I (2010) Automatic fuzzy clustering using modified differential evolution for image classification. IEEE Trans Geosci Remote Sens 48(9):3503–3510
24. Abualigah LM, Khader AT, Hanandeh ES (2018) A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. Eng Appl Artif Intell 73:111–125
25. Revathi J, Eswaramurthy VP, Padmavathi P (2019) Bacterial colony optimization for data clustering. In: IEEE international conference on electrical, computer and communication technologies (ICECCT)
26. Abualigah LM, Khader AT, Hanandeh ES, Gandomi AH (2017) A novel hybridization strategy for krill herd algorithm applied to clustering techniques. Appl Soft Comput 60:423–435
27. Sallim J, Abdullah R, Khader AT (2008) ACOPIN: an ACO algorithm with TSP approach for clustering proteins from protein interaction network. In: Proceedings of second UKSIM European symposium on computer modeling and simulation, pp 203–208
28. Pizzuti C, Rombo S (2012) Experimental evaluation of topological-based fitness functions to detect complexes in PPI networks. In: Proceedings of the 14th annual conference on genetic and evolutionary computation. ACM, New York, pp 193–200
29. Ji J, Liu Z, Zhang A, Yang C, Liu C (2013) HAM-FMD: mining functional modules in protein-protein interaction networks using ant colony optimization and multi-agent evolution. Neurocomputing 121:453–469
30. Ji J, Liu H, Zhang A, Liu C (2015) ACC-FMD: ant colony clustering for functional module detection in protein-protein interaction networks. Int J Data Min Bioinform 11(3):331–363
31. Yang C, Ji J, Zhang A (2018) BFO-FMD: bacterial foraging optimization for functional module detection in protein-protein interaction networks. Soft Comput 22(10):3395–3416
32. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: Proceedings of the 1st international conference on advances in swarm intelligence. Springer, Berlin, pp 355–364

33. Bacanin N, Tuba M (2015) Fireworks algorithm applied to constrained portfolio optimization problem. In: 2015 IEEE congress on evolutionary computation (CEC), pp 1242–1249
34. Babu TS, Ram JP, Sangeetha K, Laudani A, Rajasekar N (2016) Parameter extraction of two diode solar PV model using fireworks algorithm. Solar Energy 140:265–276
35. Reddy KS, Panwar LK, Kumar R, Panigrahi BK (2016) Binary fireworks algorithm for profit based unit commitment (PBUC) problem. Int J Electr Power Energy Syst 83:270–282
36. Xue Y, Zhao B, Ma T, Pang W (2018) A self-adaptive fireworks algorithm for classification problems. IEEE Access 6:44406–44416
37. Messaoudi I, Kamel N (2019) Community detection using fireworks optimization algorithm. Int J Artif Intell Tools 28(3):1950010
38. Barraza J, Valdez F, Melin P, González C (2020) Optimal number of clusters finding using the fireworks algorithm. In: Hybrid intelligent systems in control, pattern recognition and medicine, pp 83–93
39. Barraza J, Melin P, Valdez F, Gonzalez CI (2017) Fuzzy fireworks algorithm based on a sparks dispersion measure. Algorithms 10(3):83
40. Zhang T, Yue Q, Zhao X, Liu G (2019) An improved firework algorithm for hardware/software partitioning. Appl Intell 49(3):950–962
41. Yu C, Li J, Tan Y (2014) Improve enhanced fireworks algorithm with differential mutation. In: 2014 IEEE international conference on systems man, and cybernetics (SMC), pp 264–269
42. Zheng YJ, Xu XL, Ling HF, Chen SY (2015) A hybrid fireworks optimization method with differential evolution operators. Neurocomputing 148:75–82
43. Guo J, Liu W, Liu M, Zheng S (2019) Hybrid fireworks algorithm with differential evolution operator. Int J Intell Inf Database Syst 12(1-2):47–64
44. Zhu X, Liu C, Guo Y (2015) Design of fuzzy classification system based on fireworks optimization and differential evolution algorithm. J Zhengzhou Univ (Eng Sci) 36(6):47–51
45. Ochoa P, Castillo O, Soria J (2020) Optimization of fuzzy controller design using a differential evolution algorithm with dynamic parameter adaptation based on Type-1 and Interval Type-2 fuzzy systems. Soft Comput 24(1):193–214
46. Lien LC, Cheng MY (2012) A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization. Expert Syst Appl 39(10):9642–9650
47. Castillo O, Melin P, Valdez F, Soria J, Ontiveros-Robles E, Peraza C, Ochoa P (2019) Shadowed type-2 fuzzy systems for dynamic parameter adaptation in harmony search and differential evolution algorithms. Algorithms 12(1):17
48. Aydin ME, Kwan R, Leung C, Maple C, Zhang J (2013) A hybrid swarm intelligence algorithm for multiuser scheduling in HSDPA. Appl Soft Comput J 13(5):2990–2996
49. Chen CH, Su MT, Lin CJ, Lin CT (2014) A hybrid of bacterial foraging optimization and particle swarm optimization for evolutionary neural fuzzy classifier. Int J Fuzzy Syst 16(3):422–433
50. Castillo O, Valdez F, Soria J, Amador-Angulo L, Ochoa P, Peraza C (2019) Comparative study in fuzzy controller optimization using bee colony, differential evolution, and harmony search algorithms. Algorithms 12(1):9
51. Grosan C, Abraham A, Han S, Gelbukh A (2005) Hybrid particle swarm–evolutionary algorithm for search and optimization. Adv Artif Intell 3789:623–632
52. Zuo L, Liu L, Wang H, Tan L (2018) A hybrid differential evolution algorithm and particle swarm optimization with alternative replication strategy. Lect Notes Comput Sci 10941:487–497
53. Zhang SH, Ning XM, Ding C, Zhang XS (2010) Determining modular organization of protein interaction networks by maximizing modularity density. BMC Syst Biol 4(2):S10–S21
54. Mete M, Tang F, Xu X, Yuruk N (2008) A structural approach for finding functional modules from large biological networks. BMC Bioinform 9(9):S19
55. Schlicker A, Albrecht M (2008) FunSimMat: a comprehensive functional similarity database. Nucleic Acids Res 36(suppl 1):D434–D439
56. Stroin R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. Int J Glob Optim 11:341–369
57. Cherry JM, Adler C, Ball C et al (1998) SGD: saccharomyces genome database. Nucleic Acids Res 26(1):73–79
58. Pu S, Wong J, Turner B et al (2009) Up-to-date catalogues of yeast protein complexes. Nucleic Acids Res 37(3):825–831
59. Kikugawa S, Nishikata K, Murakami K et al (2012) PCDq: human protein complex database with quality index which summarizes different levels of evidences of protein complexes predicted from H-Invitational protein-protein interactions integrative dataset. BMC Syst Biol 6(S2):S7
60. Ruepp A, Waegele B, Lechner M et al (2010) CORUM: the comprehensive resource of mammalian protein complexes-2009. Nucleic Acids Res 38(1):D497–D501

Junzhong Ji received the PhD degree in computer science and application technology from the Beijing University of Technology. He is a professor and PhD supervisor in the Computer Science college, Beijing University of Technology, Member of Chinese Association for Artificial Intelligence, and Senior member of the China Computer Federation. He was a visiting scholar at Norwegian University and the State University of New York ab Buffalo. His research interests include data mining, machine learning, swarm intelligence and bioinformatics.

Hanghang Xiao received the master degree in computer science and application technology from Beijing University of Technology in 2017. His research interests include swarm intelligence and bioinformatics.

Cuicui Yang Lecture at Beijing University of Technology. She received her doctor degree in computer science and application technology from Beijing University of Technology in 2017. Her research interest covers machine learning, computational intelligence, bioinformatics, and brain science.