Soft Comput (2018) 22:3395–3416
https://doi.org/10.1007/s00500-017-2584-9

METHODOLOGIES AND APPLICATION

BFO-FMD: bacterial foraging optimization for functional module detection in protein–protein interaction networks

Cuicui Yang[1] · **Junzhong Ji**[1] · **Aidong Zhang**[2]

Published online: 4 April 2017

Abstract Identifying functional modules in PPI networks contributes greatly to the understanding of cellular functions and mechanisms. Recently, the swarm intelligence-based approaches have become effective ways for detecting functional modules in PPI networks. This paper presents a new computational approach based on bacterial foraging optimization for functional module detection in PPI networks (called BFO-FMD). In BFO-FMD, each bacterium represents a candidate module partition encoded as a directed graph, which is first initialized by a random-walk behavior according to the topological and functional information between protein nodes. Then, BFO-FMD utilizes four principal biological mechanisms, chemotaxis, conjugation, reproduction, and elimination and dispersal to search for better protein module partitions. To verify the performance of BFO-FMD, we compared it with several other typical methods on three common yeast datasets. The experimental results demonstrate the excellent performances of BFO-FMD in terms of various evaluation metrics. BFO-FMD achieves outstanding Recall, F-measure, and PPV while performing very well in terms of other metrics. Thus, it can accurately predict protein modules and help biologists to find some novel biological insights.

Keywords Computational biology · Protein–protein interaction network · Functional module detection · Bacterial foraging optimization

Communicated by V. Loia.

Electronic supplementary material The online version of this article (doi:10.1007/s00500-017-2584-9) contains supplementary material, which is available to authorized users.

✉ Junzhong Ji
jjz01@bjut.edu.cn

Cuicui Yang
yangcc_2008@163.com

Aidong Zhang
azhang@buffalo.edu

[1] Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology, College of Computer Science and Technology, Faculty of Information Technology, Beijing University of Technology, Beijing, China

[2] Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY, USA

1 Introduction

In the post-genomic era, proteomics research has become one of the most important areas in the life science (Ji et al. 2014a), where the analysis of protein–protein interactions (PPI) is fundamental to the understanding of cellular organization, processes, and functions (Zhang 2009). Indeed, biologists also reveal that cellular functions and biochemical events are coordinately carried out by groups of interacting proteins in functional modules (Guimera and Amaral 2005). Thus, identifying functional modules in PPI data contributes greatly to the understanding of cellular functions and mechanisms.

Nowadays, rapid advance in experimental technologies has led to an explosive growth in binary PPI data (Chin and Zhu 2013), but it is getting hard to detect functional modules only using experimental approaches, which have several limitations, such as too many processing steps and too time-consuming (Li et al. 2010; Tarassov et al. 2008). Fortunately, computational approaches based on machine learning and data mining can overcome these shortcomings to some extent and have become useful complements to the experimental methods for detecting functional modules in PPI data. In general, these computational approaches first

model PPI data as a network (graph) $G = (V, E)$, where V is the set of nodes with each node represents a protein and E is the set of edges with each edge represents an interaction (link). Then various clustering technologies are used to divide the network and get corresponding protein module partitions. So far, there have been many kinds of computational methods which adopt different ideas and schemes to mine functional modules. All these methods can be systematically categorized into two main groups (Ji et al. 2014a, b): traditional graph–theoretic approaches and non-traditional graph–theoretic approaches. The traditional graph–theoretic approaches mainly make an analysis on the topology structure of a PPI network and employ some traditional clustering technologies to identify functional modules, such as density-based clustering (Bader and Hogue 2003; Adamcsek et al. 2006; Altaf-Ul-Amin et al. 2006), hierarchy-based clustering (Ravasz et al. 2002; Arnau et al. 2005; Aldecoa and Marín 2010) and partition-based clustering (King et al. 2004; Frey and Dueck 2007; Abdullah et al. 2009). The non-traditional graph–theoretic approaches refer to the new emerging methods in recent years, which make use of novel computational technologies to detect functional modules in a PPI network. This type of approaches mainly includes flow simulation-based methods (Dongen 2000; Cho et al. 2007; Feng et al. 2011), spectral clustering-based methods (Sen et al. 2006; Qin and Gao 2010; Inoue et al. 2010), core attachment-based approaches (Leung et al. 2009; Wu et al. 2009; Ma and Gao 2012), swarm intelligence-based approaches and so on. Among them, swarm intelligence technology is a population-based metaheuristic approach, which has been increasingly popular due to its ability in solving a variety of complex scientific and engineering problems (Hinchey et al. 2007). Such technology simulates the social behavior of certain living creature and has been applied in functional module detection in PPI networks. For example, Sallim et al. (2008) for the first time introduced ant colony optimization (ACO) into functional module detection by taking it as the optimization problem of solving traveling salesman problem (TSP). Ji et al. (2012a) proposed a new algorithm based on ACO (called as NACO-FMD), which incorporates known biological knowledge—Gene Ontology (GO), into the process of detecting functional modules. To further overcome the shortcoming of easily trapped into a local optima for ACO algorithm, they combined ACO with multi-agent evolutionary (MAE) to identify functional modules (Ji et al. 2012b, 2013). Subsequently, they again presented an algorithm based on ant colony clustering (called as ACC-FMD) for functional module detection in PPI networks (Ji et al. 2015). Moreover, by uniting with functional flow clustering, artificial bee colony (ABC) algorithm was likewise used to identify functional modules in PPI networks (Wu et al. 2011).

Bacterial foraging optimization (BFO) (Passino 2002) is another famous swarm intelligence algorithm developed by Passino in 2002, which simulates the foraging behavior of *Escherichia coli* bacteria. The basic principle is that bacteria move through either tumbling or swimming to maximize the energy consumed by eating as many nutrients as they can. As the smallest creatures on the earth, bacteria contain many clever optimization mechanisms. Thus, BFO which is inspired by bacterial biological behavior has unique good performance and has been successful in a wide variety of optimization tasks (Das et al. 2009), and also found its way in the domain of protein functional module detection (Lei et al. 2011, 2013). However, there are two basic issues in Lei et al. (2011, 2013): (1) Each bacterium represents one protein node, and this individual representation is unable to realize information exchange among bacterial individuals, but information exchange among different individuals is a crucial mechanism for a swarm intelligence optimization algorithm. (2) Although three key operators including chemotaxis, reproduction, and elimination and dispersal are utilized, they are too simple to be in conformity with the biological mechanisms simulated in the original BFO algorithm. Thus, the above two basic issues make it difficult to fully release the potential of BFO algorithm for functional module detection in PPI networks. To explore the full potential of BFO algorithm in identifying functional modules, this paper conducts a further and thorough investigation and proposes a new bacterial foraging optimization algorithm for functional module detection in PPI networks, called BFO-FMD. In comparison with the previous work in Lei et al. (2011, 2013), the main contributions of this paper are summarized as follows:

- The proposed algorithm employs a completely different representation of bacterial individual. Each bacterial individual represents a candidate module partition, which is easy to exchange information among different bacterial individuals.
- In the proposed algorithm, the realization of four optimization mechanisms more faithfully follows the biological mechanisms of bacteria.
- Systematic experiments have been conducted to compare the proposed algorithm with different kinds of state-of-the-art algorithms on three yeast PPI datasets using seven evaluation metrics.

2 BFO-FMD algorithm

2.1 Basic idea

By considering functional module detection as an optimization problem, this study uses BFO algorithm to solve the optimization problem and proposes a new algorithm called BFO-FMD. In BFO-FMD, each bacterial individual corre-

sponds to a candidate module partition (i.e., a candidate solution), and is evaluated using the modularity metric, which is commonly used in the partition of a network into modules (Guimera and Amaral 2005), and is given below:

$$J(\theta) = \sum_{i=1}^{M} \left[\frac{e_i}{|E|} - \left(\frac{d_i}{2|E|} \right)^2 \right], \tag{1}$$

where θ is a candidate module partition represented by a bacterium, M is the number of modules for a bacterium θ, e_i is the number of links between nodes in the ith module, d_i is the sum of the degrees of nodes in the ith module, and $|E|$ is the number of all the links in the PPI network. In general, the modularity metric tends to favor the detected results which comprise many within-module links and as few as possible between-module links. The higher the modularity score, the better the detected result (i.e., the better the corresponding individual solution).

The procedure of BFO-FMD includes three stages: solution initialization, solution optimization, and post-processing. At first, BFO-FMD constructs each bacterial individual solution using a random-walk behavior. Then, it optimizes iteratively each individual solution using four biological mechanisms (chemotaxis, conjugation, reproduction, and elimination and dispersal) to look for superior solutions with higher modularity scores. At last, BFO-FMD carries out two post-processing steps to further refine the preliminary module partition detected. In the following, we will explain key parts of this algorithm in detail.

2.2 Solution representation and initialization

For swarm intelligence-based algorithms, it is essential to design appropriate solution representation and the initialization method for different problems. To address the problem of detecting functional modules in PPI networks, BFO-FMD makes each bacterial individual correspond to a candidate module partition and encodes it as a graph with N directed edges: $\theta = \{ (1 \rightarrow a_1), (2 \rightarrow a_2), \ldots, (i \rightarrow a_i), \ldots, (N \rightarrow a_N) \}$, where i is a node label, a_i denotes the node label that the node i points to, and N is the number of nodes in a PPI network, i.e., $N = |V|$. In the initialization process, a bacterium first randomly selects a starting node and then continuously makes use of a random-walk behavior to traverse other nodes in the PPI network. At each step, the bacterium is on a node, tries to move to a similar node which is functionally correlated or structurally similar to the current node, and then builds a link. Clearly, the greater the similarity between two nodes of a link, the stronger the connection strength. When there is no any satisfied node, the bacterium ends its current pathway by pointing to itself, picks up another untraversed node in a PPI network, and begins to a new traversal. This random-walk behavior continues until all the N nodes in a PPI network are traversed. During the initialization, each bacterium keeps on walking to a random similarity node with its current node step-by-step. For node i, its similarity node is selected randomly from the following set:

$$\Omega_i = \left\{ j | \frac{(s_{ij} + f_{ij})}{2} > \varepsilon \right\}, \tag{2}$$

where ε represents an overall similarity threshold, s_{ij} and f_{ij} denote the structural similarity and functional similarity between nodes i and j, respectively.

Given two protein nodes i and j, the structural similarity formula is given as follows (Mete et al. 2008):

$$s_{ij} = \frac{|\Gamma(i) \bigcap \Gamma(j)|}{\sqrt{|\Gamma(i)||\Gamma(j)|}}, \tag{3}$$

where $\Gamma(i)$ is a set of the neighborhood nodes of node i plus itself, and $|\Gamma(i)|$ is the size of the set.

Based on the annotation information of Gene Ontology (GO), the functional similarity formula is expressed as follows (Schlicker and Albrecht 2008):

$$f_{ij} = \frac{|g^i \bigcap g^j|}{|g^i \bigcup g^j|}, \tag{4}$$

where g^i and g^j are GO term sets of nodes i and j, respectively.

To clearly show the solution representation and initialization process, Fig. 1 illustrates an example in a simplified PPI network with ten nodes. Figure 1a shows a PPI network containing ten nodes marked from 0 to 9, Fig. 1b gives the generation process of a bacterium denoting an individual solution, where this bacterium first picks out node 3 randomly, then it moves to node 4 which is selected out from the set in Eq. (2) at random and builds a link $3 \rightarrow 4$. Next, it moves to node 8 from node 4 and builds a link $4 \rightarrow 8$ in the same way. This process is repeated until it reaches node 0 when there is not any node meeting the condition in Eq. (2), and the bacterium has to end its current pathway by pointing to itself $0 \rightarrow 0$. Afterward, this bacterium randomly chooses another unvisited node 2 and restarts a new pathway until all the ten nodes are traversed. Figure 1c gives the solution presentation by adjusting the sequence in Fig. 1b according to the node label, and Fig. 1d shows the detected modules corresponding to the individual solution shown in Fig. 1c. It is obvious, such initialization method does not need a given number of modules in advance, and it automatically makes decision according to the links of a bacterial individual.

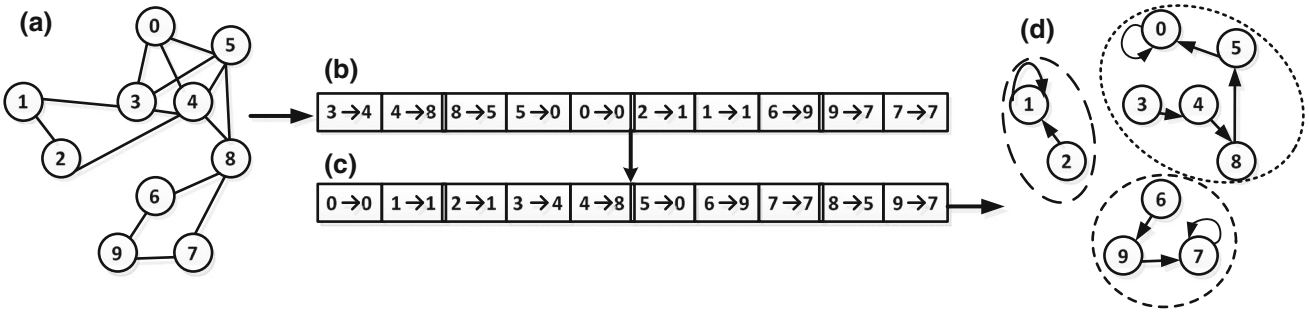

Fig. 1 Bacterial individual representation and initialization process: **a** PPI network, **b** initialization process, **c** individual representation, and **d** an initial solution represented by a bacterial individual

Operation function: Chemotaxis_Process(i)

1 Compute the modularity score of solution θ_i according to Eq. (1), noted as $J(\theta_i)$, and let $\theta_{last} = \theta_i$ and $J_{last} = J(\theta_i)$.
2 **Tumble:** generate randomly a direction vector D.
3 **Move:** select another bacterium a, when the component in D is 1, bacterium i will replace the corresponding link having weaker strength with that of bacterium a, then compute the new modularity score of bacterium i, noted $J(\theta_i)$.
4 **Swim as follows:**
i) Let $m = 0$ (the swimming counter).
ii) While $m < N_s$
- Let $m = m + 1$.
- If $J(\theta_i) > J_{last}$, let $\theta_{last} = \theta_i$ and $J_{last} = J(\theta_i)$, and then keep on the move according to step 3 and compute new $J(\theta_i)$.
- Else, let $m = N_s$, $\theta_i = \theta_{last}$ and $J(\theta_i) = J_{last}$.

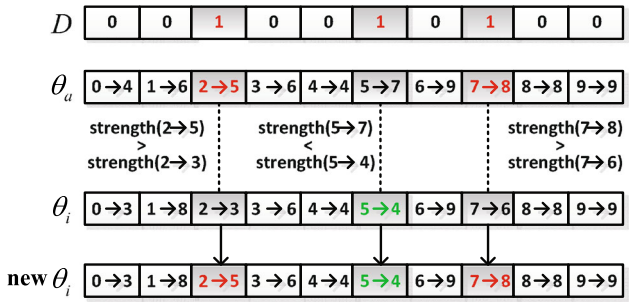

Fig. 2 Method of generating a new solution in chemotaxis operator

2.3 Chemotaxis

This mechanism simulates the foraging movement of *E.colis* through tumbling and swimming. To absorb more nutrients, each bacterium tries to find food in two ways: tumbling and swimming. A bacterium tumbles in a random direction to exploratively search for food. If the food is rich in the selected direction, the bacterium will swim along this direction, till the food gets bad or the bacterium has swum the fixed steps.

From the above description, a proper random direction generated through tumbling and an reasonable way of updating a solution are important for the realization of the chemotaxis mechanism. In BFO-FMD, a random direction is defined as a random masking vector with N dimensions, and the value of each dimension is either 0 or 1 with a certain probability P_{ch}. That is, for each dimension of a masking vector, if a uniform random number in the range of (0,1) is smaller than P_{ch}, its value will be 1, otherwise it will be 0. Thus, when the bacterial population performs a chemotaxis step, each bacterium first generates an aforesaid masking vector denoted by D (equivalent to tumbling). Then, it updates its current solution according to the following method: select randomly a different bacterium and check the value of each dimension in D one by one. When the value is 1, the bacterium will compare the connection strength of the corresponding link with the selected bacterium. If its connection strength is weaker than that of the selected bacterium, the bacterium will replace its corresponding link with that of the selected bacterium. After the bacterium checks each component of D, it gets a new solution. The next step is to compute the modularity score of the new solution according to Eq. (1). If the modularity score of the new solution is higher than that of the old one, the bacterium will continue to optimize the new solution along the direction D in the same way (equivalent to swimming), till the modularity score of the new solution does not improve any more or the bacterium has carried out the maximum number of times N_s. We summarize the chemotactic process of a bacterium described above into a function **Chemotaxis_Process(i)**, where i is a parameter showing that the ith bacterium performs the chemotactic process. For such chemotaxis mechanism, tumbling controls the direction of searching for better module partitions. Swimming is a driving force toward better module partitions when a bacterial individual finds a right direction. The chemotaxis mechanism is a complex and close combination of swimming and tumbling that keeps bacteria in these places with higher modularity scores for module partitions and plays a crucial role in searching for the module partition with the highest modularity scores.

For clarity, Fig. 2 illustrates the method of generating a new solution in the chemotaxis operator, where θ_i is a

candidate solution represented by bacterium i which performs a chemotaxis operator, $D = (0, 0, 1, 0, 0, 1, 0, 1, 0, 0)$ denotes the chemotaxis direction, and θ_a is another different solution represented by bacterium a. The 3rd, 6th, and 8th components are 1 in D, so bacterium i compares its 3rd link ($2 \rightarrow 3$), 6th link ($5 \rightarrow 4$), 8th link ($7 \rightarrow 6$) with those of bacterium a ($2 \rightarrow 5$, $5 \rightarrow 7$ and $7 \rightarrow 8$), respectively. Assume that the connection strength of 3rd link ($2 \rightarrow 3$) and 8th link ($7 \rightarrow 6$) in solution θ_i is weaker than those of θ_a ($2 \rightarrow 5$ and $7 \rightarrow 8$), bacterium i replaces its 3rd link $2 \rightarrow 3$ and 8th link $7 \rightarrow 6$ with $2 \rightarrow 5$ and $7 \rightarrow 8$, respectively. Accordingly, bacterium i obtains a new solution.

2.4 Conjugation

Conjugation, as well as chemotaxis, is an important biological characteristic of bacteria. A bacterial conjugation is the transfer of part of plasmid (genetic material) from donor bacteria to recipient bacteria by a directly physical contact and is often regarded as the sexual reproduction or mating between bacteria. Some researchers have taken the bacterial conjugation as a message passaging mechanism in their work (Perales-Graván and Lahoz-Beltra 2008; Balasubramaniam and Lio 2013).

In this paper, we also simulate the bacterial conjugation behavior to play a part of information exchange between individuals in the proposed BFO-FMD algorithm. To model this biological behavior, we define conjugation probability P_{co} and conjugation length L ($L < N$) which decides the amount of changes to links of a module partition. When a bacterium carries out a conjugation step with a given probability P_{co}, it first randomly selects another different bacterium and a conjugation point Bt ($Bt < N - L$), then the bacterium replaces its links with those of the selected bacterium from Btth dimension to $(Bt + L)$th dimension. At last, the bacterium gets a new solution and will keep the superior one after comparing the new solution and the old solution. We summarize the conjugation process of a bacterium introduced above into a function **Conjugation_Process(i)**, where i is a parameter showing that the ith bacterium performs the conjugation process. With the new conjugation mechanism, the bacterial population has a strong information exchange channel. Each bacterium in the population no longer takes action blindly but searches for the module partition with the highest modularity score under the help of the other bacteria. At the same time, this new conjugation operator can also be considered as a kind of local optimization operator which is different from the chemotaxis operator. The operator improves each individual solution by changing some continuous links, while the chemotaxis operator enhances each solution by altering some discontinuous links. Although the two operators work through different mechanisms, they are

Operation function: Conjugation_Process(i)

1 Generate a random number r in (0,1)
2 If $r < P_{co}$
i) Randomly select a conjugation point Bt ($Bt < N - L$).
ii) Pick out another different bacterium b.
iii) Bacterium i acquires some information of bacterium b in the following way:
- Let $\theta_{old} = \theta_i$ and $J_{old} = J(\theta_i)$,
- Bacterium i replaces the corresponding links with those of bacterium b from Btth dimension to $(Bt + L - 1)$th dimension.

iv) Compute new modularity score $J(\theta_i)$.
v) If $J(\theta_i) < J_{old}$, let $\theta_i = \theta_{old}$ and $J(\theta_i) = J_{old}$.

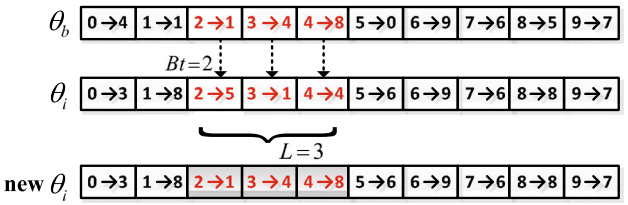

Fig. 3 Method of generating a new solution in conjugation operator

supplement each other to look for the better partition with higher modularity score.

To be more clear, Fig. 3 displays the way of generating a new solution, where θ_i is a candidate solution represented by bacterium i which performs a conjugation operator, θ_b is another different candidate solution represented by bacterium b which is selected randomly from the population. $Bt = 2$ is the conjugation point, and $L = 3$ is the conjugation length. Bacterium i, respectively, replaces three links $2 \rightarrow 5, 3 \rightarrow 1$, and $4 \rightarrow 4$ in solution θ_i with $2 \rightarrow 1, 3 \rightarrow 4$, and $4 \rightarrow 8$ in solution θ_b. Accordingly, bacterium i obtains a new solution.

2.5 Reproduction

The bacteria grow longer with the increase in absorption of nutrients. The more nutrients a bacterium gets, the healthier it is. Under appropriate conditions, some bacteria in a population which are healthy enough will asexually split into two bacteria, and the other ones will die. In the simulation of reproduction, the modularity score of a bacterium is used to measure its health degree, the higher the modularity score of a bacterium, the healthier it is. We define the number of chemotactic steps N_c to be the length of the lifetime of each bacterium, and S_r, that is half of the population size S, be the number of bacteria who are healthy enough to make a copy of themselves. After N_c chemotactic steps, a reproduction step is taken. First, sort the bacterial population in a descending order on the basis of the modularity score. Each

of the first S_r bacteria with higher modularity score split into exactly two same bacteria, and the other S_r bacteria will die, which maintains a constant size of the population. In essence, this mechanism represents a fairly abstract model of Darwinian evolution and biological genetics in genetic algorithms, which can pass the elite information among swarm individuals, and speed up the convergence.

2.6 Elimination and dispersal

When the local environment where a population of bacteria live changes, all the bacteria in this region may be killed or a group of bacteria may be dispersed into a new environment to find good food sources. To simulate this phenomenon, an elimination and dispersal step is evoked after N_{re} reproduction steps. Each bacterium in the population may be eliminated and dispersed into a new location with a given probability P_{ed}. The rule is shown in the following:

$$\theta = \begin{cases} \theta_{\text{new}}, & \text{if } q < P_{\text{ed}} \\ \theta, & \text{otherwise} \end{cases}, \tag{5}$$

where q is a random number uniformly distributed in [0, 1], θ is the current solution associated with a bacterium, θ_{new} is a new random solution generated by re-initialization using a random-walk behavior. That is, for each bacterium, if the number generated randomly is smaller than P_{ed}, it will move to a new random solution; otherwise, it will keep the original solution unchanged. This mechanism generates new random solutions for some bacteria and makes these bacteria search from new starting points. Essentially, this mechanism is quite similar to macromutation in the classical evolutionary algorithms, and is important in exploring new search regions over the whole search space, which helps the bacterial population to jump out of local optima.

2.7 Post-processing

When the iterative loops terminate, we obtain a bacterial individual with the largest modularity score. The bacterial individual can be decoded into a set of clusters, and each cluster corresponds to a protein module. Therefore, the preliminary modules are generated through the bacterial foraging optimization process. To further improve the detection quality, two post-processing steps are adopted to refine the preliminary modules.

The first step is to merge preliminary modules in light of functional similarity based on the annotation information of GO. A merging module comes from two preliminary modules which are close in function. Specifically, iteratively merge two modules with the largest functional similarity until there are no such two modules whose functional similarity is larger than the merging threshold λ. The functional similarity between two modules h_1 and h_2 is defined as:

$$S(h_1, h_2) = \frac{\sum_{i \in h_1, j \in h_2} S(i, j)}{\min(|h_1|, |h_2|)}, \tag{6}$$

where

$$S(i, j) = \begin{cases} 1 & \text{if } i = j \\ f_{ij} & \text{if } i \neq j, \text{and } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

The second step is a simple filtering operator, and its goal is to exclude some sparsely connected modules from the topological structure perspective. To do this, first compute the density of each module and then remove the module whose density is smaller than the predetermined filtering threshold δ. The density of a module is given by (Li et al. 2010):

$$D_h = \frac{e_h}{n_h \cdot (n_h - 1)/2}, \tag{8}$$

where n_h is the number of nodes and e_h is the number of interactions in module h.

By the above two post-processing strategies, the preliminary modules are further refined in the view of functional similarity and topological structure, and the final functional modules are obtained.

2.8 Algorithm description

According to the previously detailed introduction, the pseudocode of the proposed BFO-FMD algorithm is given in Algorithm 1. This algorithm first starts with an initial bacterial population, each of which represents a candidate module partition and is constructed by a random-walk behavior. Then, it executes a triple loop: chemotaxis, reproduction, and elimination and dispersal loops. At each chemotaxis loop, each bacterium in a population finds a new solution through successively carrying out two operators chemotaxis and conjugation and makes a choice between the current solution and the new solution by a greedy selection strategy. In other words, if the modularity score of the new solution is higher than that of the current one, the bacterium will choose the new solution; otherwise, it will keep the current one. After N_c chemotaxis loops, the bacterial population performs a reproduction loop. This process selects S_r bacteria with higher modularity scores and splits each of them into two bacteria while rejecting the other S_r bacteria with lower modularity scores. When N_{re} reproduction loops have been carried out, the bacterial population starts an elimination and dispersal loop. For each bacterium, if a number randomly generated between 0 and 1 is less than the given probability P_{ed}, it

Algorithm 1: The proposed BFO-FMD algorithm

Input: a PPI network $G(V, E)$
Output: the set of modules H
1 **Initialization:**
a) Set parameters: S, N_s, N_c, N_{re}, N_{ed}, P_{ed}, P_{co}, P_{ch}, L, ε, λ, δ
S: population size of the bacterial colony,
N_s: maximum number of swimming,
N_c: maximum number of chemotaxis,
N_{re}: maximum number of reproduction,
N_{ed}: maximum number of elimination and dispersal,
P_{ed}: the probability of performing elimination and dispersal,
P_{co}: the probability of performing conjugation,
P_{ch}: the probability used in generating a masking vector,
L: the conjugation length,
ε: the similarity threshold between nodes,
λ: the merging threshold,
δ: the filtering threshold.
b) Initialize the bacterial population:
For $i = 1$ to S:
do the initialization process as described in section **Solution representation and its initialization**, and get ith candidate solution represented by bacterium i.
c) Let j = k = l = 0 (three counters), and let the best solution $\theta_{best} = \theta_1$, the corresponding modularity score $J_{best} = J(\theta_1)$.
2 **Elimination and dispersal loop:** $l = l + 1$
3 **Reproduction loop:** $k = k + 1$
4 **Chemotaxis loop:** $j = j + 1$
5 **a)** For bacterium $i = 1, 2, \ldots, S$, perform the following two operators:
6 **b)** Bacterium i takes a chemotaxis operator according to **Chemotaxis_Process**(i),
7 **c)** Bacterium i takes a conjugation operator according to **Conjugation_Process**(i).
8 **d)** If $J(\theta_i) > J_{best}$, then let $\theta_{best} = \theta_i$ and $J_{best} = J(\theta_i)$.
9 **e)** If $i < S$, go to step 5.
10 If $j < N_c$, go to step 4.
11 **Reproduction:**
a) Sort bacteria based on $J(\theta)$ in descending order,
b) Abandon S_r bacteria with lower $J(\theta)$ and split each of another S_r bacteria into two ones.
12 If $k < N_{re}$, go to step 3.
13 **Elimination and dispersal:**
For each bacterium $i = 1, 2, \ldots, S$ may eliminate and disperse with a probability P_{ed} according to Eq. (5).
14 If $l < N_{ed}$, go to step 2.
15 **Take two post-processing steps** on θ_{best}.
16 Return the the set of modules H.

will be eliminated and re-initialized. After the triple loop, two post-processing operators are performed to further refine the obtained best solution. In essence, chemotaxis, conjugation, and elimination and dispersal have responsibilities to keep the balance between exploitation and exploration. BFO-FMD performs exploitation processes on each candidate solution using chemotaxis and conjugation mechanisms, and exploration processes using the elimination and dispersal mechanism. Reproduction picks the elite individuals with higher modularity scores, which can help to accelerate the convergence speed.

3 Experimental evaluation

In this section, we will present our extensive experimental results to show the performance of the proposed BFO-FMD algorithm. The experimental platform is a PC with Inter(R) Core(TM) i5-3470 CPU 3.20 GHz, 4 GB RAM, and Windows 7, and BFO-FMD is implemented using the Java language.

3.1 Datasets

Three new publicly available yeast PPI datasets including DIP ScereCR20150101, DIP Scere20150101, and MIPS datasets are used in the experiments. Table 1 shows a summary of the datasets, where the 2nd column provides the web links, the 3rd and 4th columns are the number of proteins (P) and interactions (I) in source data, and the 5th and 6th columns give the number of proteins and interactions after deleting all the self-connected and repeated interactions. To evaluate the functional modules discovered by a computational method, the set of real functional modules from Friedel et al. (2008) is selected as the benchmark. This benchmark contains 428 gold standard functional modules and is constructed from three main sources: MIPS (Mewes et al. 2004; Aloy et al. 2004) and the SGD database (Dwight et al. 2002) based on the GO notations.

3.2 Evaluation metrics

Three types of popular evaluation metrics are employed to evaluate the quality of the detected modules (Li et al. 2010).

Table 1 Datasets used in experiments

Datasets	Http address	Source data		Processed data	
		Size of P.	Size of I.	Size of P.	Size of I.
ScereCR20150101	http://dip.doe-mbi.ucla.edu/dip/Download.cgi?SM=7&TX=4932	2460	5325	2391	5031
Scere20150101	http://dip.doe-mbi.ucla.edu/dip/Download.cgi?SM=7&TX=4932	5144	22,838	5087	22,424
MIPS	http://mips.helmholtz-muenchen.de	4554	15,456	4545	12,318

3.2.1 Precision, Recall, and F-measure

Precision, Recall, and F-measure are three common evaluation metrics in information retrieval and machine learning. Using this type of metric method, it is necessary to define how well a detected module $h = (V_h, E_h)$ matches a benchmark 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:

$$\mathrm{NA}(h, s) = \frac{|V_h \bigcap V_s|^2}{|V_h| \times |V_s|}. \tag{9}$$

If $\mathrm{NA}(h, s) \geq \omega$, 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 $N_{\mathrm{ch}} = |\{h|h \in H, \exists s \in S, \mathrm{NA}(h, s) \geq \omega\}|$, while the number of the standard modules in S which at least matches one detected module in H is indicated by $N_{\mathrm{cs}} = |\{s|s \in S, \exists h \in H, \mathrm{NA}(h, s) \geq \omega\}|$. Thus, Precision and Recall are given below:

$$\mathrm{Precision} = \frac{N_{\mathrm{ch}}}{|H|}, \tag{10}$$

and

$$\mathrm{Recall} = \frac{N_{\mathrm{cs}}}{|S|}. \tag{11}$$

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

$$F\text{-measure} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{12}$$

According to the above definitions, the values of Precision and Recall are actually ratios and range from 0 to 1. For Precision, the closer it is to 1, the more the detected modules which successfully match at least one standard module, and the better the algorithm. When the value of Precision is 1, all the detected modules can match at least one standard module. For Recall, the closer it is to 1, the more the standard modules which successfully match at least one detected module, and the better the algorithm. When the value of Recall is 1, all the standard modules can match at least one detected module. As an overall indicator of Precision and Recall, F-measure also takes values in the range of 0–1, and the closer it is to 1, the better the algorithm.

3.2.2 Sensitivity, positive predictive value, and accuracy

Sensitivity (Sn), positive predictive Value (PPV), and accuracy (Acc) are three usual measures 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:

$$\mathrm{Sn} = \frac{\sum_{i=1}^{|S|} \max_j \{T_{ij}\}}{\sum_{i=1}^{|S|} N_i}, \tag{13}$$

and

$$\mathrm{PPV} = \frac{\sum_{j=1}^{|H|} \max_i \{T_{ij}\}}{\sum_{j=1}^{|H|} T_{\cdot j}}, \tag{14}$$

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

Acc is a comprehensive metric and is defined as the geometric mean of Sn and PPV:

$$\mathrm{Acc} = (\mathrm{Sn} \times \mathrm{PPV})^{1/2}. \tag{15}$$

From the above definitions, the values of Sn and PPV are also ratios and range from 0 to 1. For Sn, the closer it is to 1, the detected modules give the better coverage of the standard modules, and the better the algorithm. When the value of Sn is 1, the detected modules can completely cover the standard modules. For PPV, the closer it is to 1, the detected modules are more likely to be the standard modules, and the better the algorithm. When the value of PPV is 1, the detected modules are the same as the standard modules, and the better the algorithm. As an overall indicator of Sn and PPV, Acc also takes values in the range of 0–1, and the closer it is to 1, the better the algorithm.

3.2.3 P value measure

P value, known as a metric of functional homogeneity, is usually used to substantiate the biological significance of the detected modules. It is the probability of co-occurrence of proteins in a detected module with a common function and is expressed as follows:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{|F|}{i}\binom{|V| - |F|}{|h| - i}}{\binom{|V|}{|h|}}, \tag{16}$$

where $|V|$ is the number of all the proteins in a PPI network, $|h|$ is the number of proteins in a detected module h, $|F|$ is the number of proteins in a reference function F, and k represents the number of common proteins in the reference function F and the detected module h. Low p value of a detected module

Table 2 Results of various algorithms on different datasets

Datasets	Results	Algorithms						
		BFO-FMD	NACO-FMD	ACC-FMD	COACH	Jerarca	CFinder	MCODE
ScereCR20150101	Number of modules	324	310	382	363	318	178	102
	Average size of modules	5.18	4.27	11.56	5.73	7.16	7.76	5.71
	$N_{ch} > 0.2$	158	110	230	179	118	90	59
	$N_{cs} > 0.2$	244	161	148	202	184	130	95
Scere20150101	Number of modules	348	479	270	916	588	204	60
	Average size of modules	5.83	4.48	31.46	8.81	8.23	13.13	13.83
	$N_{ch} > 0.2$	172	50	93	223	92	50	22
	$N_{cs} > 0.2$	263	80	81	213	140	67	38
MIPS	Number of modules	397	472	259	489	536	178	65
	Average size of modules	4.28	4.11	47.48	9.22	7.98	9.29	7.65
	$N_{ch} > 0.2$	155	74	63	144	66	55	29
	$N_{cs} > 0.2$	236	104	60	149	94	85	52

generally means that this module is not merely enriched by proteins from the same function by chance, and thus, this module has high statistical significance and is likely to be a real protein module. In general, a detected module with p value <0.01 is considered to be statistically significant.

3.3 Comparative evaluations

To demonstrate the performance of the proposed BFO-FMD algorithm, we compared it with six competitive methods: NACO-FMD (Ji et al. 2012a), ACC-FMD (Ji et al. 2015), COACH (Wu et al. 2009), Jerarca (Aldecoa and Marín 2010), CFinder (Adamcsek et al. 2006), and MCODE (Bader and Hogue 2003) on three yeast datasets in Table 1. Among them, NACO-FMD and ACC-FMD are two swarm intelligence-based algorithms introduced before. COACH is a core attachment-based algorithm, which first detects protein cores and then includes attachments into these cores to form biologically meaningful structures. Jerarca belongs to hierarchy-based algorithm, and it can provide optimal partitions of the trees using statistical criteria based on the distribution of intra-cluster and inter-cluster connections. CFinder and MCODE are two density-based algorithms. CFinder first identifies the k-cliques using clique percolation and then combines the adjacent k-cliques to get the functional modules. 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. In the experiments, NACO-FMD adopted the same parameters as Ji et al. (2012a), and ACC-FMD employed the same parameters as Ji et al. (2015). For methods COACH, Jerarca, CFinder, and MCODE, we obtained their software implementations and used the default values for their parameters in the experiments.

As for our BFO-FMD algorithm, there are several parameters. In the following, we will describe their roles. S is the size of bacterial population and determines number of individuals to search better module partitions. N_c is the maximum number of chemotaxis and determines the number of times searching around a candidate solution. N_s is the maximum number of swimming during each chemotaxis and determines the number of steps that an individual swims toward a specific direction. N_{re} is the maximum number of reproduction and determines the number of times that the algorithm selects superior solutions. N_{ed} is the maximum number of elimination and dispersal, which determines the number of times that the bacterial population explores some other regions of the search space. P_{ed} is the probability of elimination and dispersal and determines the number of individuals dispersed into some other regions to search better module partitions. P_{co} is the probability of conjugation and determines the number of individuals searching for better module partitions under the help of other individuals. P_{ch} is the probability used in generating a masking vector and determines the strength of local search around a candidate solution in the chemotaxis process. L is the conjugation length and determines the strength of local search around a candidate solution in the conjugation process. ε is the similarity threshold between nodes and determines the quality of initial solutions. λ is the merging threshold, and δ is the filtering threshold, and they determine the strength of post-processing operators. For these parameters, too big or too small values have serious effects on the solution quality and the runtime efficiency. To select a set of relatively good values, we have done a series of hand-tuning experiments and took $S = 100$, $N_c = 100$, $N_s = 4$, $N_{re} = 4$, $N_{ed} = 2$, $P_{ed} = P_{co} = 0.2$, $P_{ch} = 0.05$, $L = 0.05 * |V|$, $\varepsilon = 0.4$, $\lambda = 0.2$, and $\delta = 0.05$.

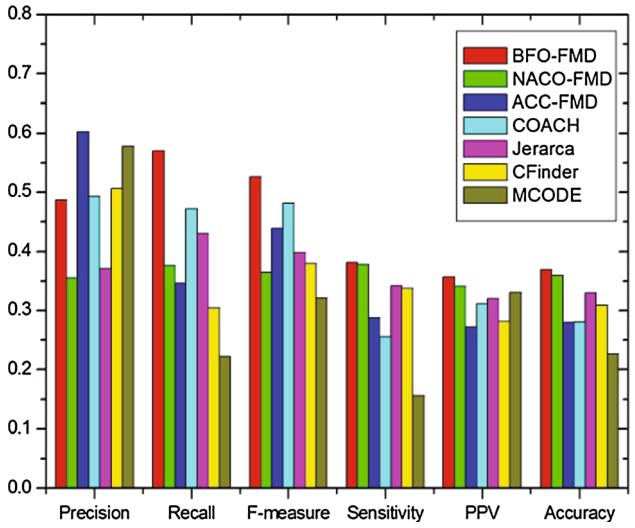

Fig. 4 Comparative results of seven methods in terms of various evaluation metrics on ScereCR20150101 dataset

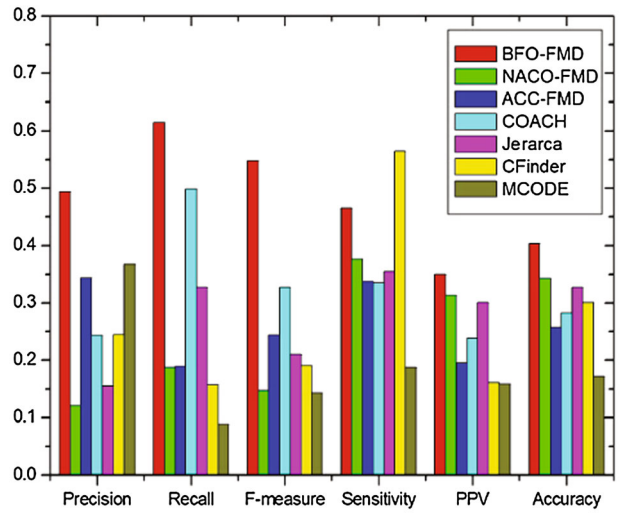

Fig. 5 Comparative results of seven methods in terms of various evaluation metrics on Scere20150101 dataset

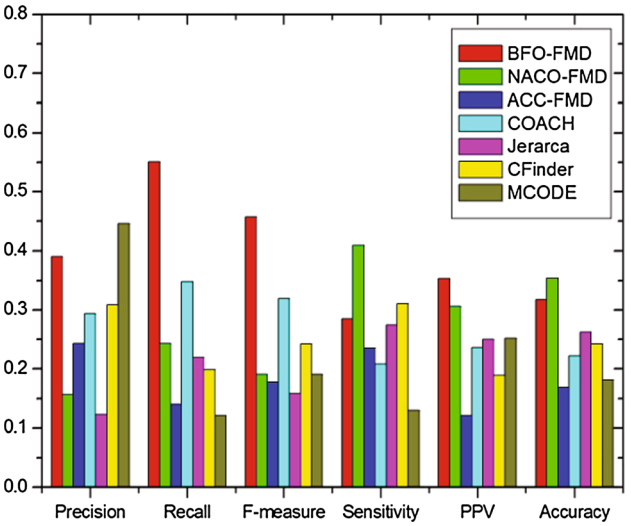

Fig. 6 Comparative results of seven methods in terms of various evaluation metrics on MIPS dataset

Table 2 provides the basic information of the detection results for the seven algorithms on the three datasets. For each algorithm, we 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 BFO-FMD on ScereCR20150101 dataset as an example, it has detected 324 modules, of which 158 match 244 gold standard modules, and the average size of the 324 detected modules is about five proteins. From the table, MCODE always obtains the least number of modules (number of modules), ACC-FMD gets the biggest module (average size of modules), and BFO-FMD usually generates more and smaller modules and the number of gold standard modules that match at least one detected module (N_{cs}) is always much more than all the other algorithms on the three datasets.

Figures 4, 5, and 6 show the overall comparison results of seven algorithms on the three datasets in terms of various evaluation metrics including Precision, Recall, F-measure, Sensitivity, PPV, and Accuracy. On the core dateset of yeast (ScereCR20150101) in Fig. 4, one easily see that our BFO-FMD algorithm achieves the best performance on five out of six metrics except Precision statistic. In detail, ACC-FMD and MCODE take the first and second places in term of Precision, respectively. However, ACC-FMD usually mines high overlapping modules and many of which are very similar, and MCODE detects very few protein modules (only 102 in Table 2), which leads to higher Precision values. The Precision of our algorithm is 0.487, which is almost the same as those of COACH (0.493) and CFinder (0.506), and higher than those of the remaining two methods NACO-FMD (0.355) and Jerarca (0.371). BFO-FMD algorithm matches the most real protein modules and obtains the highest Recall (0.57), which is much better than all the other six algorithms. On the whole, due to the balance of Precision and Recall, BFO-FMD algorithm also attains the highest F-measure (0.526). As for another three metrics of Sensitivity, PPV, and Accuracy, our algorithm still plays the best among the seven algorithms. Its corresponding values are 0.381, 0.357, 0.359 in terms of Sensitivity, PPV, Accuracy, respectively, which are a little better than those of NACO-FMD (0.378, 0.341, and 0.359) and much better than the other five algorithms.

Figure 5 shows that our BFO-FMD algorithm performs extremely well on larger Scere20150101 dataset, while other algorithms suffer performance degradation compared to their

Table 3 Distribution comparisons of the p values of protein modules obtained by different algorithms on ScereCR20150101

p values	Algorithms	Distribution ranges				Ratio
		(0, 1.0e−30]	(1.0e−30, 1.0e−20]	(1.0e−20, 1.0e−10]	(1.0e−10, 0.01]	
Biological process	BFO-FMD	6	8	37	201	0.778
	NACO-FMD	0	4	16	178	0.639
	ACC-FMD	1	33	122	205	0.945
	COACH	0	3	53	256	0.860
	Jerarca	0	13	36	181	0.534
	CFinder	4	9	23	109	0.815
	MCODE	0	0	15	79	0.922
Cellular component	BFO-FMD	9	12	33	155	0.645
	NACO-FMD	0	5	18	153	0.568
	ACC-FMD	9	49	119	153	0.864
	COACH	0	10	63	213	0.788
	Jerarca	3	8	40	143	0.610
	CFinder	4	7	27	92	0.730
	MCODE	0	0	23	69	0.902
Molecular function	BFO-FMD	1	7	25	123	0.481
	NACO-FMD	0	2	11	104	0.377
	ACC-FMD	0	16	60	197	0.715
	COACH	0	4	27	176	0.570
	Jerarca	0	3	24	120	0.462
	CFinder	0	2	20	76	0.551
	MCODE	0	0	8	57	0.637

results on smaller cereCR20150101 dataset. More specifically, BFO-FMD algorithm has overwhelming superiority in terms of five metrics: Precision, Recall, F-measure, PPV, and Accuracy. Only on Sensitivity statistic, CFinder achieves better than our BFO-FMD algorithm. It is actually because this algorithm obtained an impossibly huge module including 1858 proteins, so the great majority of proteins in benchmark set are covered by this very big module, which results in a high Sensitivity value for CFinder according to the calculation formula in Eq. (13).

For another larger MIPS dataset shown in Fig. 6, the proposed BFO-FMD algorithm still yields good results, even though it does not achieve as well as it does on Scere20150101. BFO-FMD attains the best performance in terms of Recall, F-measure, and PPV. On Precision statistic, BFO-FMD is only worse than MCODE. However, we do not think we should give MCODE a highest ranking on this statistic, since it predicted very few modules (65 in Table 2). The sensitivity value of our algorithm is 0.285, which is in third place out of the seven algorithms and is slightly lower than that of the second place CFinder (0.31). There is a relatively large gap between BFO-FMD and the first place NACO-FMD on this statistic. So BFO-FMD obtains the next best performance after NACO-FMD in term of Accuracy which is a comprehensive metric of Sensitivity and PPV, despite the fact that our algorithm achieves the best performance on PPV statistic.

The above outstanding experimental results of BFO-FMD on the three datasets fully demonstrate that BFO-FMD not only performs perfectly on smaller dataset (ScereCR2015 0101), but also plays very well on some relatively larger datasets (Scere20-150101 and MIPS). Therefore, the proposed BFO-FMD algorithm is a robust algorithm whose superior performance is not dependent on the underlying data.

Table 3 illustrates the relative performance of various algorithms in terms of p values, where the first column gives three types of p values, the second column lists seven algorithms, the third to sixth columns present the number of statistically significant modules located in the corresponding range. The last column is the proportion of significant modules (p value <0.01). In this table, we notice that MCODE and ACC-FMD get higher ratio than other methods. It is mainly because MCODE obtains the minimum amount of modules, and ACC-FMD is designed to search high overlapping modules from the PPI networks, which leads to their high ratios. As for other five algorithms, there is no major difference on the ratio, and our BFO-FMD method ranks in the middle. It is worth noting that most modules detected by various algorithms stand in ranges (1.0e−20, 1.0e−10] and (1.0e−10,

Table 4 Some functional modules predicted by BFO-FMD using ScereCR20150101 data

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	5	yjl053w yor069w yor132w yhr012w yjl154c	Retromer	1.000	6.98e−09	4.36e−12	3.55e−07
2	20	ydr228c ydr195w ynl317w yor250c ymr061w ydr301w ykr002w ygl044c ylr115w yjr093c ykl059c ylr277c yal043c ykl018w yor179c ygr156w ypr107c ynl222w yol123w ygl122c	mRNA cleavage factor	0.903	2.84e−34	6.45e−42	2.73e−12
3	14	ygl003c ydr260c ykl022c ylr102c ydr118w ynl172w yir025w ybl084c yfr036w ylr127c ygl240w ydl008w yhr166c yor249c	Anaphase−promoting	0.875	1.42e−18	5.81e−27	1.72e−12
4	14	yhr069c ygr095c ycr035c ygr195w yor076c yol021c ydl111c yor001w ydr280w ynl232w ygr158c yol142w yhr081w ylr398c	Exosome	0.862	1.28e−26	5.11e−28	6.37e−09
5	11	ydr407c yel048c ymr218c ygr166w ybr254c ydr108w ykr068c ydr472w yor115c ydr246w yml077	TRAPP	0.909	5.88e−12	1.56e−27	5.14e−14
6	8	ydl232w yel002c ymr149w yjl002c ygl226c-a ygl022w yor085w P46964	Oligosaccharyl transferase	0.681	7.26e−15	1.69e−17	4.59e−15
7	7	ynr035c ydl029w yjr065c ykl013c yil062c ybr234c ylr370c	Arp2/3	0.875	1.64e−15	3.98e−17	4.79e−08
8	7	ylr015w ypl138c yar003w yhr119w ydr469w ybr175w ybr258c	Set1C/COMPASS	0.875	6.12e−14	4.28e−17	4.28e−17
9	7	ylr384c ypl086c ykl110c yhr187w ymr312w ygr200c ypl101w	Elongator holoenzyme	0.857	5.43e−12	7.55e−15	5.88e−06
10	5	ylr200w yel003w yml094w ygr078c ynl153ccc	Prefoldin	0.714	6.91e−11	2.74e−13	4.66e−08

Table 5 Some functional modules predicted by NACO-FMD using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	4	yjl154c yjl053w yhr012w Q96QK1	Retromer	0.450	9.30e−07	3.68e−09	2.04e−05
2	11	ydr407c ymr218c ybr254c ydr108w ydr472w ykr068c yor115c ydr246w yml077w ygr166w yel048c	TRAPP	0.909	5.36e−12	1.55e−27	5.12e−14
3	14	yhr069c ygr095c ygr195w yor076c yol021c ygl213c ydl111c ydr280w ynl232w ygr158c yol142w ycr035c ylr398c ybr059c	Cytoplasmic exosome	0.714	5.82e−28	9.74e−24	9.64e−06
4	8	ynr035c yor181w yjr065c yil062c ydl029w ybr234c ykl013c ylr370c	Arp2/3	0.766	3.19e−18	3.36e−16	1.12e−09
5	10	ydr334w Q12692 ygr002c yhr099w yor244w ynl107w yfl024c ypr023c ydr359c yhr090c	NuA4 histone acetyltransferase	0.485	1.55e−07	6.55e−16	3.05e−06
6	8	ylr291c ykr026c ygr083c ypl237w ydr211w yer025w yor260w yjr007w	Eukaryotic translation initiation factor 2B	0.625	7.21e−13	2.51e−11	4.58e−14
7	7	ymr309c ydr429c ynl062c yil071c ymr146c yor361c ybr079c	Eukaryotic translation initiation factor 3	0.510	3.42e−10	1.44e−10	2.37e−09
8	4	yor358w ybl021c ygl237c ykl109w	CCAAT-binding factor	1.000	7.77e−10	1.55e−10	2.73e−06
9	6	yfl033c yml100w ybr126c ydr074w yil177c ymr261c	Alpha,alpha-trehalose-phosphate synthase	0.667	2.76e−09	2.76e−09	2.00e−06
10	4	ylr015w yar003w ybr258c ybr175w	Set1C/COMPASS	0.500	2.96e−07	1.14e−08	1.14e−08

Table 6 Some functional modules predicted by ACC-FMD using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	5	yor069w yor132w yjl154c yjl053w yhr012w	Retromer	1.000	6.80e−09	4.25e−12	3.46e−07
2	16	yjr093c ydr228c yer133w yjl033w ynl317w ymr061w ydr301w ykr002w ylr115w ykl059c ygr048w yal043c ykl018w yor179c ygr156w ypr107c	mRNA cleavage factor	0.613	1.08e−14	6.42e−22	2.71e−13
3	17	yor249c ydr260c ykl022c ylr102c ydr118w ynl172w yer161c ybl084c yfr036w ylr127c ygl240w ydl008w yhr166c yol133w ygl003c ypr119w P10815	Anaphase-promoting	0.621	1.01e−15	3.23e−26	2.73e−12
4	11	ydr246w yfl038c ydr407c yel048c ymr218c ybr254c ydr108w ykr068c ydr472w yor115c yml077w	TRAPP	0.736	7.08e−12	4.87e−23	6.83e−11
5	13	ykl144c yor210w ybr154c yor116c ypr110c ynr003c ypr190c ypr010c yor224c yjl011c yor207c ynl113w ygr246c	DNA-directed RNA polymerase III	0.548	1.31e−21	4.63e−22	4.62e−22
6	10	ylr370c ynr035c P36006 ydl029w ymr109w yjr065c ykl013c yil062c ybr234c ylr429w	Arp2/3	0.800	3.37e−17	2.05e−19	2.38e−11
7	10	ybr175w ylr015w ynl317w ykl018w yor179c ypl138c yar003w yhr119w ydr469w ybr258c	Set1C/COMPASS	0.800	1.24e−14	9.69e−19	9.69e−19
8	13	yjr015w ydl232w ymr149w yjl002c P46964 yml019w yor085w ybl105c yel002c ygl226c-a ygl022w yor231w P53620	Oligosaccharyl transferase	0.547	3.17e−14	1.07e−17	1.67e−12
9	11	yor144c yol094c ynl290w ybr087w yhr191c ybr088c yjr068w ycl016c ybl035c ymr078c yor217w	Ctf18 RFC-like	0.636	5.77e−13	1.86e−15	1.75e−09
10	4	ydr074w yml100w ybr126c ymr261c	Alpha,alpha-trehalose-phosphate synthase	1.000	9.75e−11	9.75e−11	2.12e−07

Table 7 Some functional modules predicted by COACH using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	5	yjl053w yjl154c yor069w yor132w yhr012w	Retromer	1.000	6.80e−09	4.25e−12	3.46e−07
2	13	ygr005c yhr143w-a yil021w ybr154c ydl140c ydr404c ygl070c yjl140w yol005c yor151c yor210w yor224c ypr187w	DNA-directed RNA polymerase II, core	0.923	3.37e−09	1.64e−28	1.64e−28
3	11	ybl084c ydl008w ydr118w yfr036w ygl240w yhr166c ykl022c ylr127c ynl172w yor249c ylr102c	Anaphase-promoting	0.688	1.13e−13	1.71e−20	4.05e−15
4	16	yer012w yjl001w ypr103w p21242 p38886 q12250 ybl041w ydl188c yer094c yfr050c ygl011c ygr135w yml092c ymr314w yol038w yor157c	Proteasome core	0.800	2.00e−27	1.40e−26	2.00e−27
5	11	ycr035c ygr095c ygr195w yhr069c yol021c ydl111c ydr280w ygr158c ynl232w yol142w yor076c	Cytoplasmic exosome	0.909	6.91e−24	1.90e−26	2.11e−06
6	8	yar003w ybr175w ybr258c ydr469w yhr119w ylr015w ypl138c ykl018w	Set1C/COMPASS	1.000	2.73e−16	2.12e−20	2.12e−20
7	8	p46964 ydl232w yel002c ygl022w ygl226c-a yjl002c ymr149w yor085w	Oligosaccharyl transferase	0.681	7.23e−15	1.69e−17	4.58e−15
8	7	ydl029w ylr429w ynr035c ybr234c yil062c yjr065c ylr370c	Arp2/3	0.875	1.80e−15	4.36e−17	6.11e−08
9	6	ygr200c yhr187w ylr384c ymr312w ypl086c ypl101w	Elongator holoenzyme	1.000	4.46e−10	9.40e−16	2.20e−06
10	5	ydr211w ygr083c ykr026c ylr291c yor260w	Eukaryotic translation initiation factor 2B	1.000	5.92e−09	2.91e−13	2.86e−08

Table 8 Some functional modules predicted by Jerarca using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	6	yor069w yjl154c yjl053w yor132w yhr012w q96qk1	Retromer	0.833	7.56e−11	5.24e−15	9.19e−09
2	25	ydr228c yor250c ydr301w ymr061w ygl044c ydr195w ynl317w ykl059c ylr277c yal043c ykr002w yjr093c ypr107c ykl018w yor179c ylr115w q07509 ynl222w ylr015w ypl138c yar003w yhr119w ybr175w ydr469w ybr258c	mRNA cleavage factor	0.578	4.43e−23	9.43e−32	2.10e−14
3	14	ykl022c ydr118w ydl008w yor249c ylr127c yhr166c ynl172w yfr036w ybl084c ygl240w ylr102c p10815 yir025w ygr225w	Anaphase-promoting	0.754	1.91e−14	9.17e−25	3.59e−13
4	11	ydr407c ymr218c ygr166w ybr254c ydr472w ydr108w kr068c yor115c ydr246w yml077w yel048c	TRAPP	0.909	5.36e−12	1.55e−27	5.12e−14
5	13	yjl194w ypr162c yll004w ynl261w yhr118c ybr060c yml065w ybl023c yel032w yil150c ybr202w p53091 ylr274w	Pre-replicative	0.621	3.36e−22	1.18e−22	1.21e−22
6	10	yjr068w ynl290w yol094c ybr087w yhr191c ymr078c yor144c ybl035c yor217w ycl016c	Ctf18 RFC-like	0.700	2.62e−11	5.03e−16	7.11e−10
7	4	yor358w ygl237c ybl021c ykl109w	CCAAT-binding factor	1.000	7.77e−10	1.55e−10	2.73e−06
8	4	ydr148c yfr049w yfl018c yil125w	Mitochondrial oxoglutarate dehydrogenase	0.750	1.90e−10	1.90e−10	4.12e−07
9	5	yml100w ydr074w ybr126c ymr261c yil177c	Alpha, alpha-trehalose-phosphate synthase	0.800	6.27e−10	6.27e−10	6.81e−07
10	7	ygr180c yjl026w yor229w yor230w yer070w yml058w yil066c	Ribonucleoside-diphosphate reductase	0.571	4.83e−08	9.68e−09	9.68e−09

Table 9 Some functional modules predicted by CFinder using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	5	yhr012w yjl053w yor069w yjl154c yor132w	Retromer	1.000	6.80e−09	4.25e−12	3.46e−07
2	21	ydl140c yer133w P04147 ygl044c ymr061w ylr115w yol123w yal043c ydr228c ydr301w yjr093c ykl018w ykl059c ylr277c ypr107c yor250c ydr195w ykr002w ynl317w yor179c ynl222w	mRNA cleavage factor	0.860	1.51e−27	1.09e−41	6.17e−09
3	13	ycr035c yol021c ydl111c ygr095c yhr069c ydr280w ygr195w yor076c ygr158c ynl232w yol142w yhr081w yor001w	Exosome	0.929	1.68e−27	3.34e−29	4.37e−08
4	10	ybr254c ygr166w yor115c ydr108w ydr472w ykr068c ydr246w ydr407c yml077w ymr218c	TRAPP	1.000	9.33e−11	3.29e−24	1.83e−14
5	10	P46964 ydl232w ygl022w ygl226c-a yor085w yel002c ybl105c yjl002c ymr149w yjr015w	Oligosaccharyl transferase	0.544	5.77e−13	1.35e−15	1.22e−13
6	7	ylr384c ygr200c ymr312w yhr187w ykl110c ypl101w ypl086c	Elongator holoenzyme	0.857	5.42e−12	7.53e−15	5.87e−06
7	11	ybr088c ybl035c ybr087w ynl290w yor217w yer173w yol094c yor144c yjr068w yhr191c ymr078c	Ctf18 RFC-like	0.468	1.29e−10	5.72e−12	8.18e−13
8	9	ykr026c ydr211w yjr007w ylr291c ypl237w yer025w ypr041w ygr083c yor260w	Eukaryotic translation initiation factor 2B	0.556	1.11e−14	6.31e−11	4.71e−16
9	4	ybr126c yml100w ymr261c ydr074w	Alpha,alpha-trehalose-phosphate synthase	1.000	9.75e−11	9.75e−11	2.12e−07
10	5	ybr091c yhr005c-a yjl054w yor297c ydl217c	Mitochondrial inner membrane protein insertion	0.800	1.41e−10	9.96e−09	2.19e−07

Table 10 Some functional modules predicted by MCODE using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module	NA	*p* values		
					Biological	Cellular	Molecular
1	5	yhr012w yor069w yor132w yjl154c yjl053w	Retromer	1.000	6.80e−09	4.25e−12	3.46e−07
2	8	yll004w yjl194w ypr162c yml065w ybl023c ynl261w yhr118c ybr060c	Pre-replicative	0.533	5.06e−16	2.68e−16	3.25e−14
3	10	yor249c ykl022c ydr118w ynl172w ybl084c yfr036w ylr127c ygl240w ydl008w yhr166c	Anaphase-promoting	0.625	1.05e−14	2.32e−18	1.49e−13
4	8	ygl022w ydl232w yel002c ymr149w yjl002c ygl226c-a yor085w p46964	Oligosaccharyl transferase	0.681	7.23e−15	1.69e−17	4.58e−15
5	6	ymr146c ymr309c ydr429c yor361c ypr041w ybr079c	Multi-eIF	0.521	4.28e−11	1.97e−13	3.54e−10
6	4	ymr261c yml100w ydr074w ybr126c	Alpha, alpha-trehalose-phosphate	1.000	9.75e−11	9.75e−11	2.12e−07
7	5	yor297c yhr005c-a ydl217c ybr091c yjl054w	Mitochondrial inner membrane protein insertion	0.800	1.41e−10	9.96e−09	2.19e−07
8	6	ymr078c yjr068w yol094c ynl290w ybr087w yhr191c	Ctf18 RFC-like	0.857	2.12e−09	6.39e−15	3.23e−08
9	5	ydr211w ylr291c yor260w ykr026c ygr083c	Eukaryotic translation initiation factor 2B	1.000	5.92e−09	2.91e−13	2.86e−08
10	9	q06406 ycr077c yjr022w yer146w ylr438c-a ynl147w yer112w ybl026w yjl124c	snRNP U6	0.500	1.06e−07	2.29e−13	1.18e−05

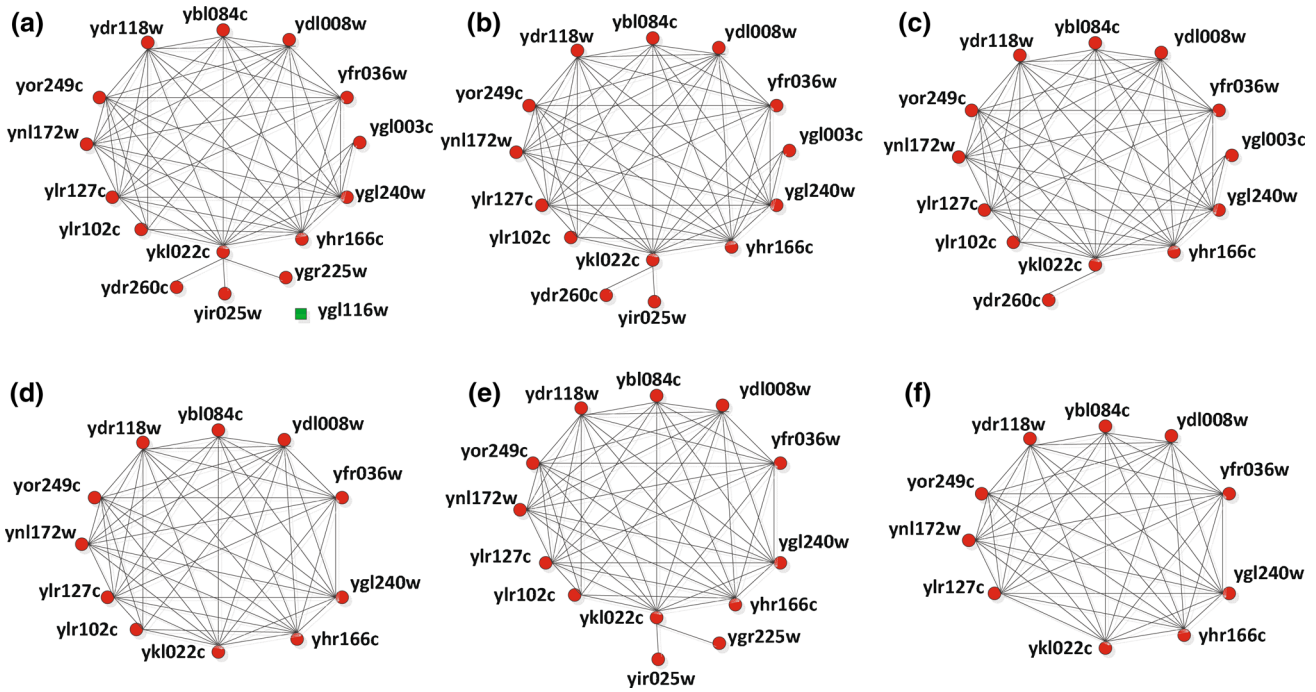

Fig. 7 Anaphase-promoting module detected by various algorithms: **a** Benchmark, **b** BFO-FMD, **c** ACC-FMD, **d** COACH, **e** Jerarca, **f** MCODE

0.01], while only a few modules fall into the intervals (0, 1.0e−30] and (1.0e−30, 1.0e−20], where BFO-FMD has obvious advantages comparing with other algorithms except ACC-FMD. This result illustrates that the proposed BFO-FMD method is able to detect more modules with very high statistical significance.

To further investigate the computational results, 10 modules with low p values and high matching degree mined by different methods on ScereCR20150101 dataset are presented in Tables 4, 5, 6, 7, 8, 9, and 10. In these tables, the first column is a module identifier. The second and third columns give the number of proteins, and the proteins contained in each module, respectively. The fourth column provides the corresponding gold standard protein module. The fifth column refers to the matching degree measured by the neighborhood affinity score (NA) between a predicted module and a gold standard module. The last three columns list three types of p values of predicted modules from the view of biological process, cellular component, and molecular function. According to the NA metric in the fifth column, it is known that many modules detected by the seven methods match well with the benchmark modules. All the p values in these tables are very low, which perfectly demonstrates the modules detected by these computational methods have high statistical significance from three different GO categories.

To explicitly reveal the results obtained by our BFO-FMD algorithm, we take the anaphase-promoting module as an example to explain. Tables 4, 5, 6, 7, 8, 9 and 10 show that five algorithms BFO-FMD, ACC-FMD, COACH, Jerarca, and MCODE have detected the anaphase-promoting module (the 3rd module in Tables 4, 5, 6, 7, 8, 9, and 10). Moreover, according to the NA statistic, BFO-FMD obtains the highest matching degree between the predicted anaphase-promoting module and the standard one among the five algorithms. Figure 7 illustrates the module structures for the gold standard anaphase-promoting module and the detected ones for the five algorithms. Figure 7a shows the gold standard anaphase module, which contains 16 proteins, and one of them (ygl116w) is isolated by other proteins in the same module. The anaphase-promoting module detected by our BFO-FMD algorithm is given in Fig. 7b. This module consists of 14 proteins and succeeds in matching all the 14 proteins, which are more than those of other algorithms, and only one protein is missing in contrast with the standard one except the isolated protein (ygl116w). The other four algorithms ACC-FMD, COACH, Jerarca, and MCODE only cover 13, 11, 13, 10 proteins in standard anaphase-promoting module, as shown in Fig. 7c–f, respectively. This example intuitively demonstrates that the proposed BFO-FMD algorithm is able to more accurately detect functional modules.

In addition, the proposed BFO-FMD algorithm has the potential to discover some new modules with biological significance from the statistical point of view. Table 11 lists five new modules with very low p values on the ScereCR20150101 dataset. These five modules can be inquired on the Web site, but are not described in the set

Table 11 Some new functional modules predicted by BFO-FMD using ScereCR20150101 dataset

ID	Size	Proteins in the predicted module	Real protein module[a]	*p* values		
				Biological	Cellular	Molecular
1	28	ydl140c yol005c yor210w ypr187w yil021w yhr143w-a yor224c ynl113w ybr154c yjl140w yor151c ykl144c ygl070c ypr110c ynr003c ypr190c yjl011c ykr025w ydl150w yor116c yor207c ydr045c ynl248c ypr010c yor340c yor341w yjr063w ydr404c	Nuclear DNA-directed RNA polymerase	8.57e−32	1.16e−39	1.49e−59
2	20	ydr308c ynl236w yol135c ygl025c ycr081w ydl005c yor174w ygr104c yer022w ylr071c ypr070w yhr058c yhr041c ybl093c ypl129w ymr112c ybr253w yol051w ygr005c ygr186w	Mediator complex	9.68e−16	1.43e−34	5.49e−27
3	9	ydr261w-a ycl019w ydr261w-b ygr161w-b ybl100w-a ypr158w-a yol103w-b ynl054w-b ypl257w-b	Retrotransposon nucleocapsid	6.29e−18	3.74e−22	6.37e−14
4	15	ydr356w ynl126w ylr212c yhr172w ykl042w ydl239c ynl225c ylr045c yor373w ypl124w yal047c yol091w yhr184w ypl022w yml095c	Spindle pole body	9.00e−16	2.58e−16	4.71e−16
5	11	ydr507c ycl024w ycr002c yhr107c ylr314c ydl225w yjr076c ynl166c ydr218c ymr139w ykr048c	Septin cytoskeleton	2.36e−12	6.87e−13	5.08e−06

[a] Available on the Web site http://amigo1.geneontology.org/cgi-bin/amigo/term_enrichment

of benchmark modules, and the 1st, 3rd, and 5th modules are not found by any of other six algorithms. This result suggests that BFO-FMD has more powerful exploratory ability to detect functional modules from a PPI network and can help biologists predict some new protein modules to a certain extent.

In short, the above experimental results and analysis have justified the effectiveness of the proposed BFO-FMD algorithm. Specifically, BFO-FMD is not only better than some famous non-swarm intelligence-based algorithms (COACH, Jerarca, CFinder, and MCODE), but is also superior to some new developed swarm intelligence-based algorithms (NACO-FMD and ACC-FMD). It is because that BFO-FMD has four biological mechanisms (chemotaxis, conjugation, reproduction, and elimination and dispersal) designed appropriately and obtains good ability of balancing global search and local search.

4 Conclusions

The identification of functional modules in a PPI network is important for biological knowledge discovery since many important biological processes in the cell are carried out through the formation of protein modules. Nowadays, the computational approaches based on swarm intelligence have been a kind of effective ways for functional module detection due to their good robustness performance. In this paper, a new swarm intelligence algorithm based on bacterial foraging optimization was proposed for detecting protein modules in a PPI network (called as BFO-FMD). BFO-FMD first combines the topology and function information between protein nodes to construct a candidate solution for each bacterial individual according to a random-walk behavior. Then, it accomplishes information exchange and optimizes each candidate solution by four biological mechanisms: chemotaxis, conjugation, reproduction, and elimination and dispersal. At last, two post-processing operators are performed to refine the detected result in light of function similarity and topological structure. To demonstrate the performance of BFO-FMD, we have carried out a series of experiments on three common yeast datasets in terms of various evaluation metrics. The empirical results illustrate that BFO-FMD achieves prominent Recall, F-measure, and PPV while performing very well in terms of other metrics, such as Precision, Sensitivity, Accuracy, and P values. These results indicate that the proposed BFO-FMD algorithm is able to detect effectively protein modules and can be considered as a complement to help biologists to get some novel biological insights.

Acknowledgements This work was partly supported by the NSFC Research Program (61672065, 61375059) and the Beijing Municipal Education Research Plan Key Project (Beijing Municipal Fund Class B) (KZ201410005004).

Compliance with ethical standards

Conflict of interest All the authors declare that there is no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

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

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

Aldecoa R, Marín I (2010) Jerarca: efficient analysis of complex networks using hierarchical clustering. PLoS ONE 5(7):e11585

Aloy P, Böttcher B, Ceulemans H et al (2004) Structure-based assembly of protein complexes in yeast. Science 303(5666):2026–2029

Altaf-Ul-Amin M, Shinbo Y, Mihara K, Kurokawa K, Kanaya S (2006) Development and implementation of an algorithm for detection of protein complexes in large interaction networks. BMC Bioinform 7(1):207

Arnau V, Mars S, Marín I (2005) Iterative cluster analysis of protein interaction data. Bioinformatics 21(3):364–378

Bader GD, Hogue CWV (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinform 4(1):1

Balasubramaniam S, Lio P (2013) Multi-hop conjugation based bacteria nanonetworks. IEEE Trans Nanobiosci 12(1):47–59

Chin E, Zhu J (2013) B3Clustering: identifying protein complexes from protein–protein interaction network. In: Proceedings of Asia-Pacific web conference. Springer, Berlin, pp 108–119

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

Das S, Biswas A, Dasgupta S, Abrham A (2009) Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. Found Comput Intell 3:23–55

Dwight SS, Harris MA, Dolinski K et al (2002) Saccharomyces Genome Database (SGD) provides secondary gene annotation using the Gene Ontology (GO). Nucleic Acids Res 30(1):69–72

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

Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972–976

Friedel CC, Krumsiek J, Zimmer R (2008) Bootstrapping the interactome: unsupervised identification of protein complexes in yeast. In: Proceedings of annual international conference on research in computational molecular biology. Springer, Berlin, pp 3–16

Guimera R, Amaral LAN (2005) Functional cartography of complex metabolic networks. Nature 433(7028):895–900

Hinchey MG, Sterritt R, Rouff C (2007) Swarms and swarm intelligence. Computer 40(4):111–113

Inoue K, Li W, Kurata H (2010) Diffusion model based spectral clustering for protein–protein interaction networks. PLoS ONE 5(9):e12623

Ji J, Liu Z, Zhang A, Jiao L, Liu C (2012a) Improved ant colony optimization for detecting functional modules in protein–protein interaction networks. In: Proceedings of international conference on information computing and applications. Springer, Berlin, pp 404–413

Ji J, Liu Z, Zhang A, Jiao L, Liu C (2012b) Ant colony optimization with multi-agent evolution for detecting functional modules in protein–protein interaction networks. In: Proceedings of international conference on information computing and applications. Springer, Berlin, pp 445–453

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

Ji J, Zhang A, Liu C, Quan X (2014a) Survey: functional module detection from protein–protein interaction networks. IEEE Trans Knowl Data Eng 26(2):261–277

Ji JZ, Liu ZJ, Liu HX, Liu CN (2014b) An overview of research on functional module Detection for protein–protein interaction networks. Acta Autom Sin 40(4):577–593

Ji J, Liu H, Zhang A, Liu Z, 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

King AD, Pržulj N, Jurisica I (2004) Protein complex prediction via cost-based clustering. Bioinformatics 20(17):3013–3020

Lei X, Wu S, Ge L, Zhang A (2011) Clustering PPI data based on bacteria foraging optimization algorithm. In: Proceedings of 2011 IEEE international conference on bioinformatics and biomedicine (BIBM), pp 96–99

Lei X, Wu S, Ge L, Zhang A (2013) Clustering and overlapping modules detection in PPI network based on IBFO. Proteomics 13(2):278–290

Leung HCM, Xiang Q, Yiu SM, Chin FYL (2009) Predicting protein complexes from PPI data: a core-attachment approach. J Comput Biol 16(2):133–144

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

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

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

Mewes HW, Amid C, Arnold R et al (2004) MIPS: analysis and annotation of proteins from whole genomes. Nucleic Acids Res 32(suppl 1):D41–D44

Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. Control Syst 22(3):52–67

Perales-Graván C, Lahoz-Beltra R (2008) An AM radio receiver designed with a genetic algorithm based on a bacterial conjugation genetic operator. IEEE Trans Evolut Comput 12(2):129–142

Qin G, Gao L (2010) Spectral clustering for detecting protein complexes in protein–protein interaction (PPI) networks. Math Comput Model 52(11):2066–2074

Ravasz E, Somera AL, Mongru DA, Oltvai ZN, Barabási AL (2002) Hierarchical organization of modularity in metabolic networks. Science 297(5586):1551–1555

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

Schlicker A, Albrecht M (2008) FunSimMat: a comprehensive functional similarity database. Nucleic Acids Res 36(suppl 1):D434–D439

Sen TZ, Kloczkowski A, Jernigan RL (2006) Functional clustering of yeast proteins from the protein–protein interaction network. BMC Bioinform 7(1):355

Tarassov K, Messier V, Landry CR, Radonovic S (2008) An in vivo map of the yeast protein interactome. Science 320(5882):1465–1470

Van Dongen S (2000) A cluster algorithm for graphs. Rep Inf Syst 10:1–40

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

Wu S, Lei X, Tian J (2011) Clustering PPI network based on functional flow model through artificial bee colony algorithm. In: Proceedings of 2011 seventh international conference on natural computation (ICNC'11), pp 92–96

Zhang A (2009) Protein interaction networks: computational analysis. Cambridge University Press, Cambridge