2014 IEEE International Conference on Bioinformatics and Biomedicine

# Detecting Functional Modules in Dynamic Protein-Protein Interaction Networks Using Markov Clustering and Firefly Algorithm

Xiujuan Lei*
School of Computer Science,
Shaanxi Normal University,
Xi'an, Shaanxi 710062, China
* xjlei@snnu.edu.cn

Fei Wang
School of Computer Science,
Shaanxi Normal University,
Xi'an, Shaanxi 710062, China
flykingwf@snnu.edu.cn

Fang-Xiang Wu
Division of Biomedical Engineering,
University of Saskatchewan,
Saskatoon, SK S7N 5A9, Canada
faw341@mail.usask.ca

Aidong Zhang
Department of Computer Science and Engineering,
State University of New York at Baffalo,
NY 14260-2000, USA
azhang@buffalo.edu

***Abstract*—Markov Clustering (MCL) is a popular algorithm for clustering networks in bioinformatics such as Protein-Protein Interaction (PPI) networks and especially, shows excellent performance in clustering Dynamic Protein-protein Interaction Networks (DPIN). However, a limitation of MCL and its variants (e.g. regularized MCL and soft regularized MCL) is that the clustering results are mostly dependent on the parameters that user-specified. However we know that different networks with various scales need different parameters. In this article, we propose a new MCL method based on the Firefly Algorithm (FA) to optimize its parameters. The results on DIP dataset show that the new algorithm outperforms the state-of-the-art approaches in terms of accuracy of identifying functional modules on a real DPIN.**

***Keywords—Dynamic Protein-Protein Interaction Network (DPIN); Markov Clustering (MCL) algorithm; Firefly Algorithm (FA)***

## I. INTRODUCTION

Proteomics is the systematic study of the diverse properties of proteins and provide detailed descriptions of the structure, function and control of biological systems in health and disease [1]. Usually, proteins seldom act as single or isolated entity. However, proteins involved in the same cellular processes often interact with each other to incorporate into a large molecule to accomplish the biological functions [2]. This is the Protein-Protein Interaction (PPI) network which involves many proteins and interactions. As biological functions are time-sensitive, proteins and interactions do not always exist. In response to a stimulus or a new condition in a living cell, the amounts and locations of proteins change [3], meanwhile the PPI networks change too [4]. Moreover, there are a large number of false-positives or negatives existing in the current available PPI data [35]. To express the dynamic traits and reduce the effect of the false positives, gene expression profiles have been used [36]. This is the Dynamic Protein-protein Interaction Network (DPIN).

Traditional clustering methods perform not so well for PPI data due to its small world and scale free properties [5, 6]. Many new methods are proposed for clustering PPI networks. In 2002, Girvan and Newman proposed a new method, named GN Algorithm, to detect function modules in networks [7]. And Newman also proposed a Newman fast algorithm in 2003, which is based on hierarchy condensations [8]. In 2006 Adamcsek *et al.* developed the software CFinder to uncover the overlapping clusters in biological networks [10]. In 2009, Leung developed a Core-Attachment approach for predicting protein complexes from the PPI network of single species based on a recent study on the organization of protein complexes [11]. There are also many other algorithms, like MCODE [9], COACH [12], RNSC [13], DPClus [14], IPCA [15], Link Community (LinkCom) [16] and Markov Clustering (MCL) [17].

MCL is a graph clustering algorithm based on stochastic flow simulation, which has proven to be effective on clustering biological networks [18, 19]. However, in spite of its popularity in the bioinformatics community, MCL has drawn limited attention from the data mining area primarily because it does not scale very well to moderate sized graphs [22]. To retain the strengths of MCL and redress its weakness, Vene Satuluri proposed a Regularized MCL (R-MCL) [20] and Yu-Keng Shih proposed a 'Soft' R-MCL (SR-MCL) [23]. Although R-MCL and SR-MCL are better than MCL, their parameters are still user-specified which makes it incapable of dealing with a variety of PPI data sets effectively. So we proposed a novel algorithm to automatically adjust the parameters by introducing Firefly

978-1-4799-5669-2/14/$31.00 ©2014 IEEE

Algorithm (FA), which is a kind of swarm intelligence optimization algorithm.

Swarm intelligence is a type of population-based metaheuristic approaches. It seeks near-optimal solutions of the difficult optimization problems by simulating the collective behavior of social insects (like bees, ants, and fishes). In 2012, Lei proposed an improved functional-flow based approach through Quantum-behaved Particle Swarm Optimization (QPSO) algorithm, which can find the optimum threshold automatically when calculating the lowest similarity between modules [33]. In 2013, Lei raised an improved Bacteria Foraging Optimization (BFO) based on BFO mechanism and intuitionistic fuzzy set [34], and a novel PMABC-ACE model based on the propagating mechanism of artificial bee colony [24]. The two approaches could detect the overlapping modules and the time complexity was greatly reduced.

Firefly Algorithm is also a swarm intelligent optimization algorithm inspired by simulating the luminescence properties of fireflies based on the group search [25]. The main principle of this method is that for any two flashing fireflies, the less bright one will move towards the brighter one. Then all fireflies will gather around the brightest individual.

In this article we seek to develop a method, named F-MCL that combines MCL and its variants with FA and redresses their weaknesses. In MCL, the inflation parameter *rp* influences the size of each cluster. But a global static parameter *rp* cannot be suited for all scale. So we want to use a different *r* in a different scale. To make sure the value of *r* is suitable, the FA is introduced to optimize *r* in every sub-network. At last we check every clusters' quality and gain a more accurate result. In our empirical study, we compare and contract our method with MCL and some other commonly used methods.

The outline of the paper is as follows. Section 2 describes some preliminary theories and the details of our algorithms, section 3 shows the experiments results and analysis, and section 4 concludes the paper.

## II. Preliminary and Method

### A. Matrix Construction

Let $G=(V, E)$ be the input graph which is undirected and includes self-loop, denotes a PPI network that $V$ denotes the node set and $E$ denotes the edge set. Then $v_i$ is a node in $V$ and each edge is denoted by $(v_i, v_j)$, $(v_i, v_j)\in E$. And $w(v_i, v_j)$ is the weight of this edge, which represents the confidence level of the interaction in a weighted PPI networks. Let $\boldsymbol{A}$ be the adjacency matrix of the graph such that,

$$A(i, j) = \begin{cases} w(v_i, v_j) & if(v_i, v_j) \in E \\ \max_{x \neq j} w(v_i, v_x) & if(v_i = v_j) \\ 0 & else \end{cases} \quad (1)$$

A canonical flow matrix $\boldsymbol{M}$ is an $n\times n$ matrix that can be interpreted as the matrix of the transition probabilities of a random walk (or a Markov chain) defined on the graph and $n$ is the number of nodes. The $i$th column of $\boldsymbol{M}$ represents the transition probabilities out of the $v_i$ and the $j$th row of $\boldsymbol{M}$ represents the transition probabilities into the $v_j$. Specifically, $M(i,j)$ represents the probability of a transition from vertex $v_i$ to $v_j$. We also refer to the transition probability from $v_i$ to $v_j$ as the stochastic flow or simply the flow from $v_i$ to $v_j$. Note that all the columns should sum to 1. And $M(i,j)$ is defined as,

$$M(i, j) = \frac{A(i, j)}{\sum_{k=1}^{n} A(k, j)} \quad (2)$$

### B. Markov Clustering Algorithm

MCL, R-MCL and SR-MCL are graph clustering algorithms based on a simulation of stochastic flows on the graph. The MCL process consists of two operations on a stochastic matrix: 'Expand' and 'Inflate' [17]. The Expand operation is simply $\boldsymbol{M}=\boldsymbol{M}\times \boldsymbol{M}$, and the Inflate operation raises each entry in the matrix $M$ to the inflation parameter *rp* (*rp*>1, and typically set to 2) followed by re-normalizing the sum of each column to 1. These two operations are applied in alternation iteratively, starting with the canonical flow matrix $\boldsymbol{M}$. In R-MCL [20], Expand is replaced by 'Regularize', which is $\boldsymbol{M}=\boldsymbol{M}\times \boldsymbol{M_R}$. In this operation, $M_R$ is determined by

$$M_R = Normalize\left(M_G \times P^{-b}\right) \quad (3)$$

where $\boldsymbol{P}$ is the diagonal matrix with the propensity vector along the diagonal, $b$ is a user-specified balance parameter which is used to penalize higher-propensity neighbors, and $\boldsymbol{M_G}$ is the original $\boldsymbol{M}$ [21]. The normalization operation rescales each column so that each column sums to 1. In SR-MCL [23], it brings in a process to penalize attractor nodes that it executes R-MCL repeatedly at first. Then if the count that a node has been an attractor node is $x$ in previous iterations, the inflation parameter *rp* will be replaced by $rp\times\beta^x$, where $\beta$ is a user-specified penalty ratio.

### C. Firefly Algorithm

The FA is a class of random optimization algorithm constructed by simulating behavior of the firefly group. It consists of two important elements: brightness and attractiveness [25]. The brightness reflects the advantage and disadvantage of each firefly's location and determines its moving direction. The attractiveness determines each firefly's moving distance. This algorithm can realize the target of optimization through updating the brightness and attractiveness constantly. And the mathematical mechanism of FA is described as follows.

The relative brightness of each firefly is expressed as,

$$I = I_0 \times e^{-\gamma r_{ij}^2} \quad (4)$$

where $I_0$ is the maximum of brightness (it's one firefly's absolute brightness, as $r$=0), which is related to the objective function value, the more optimal the value of objective function is, the higher the $I_0$ is. The $\gamma$ is light absorption coefficient, which is set to reflect the features that brightness will increase gradually with the increase of distance and the absorption of medium. It can be set to a constant. The $r_{ij}$ is the distance between firefly $i$ and firefly $j$.

The attractiveness of each firefly is expressed as,

$$\beta = \beta_0 \times e^{-\gamma r_{ij}^2} \tag{5}$$

where $\beta$ is the maximum of attractiveness (it's the biggest value for one firefly to attract another, as $r$=0).

If firefly $i$ is attracted to firefly $j$, then its location would be updated as,

$$x_i = x_i + \beta \times \left(x_j - x_i\right) + \alpha \times \left(Rand - 0.5\right) \tag{6}$$

where $x_i$ and $x_j$ are the location of firefly $i$ and firefly $j$. $\alpha$ is the step length of each movement and it is a constant between 0 and 1. *Rand* is a random factor which obeys uniform distribution.

The FA first distributes fireflies to the solution space randomly and each firefly has a brightness which is based on its location. One firefly with high brightness can attract the firefly with low brightness to move toward it. And the moving distance depends on the attractiveness, according to Eq. 5. The updated location of each firefly can be calculated by Eq. 6 and it adds a disturbing term to enlarge searching space and avoids immature convergence. After repetitive computation and moving, all fireflies would gather in the location of the one with maximum brightness. This is the optimal solution.

## *D. F-MCL*

In this proposed algorithm, we introduce FA to MCL in order to get a better output clustering. First we set a range of the parameter *rp* and randomly locate fireflies in this interval. Each value is a firefly's position. Then we calculate every firefly's brightness by using MCL with parameter *rp*. After running MCL completely, an *F-measure* is received, it has the same meaning with the one in part B of Section III and it is the firefly's brightness. After all fireflies' brightness has been calculated, we can get their relative brightness and attractiveness. Then we update all positions and calculate their brightness in the same way. When the repetition has met the value we set previously, this algorithm would stop and output a clustering result. For R-MCL and SR-MCL, the process of calculating brightness is similar and we get FR-MCL and FSR-MCL. Fig.1 is the flow chart of our improved algorithm.

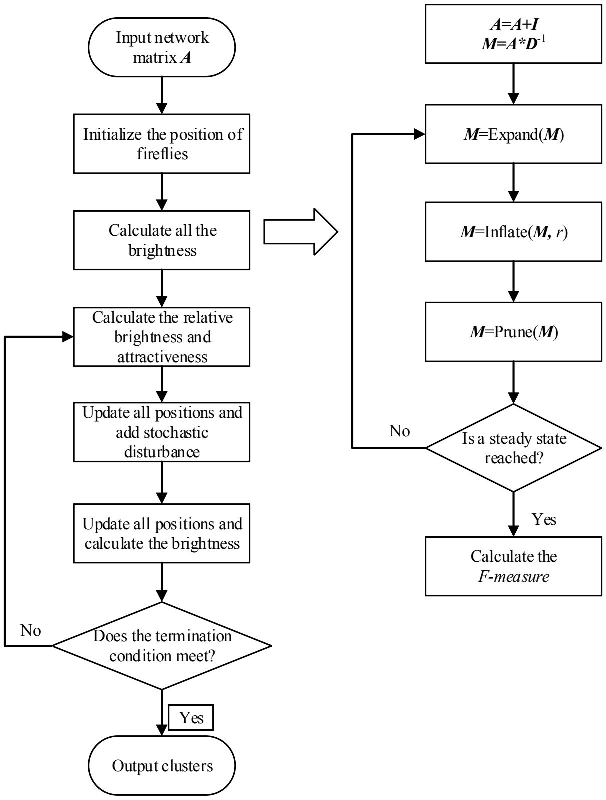
Input network matrix *A*
Initialize the position of fireflies
Calculate all the brightness
Calculate the relative brightness and attractiveness
Update all positions and add stochastic disturbance
Update all positions and calculate the brightness
Does the termination condition meet?
No
Yes
Output clusters
*A*=*A*+*I*
*M*=*A***D*$^{-1}$
*M*=Expand(*M*)
*M*=Inflate(*M*, *r*)
*M*=Prune(*M*)
Is a steady state reached?
No
Yes
Calculate the *F-measure*

Fig. 1. Flow chart of the improved algorithm

### E. Dynamic Model Construction

We use a Three-sigma principle [28] to process the static PPI network combined gene expression data. An expression value of each protein is calculated based on its gene expression data. A protein is active at a timestamp if its expression value is higher than a threshold of $\sigma$ set by the principle. And two proteins are regarded as co-expression if both of them are active at the same timestamp [37]. As we can see in Fig.2, there is a sub-network at each timestamp and an interaction won't exist if one protein of it is not active, for example, A-C and B-E in Fig.2.

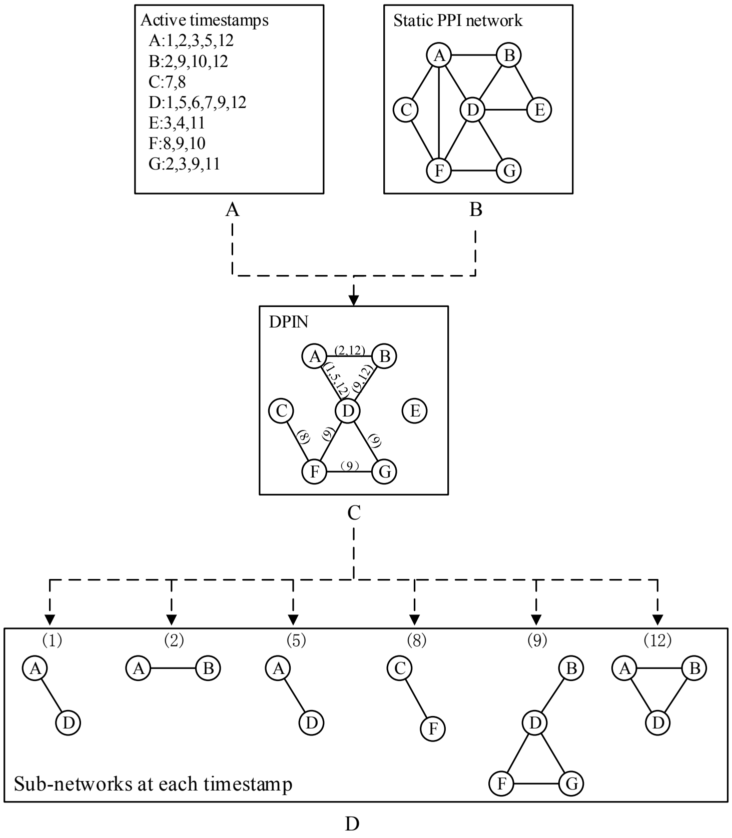

Fig. 2. DPIN construction. (A) The Active timestamps set of each protein. (B) The static PPI network. (C) The DPIN which is constructed based on (A) and (B). (D) The sub-networks contained in the DPIN

## III. Results and Discussions

### A. DPIN

We firstly downloaded PPI network data of *S.cerevisiae* from DIP database (version DIP20140427) [26]. The DIP database catalogs experimentally determined interactions between proteins. It combines information from a variety of sources to create a single, consistent set of protein-protein interactions. The data stored within the DIP database were curated, both manually by expert curators and also automatically using computational approaches that utilize the knowledge about the protein-protein interaction networks extracted from the most reliable, core subset of the DIP data. It's unweighted. After deleting self-interactions and repeated interactions, we get the static PPI network with 4995 proteins and 21554 interactions. Then we download gene expression data set GSE3431 about the logic of the yeast metabolic cycle from GEO dataset [27]. This data set includes 6777 gene products that has covered 95% proteins in the static PPI network.

Based on the model in Fig.2, we get the DPIN which contains 12 static PPI sub-networks at 12 timestamps. Different sub-network has different scale, shown in Table I.

TABLE I. THE NUMBER OF PROTEINS AND INTERACTIONS EACH SUB-NETWORK OF THE DPIN CONTAINS

| Timestamp | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Proteins | 797 | 941 | 796 | 623 | 601 | 530 |
| Interactions | 981 | 1444 | 1188 | 745 | 750 | 646 |
| Timestamp | 7 | 8 | 9 | 10 | 11 | 12 |
| Proteins | 493 | 944 | 1090 | 592 | 661 | 461 |
| Interactions | 573 | 1705 | 2185 | 856 | 974 | 526 |

These 12 sub-networks are still unweighted. Then we use the aggregation coefficient of edge [29] to be the weight of interactions, which is defined as follows:

$$C_{i,j} = \frac{\left|N_i \cap N_j\right| + 1}{\min\left(\left|N_i\right|,\left|N_j\right|\right)} \tag{7}$$

where $N_i$ and $N_j$ respectively stand for the set of neighbor nodes of nodes $i$ and $j$. And the aggregation coefficient of edge $C_{i,j}$ is a local variable describes the similarity of nodes $i$ and $j$.

The known protein functional module set includes 408 modules and 1628 proteins [30].

## *B. Metrics of Performance Evaluation*

For each clustering result, it contains many proteins and many clusters, then we record the total number of clusters, the average number of proteins in one cluster which uses the total number of proteins to divide the number of clusters, and the number of proteins that is unique.

We adopt the widely used metric *f-measure* [31], which is derived from *precision* and *recall* [31], to evaluate the accuracy of clusters produced. The *f-measure* can evaluate not only the accuracy of the clusters matching functional modules but also the accuracy of functional modules matching the clusters.

Given a clustering result $C=\{c_1, c_2, \ldots, c_k\}$ which is achieved by the algorithm, and the gold standard clusters $G=\{g_1, g_2, \ldots, g_l\}$, for any predicted cluster $c_i$ and known cluster $g_j$, the optimal score $OS$ is defined as:

$$OS\left(c_i,g_j\right) = \frac{\left|c_i \cap g_j\right|^2}{\left|c_i\right| \times \left|g_j\right|} \tag{8}$$

The bigger $OS(c_i, g_j)$ is, the more matching the two clusters $c_i$ and $g_j$ are, and the more accurate predicted cluster $c_i$ is. If it's bigger than a user-specified threshold, we can say that they are matched. Usually we set the threshold for 0.2 [28]. And if $OS(c_i, g_j)=1$, they are perfectly matched. In order to facilitate future use, we create a matrix $\boldsymbol{OS}$ to sort each $OS(c_i, g_j)$ and $\boldsymbol{OS}=\{os_{1,1},os_{1,2},\ldots,os_{1,l},os_{2,1},\ldots,os_{2,l},\ldots,os_{k,1},\ldots,os_{k,l}\}$, *that* $os_{i,j}$ *is equal to* $OS(c_i,g_j)$.

For the entire clustering result $C$ which contains all clusters whose *os* is bigger than the threshold, the *f-measure* is the harmonic mean of the *precision* and *recall.*

$$precision = \frac{\left|C \cap G\right|}{\left|C\right|} \tag{9}$$

$$recall = \frac{\left|C \cap G\right|}{\left|G\right|} \tag{10}$$

$$f - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{11}$$

## *C. Comparison with MCL, R-MCL, SR-MCL and Other State-of-the-art Algorithms*

In this section, we compare F-MCL, FR-MCL and FSR-MCL with MCODE [9], CFinder [10], CORE [11], COACH [12], RNSC [13], DPClus [14], MCL [17], RMCL [20] and SR-MCL [23]. All these algorithms are compared with each other on DIP data. We tuned each algorithm to its best parameter setting but generally found that the default parameter setting generates the best results. The parameters are shown in Table II, among them *rp* is decided by the FA algorithm and others are set based on reference [23] and [38].

TABLE II. THE PARAMETERS' SETTING

| **parameter** | **variable** | **MCL[17]** | **R-MCL[20]** | **SR-MCL[23]** | **F-MCL** | **FR-MCL** | **FSR-MCL** |
|---|---|---|---|---|---|---|---|
| Inflation parameter | *r* | 2.0 | 2.0 | 2.0 | automatic | automatic | automatic |
| Minimum of *r* | *minr* | - | - | - | 1 | 1 | 1 |
| Maximum of *r* | *maxr* | - | - | - | 4 | 4 | 4 |
| Repeating count | *t* | 30 | 30 | 30 | 10 | 10 | 10 |
| Balance | *b* | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| Penalty ratio | $\beta$ | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 |
| Number of fireflies | *m* | - | - | - | 5 | 5 | 5 |
| Light absorption coefficient | $\gamma$ | - | - | - | 1.0 | 1.0 | 1.0 |
| Maximal attractiveness | $\beta_0$ | - | - | - | 1.0 | 1.0 | 1.0 |
| Step length | $\alpha$ | - | - | - | 0.05 | 0.05 | 0.05 |
| Iteration count | *maxT* | - | - | - | 5 | 5 | 5 |

TABLE III. THE INFORMATION OF CLUSTERS PRODUCED BY SEVERAL TYPICAL ALGORITHMS

| Algorithm | # Clusters | Avg(C) | Coverage |
|---|---|---|---|
| MCODE[9] | 63 | 19.00 | 1032 |
| CFinder[10] | 609 | 6.18 | 2135 |
| CORE[11] | 772 | 5.30 | 2471 |
| COACH[12] | 903 | 8.90 | 1999 |
| RNSC[13] | 549 | 3.89 | 2133 |
| DPClus[14] | 827 | 5.28 | 3258 |
| MCL[17] | 623 | 6.57 | 4096 |
| F-MCL | 1588 | 4.62 | 2808 |
| R-MCL[20] | 848 | 5.25 | 4456 |
| FR-MCL | 1766 | 5.16 | 2813 |
| SR-MCL[23] | 1038 | 9.84 | 2047 |
| FSR-MCL | 1662 | 6.07 | 2814 |

TABLE IV. THE PERFORMANCE COMPARISON OF SEVERAL TYPICAL ALGORITHMS

| Algorithm | Precision | Recall | F-measure |
|---|---|---|---|
| MCODE[9] | 0.6364 | 0.2266 | 0.3342 |
| CFinder[10] | 0.5607 | 0.3528 | 0.4331 |
| CORE[11] | 0.47191 | 0.4813 | 0.4766 |
| COACH[12] | 0.5038 | 0.5 | 0.5019 |
| RNSC[13] | 0.4067 | 0.4696 | 0.4359 |
| DPClus[14] | 0.43 | 0.507 | 0.4653 |
| MCL[17] | 0.3569 | 0.3879 | 0.3717 |
| F-MCL | 0.6840 | 0.5540 | 0.6122 |
| R-MCL[20] | 0..3814 | 0.4866 | 0.4276 |
| FR-MCL | 0.6054 | 0.6122 | 0.6088 |
| SR-MCL[23] | 0.4443 | 0.5709 | 0.4997 |
| FSR-MCL | 0.5938 | 0.6132 | 0.6033 |

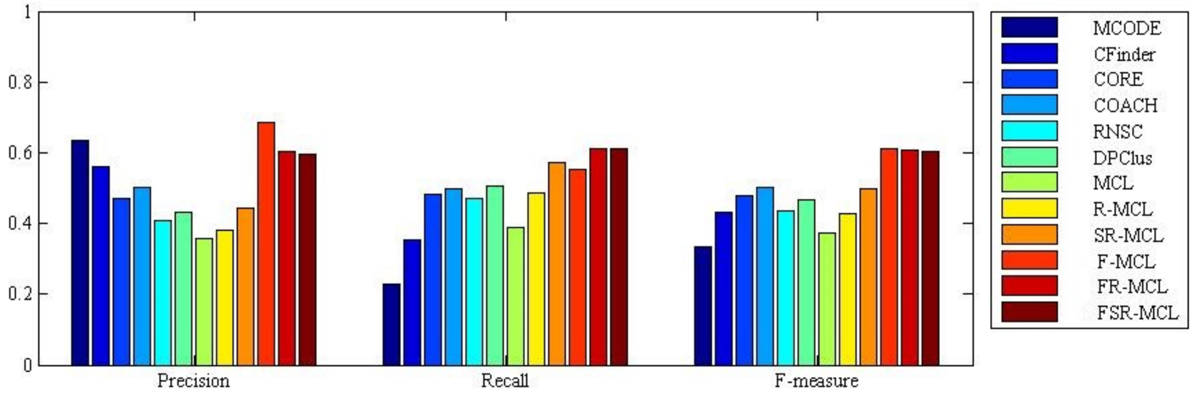

Fig. 3. The Precision, Recall and F-measure of clusters produced by several typical algorithms on DIP Example of a figure caption. *(figure caption)*

The information of all clustering results is reported in Table III and the performance of all clustering results is reported in Table IV and Fig 3. In Table III, the relevant data of the algorithms besides the previously mentioned three is got from [23] and in Table IV, the relevant data of the algorithms besides R-MCL, SR-MCL and proposed three is got from [39], the relevant data of R-MCL and SR-MCL can be seen in [23]. The number of clusters obtained by the three proposed algorithms are significantly larger than the original counterpart, that is because they are the set of results gained from 12 sub-networks without any filtration. One of the proposed algorithm, F-MCL, obtains the biggest precision and *F-measure* and the average size of its output clusters is the second smaller, only bigger than RNSC. The precision of the other two improved algorithms is bigger than all of the rest algorithms and smaller than only one algorithm MCODE. Especially, the *precision* and *F-measure* of these three proposed algorithms are all bigger than the original three, and the average size of the clustering results obtained by the proposed three algorithms is smaller than the original three. However, the performance of F-MCL, FR-MCL and FSR-MCL is similarity, the reason is that the similiar FA operation is adopted in the three algorithms.

## IV. CONCLUSION

In this article, we proposed a new approach for identifying functional modules in DPIN—F-MCL. In DPIN, all sub-networks have different scale, and though MCL is effective on clustering biological networks, its parameters are user-specified and cannot deal with all network scale very well. Then we use FA to optimize it and adjust its parameter automatically. The F-MCL's clustering result is more accuracy than others and is more closer to the golden standard clustering result in the property of average size. We empirically find that F-MCL outperforms a range of existing algorithms in terms of its precision on identifying functional modules.

## ACKNOWLEDGMENT

This paper is supported by the National Natural Science Foundation of China (61100164, 61173190), Scientific Research Start-up Foundation for Returned Scholars, Ministry of Education of China ([2012]1707) and the Fundamental Research Funds for the Central Universities, Shaanxi Normal University (GK201402035, GK201302025).

## REFERENCES

[1] S.D. Patternson and R.H. Aebersold, “Proteomics: The first decade and beyond,” Nature Genetics. 2003, 33, 311-323.

[2] A. Zhang, Protein interaction networks. New York, USA: Cambridge University Press. 2009.

[3] A. A. Cohen, N. Geva-Zatorsky, E. Eden, M. Frenkel-Morgenstern, I. Issaeva, *et al.*, “Dynamic proteomics of individual cancer cells in response to a drug,” Science 2008, 322, 1511–1516.

[4] J. Wang, X. Peng, W. Peng, and F.X. Wu, “Dynamic protein interaction network construction and applications,” Proteomics. 2014, 14, 338-352.

[5] D.J. Watts and S.H. Strogatz, “Collective dynamics of ‘small-world’ networks,” Nature 1998, 393, 440–442.

[6] S. Antonio and O’M. Paul, “Small-world network approach to identify key residues in protein–protein interaction,” PROTEINS: Structure, Function, and Bioinformatics. 2005, 58(3), 672–682.

[7] M. Girvn and M.E.J. Newman, “Community structure in social and biological networks,” Proceedings of the National Academy of Science. 2002, 99(12), 7821-7826.

[8] M.E.J. Newman, “Fast algorithm for detecting community structure in networks,” Physical Review E. 2004, 69(6), 066133.

[9] G.D. Bader and C.W. Hogue, “An automated method for finding molecular complexes in large protein interaction networks,” BMC Bioinformatics. 2003, 4, article 2.

[10] B. Adamcsek, G. Palla, I.J. Farkas, I. Derényi, and T. Vicsek, “CFinder: Locating cliques and overlapping modules in biological networks,” Bioinformatics. 2006, vol. 22, no. 8, pp. 1021-1023.

[11] H.C.M. Leung, Q. Xiang, S.M. Yiu, and F.Y.L. Chin, “Predicting protein complexes from ppi data: a core attachment approach,” J. Comput. Biol., 2009, 16, 133–144.

[12] M. Wu, X.L. Li, C.K. Kwoh, and S.K.Ng, “A core-attachment based method to detect protein complexes in ppi networks,” BMC Bioinformatics, 2009, 10, 169.

[13] A.D. King, N. Pržulj, and I. Jurisica, “Protein complex prediction via cost-based clustering,” Bioinformatics. 2004, 20, 3013-3020.

[14] M. Altaf-Ul-Amin, Y. Shinbo, K. Mihara, K. Kurokawa, and S. Kanaya, “Development and implementation of an algorithm for detection of protein complexes in large interaction networks,” BMC Bioinformatics. 2006, 7, 207.

[15] M. Li, J. Chen, J. Wang, B.Hu, and G.Chen, “Modifying the DPClus algorithm for identifying protein complexes based on new topological structures,” BMC Bioinformatics. 2008, 9, 398.

[16] Y.Y. Ahn, J.P. Bagrow, and S. Lehmann, “Link communities reveal multiscale complexity in networks,” Nature 2010, 466, 761–764.

[17] S.V. Dongen, “Graph clustering by flow simulation,” PhD thesis, Center for Math. and Computer Science (CWI), 2000.

[18] S. Brohee and J.V. Helden, “Evaluation of clustering algorithms for protein–protein interaction networks,” BMC Bioinformatics. 2006, 7, 488.

[19] J. Vlasblom and S.J. Wodak, “Markov clustering versus affinity propagation for the partitioning of protein interaction graphs,” BMC Bioinformatics. 2009, 10,99.

[20] V. Satuluri and S. Parthasarathy, “Scalable graph clustering using stochastic flows: Applications to community discovery,” In KDD’09, pages 737-746, New York, NY, USA, 2009. ACM.

[21] V. Satuluri, S. Parthasarathy, and D. Ucar, “Markov clustering of protein interaction networks with improved balance and scalability,” ACM-BCB 2010, Pages 247-256, Niagara Falls, NY, USA.

[22] D. Chakrabarti and C. Faloutsos, “Graph mining: Laws, generators, and algorithms,” ACM Comput. Surv., 2006, 38(1):2.

[23] Y.K. Shih and S. Parthasarathy, “Identifying functional modules in interaction networks through overlapping Markov clustering,” Bioinformatics. 2012, 28, 473-479.

[24] X. Lei, J. Tian, L. Ge, and A. Zhang, “The clustering model and algorithm of PPI network based on propagating mechanism of artificial bee colony,” Information Science. 2013, 247, 21-39.

[25] X. Yang, Nature-Inspired metaheuristic algorithms. Luniver Press, 2008, 83-96.

[26] L. Xenarios, L. Salwinski, X. Duan, P. Higney, S.Kim, *et al.*, “DIP: the database of interaction proteins: a research tool for studying cellular networks of protein interactions,” Nucl. Acids Res. 2002, 30, 303-305.

[27] B.P. Tu, A. Kudlicki, M. Rowicka, S.L. Mcknight, “Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes,” Science 2005, 310(5751):1152-1158.

[28] J. Wang, X. Peng, M. Li, and Y. Pan, “Construction and application of dynamic protein interaction network based on time course gene expression data,” Proteomics. 2012, 00, 1–13.

[29] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, “Defining and identifying communities in networks,” Proceeding of the National Academy of Sciences of the USA 101 (9) (2004) 2658–2663.

[30] S. Pu, J. Wong, B. Turner, E. Cho, and S.J. Wodak, “Up-to-date catalogues of yeast protein complexes,” Nucleic Acids Research. 2009, Vol. 37, No. 3 825–831.

[31] Y.R. Cho, H. Hwang, M. Ramanathan, and A. Zhang, “Semantic integration to identify overlapping functional modules in protein interaction networks,” BMC Bioinformatics. 2007, 8:265.

[32] J. Wang, M. Li, J. Chen, and Y. Pan, “A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks,” IEEE/ACM Trans Comput Biol Bioinform. 2011, 8(3): 607-620.

[33] X. Lei, X. Huang, L. Shi, and A. Zhang. “Clustering PPI data based on improved functional-flow model through quantum-behaved PSO,” Int. J. Data Mining and Bioinformatics. 2012, Vol. 6, No. 1, pp.42–60.

[34] X. Lei, S. Wu, L. Ge, and A. Zhang. “Clustering and overlapping modules detection in PPI network based on IBFO,” Proteomics. 2013, 13, 278–290.

[35] Montanez, George, Cho, and Young-Rae, “Predicting false positives of protein-protein interaction data by semantic similarity measures,” Current Bioinformatics, 2013, 8(8): 339-346.

[36] M. Li, R. Zheng, H. Zhang, J. Wang, and Y. Pan, “Effective identification of essential proteins based on priori knowledge, network topology and gene expressions,” Methods, 2014. doi:10.1016/j.ymeth.2014.02.016

[37] N. Du, Y. Zhang, K. Li, J. Gao, S.D. Mahajan, *et al.*, “Evolutionary Analysis of Functional Modules in Dynamic PPI networks,” ACM-BCB, 2012.

[38] S. Łukasik and S. Żak, “Firefly Algorithm for Continuous Constrained Optimization Tasks” Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems Lecture Notes in Computer Science Volume 5796, 2009, pp 97-106.

[39] X. Lei, C. Ying, F. Wu and J. Xu, “Clustering PPI Data through inproved Synchronization-Based Hierarchical Clustering Method” 10th International Symposium, ISBRA 2014.