Hindawi Publishing Corporation
Journal of Biomedicine and Biotechnology
Volume 2011, Article ID 480294, 14 pages
doi:10.1155/2011/480294

Research Article

HKC: An Algorithm to Predict Protein Complexes in Protein-Protein Interaction Networks

Xiaomin Wang,[1] Zhengzhi Wang,[1] and Jun Ye[2,3]

[1] *Institute of Mechanical Engineering and Automation, National University of Defense Technology, Changsha 410073, China*
[2] *Department of Software Engineering, Jiangnan Institute of Computing Technology, Wuxi 214083, China*
[3] *School of Computer, National University of Defense Technology, Changsha 410073, China*

Correspondence should be addressed to Xiaomin Wang, wangxiaomin@nudt.edu.cn

Received 23 May 2011; Accepted 24 August 2011

Academic Editor: Paul W. Doetsch

With the availability of more and more genome-scale protein-protein interaction (PPI) networks, research interests gradually shift to Systematic Analysis on these large data sets. A key topic is to predict protein complexes in PPI networks by identifying clusters that are densely connected within themselves but sparsely connected with the rest of the network. In this paper, we present a new topology-based algorithm, HKC, to detect protein complexes in genome-scale PPI networks. HKC mainly uses the concepts of highest k-core and cohesion to predict protein complexes by identifying overlapping clusters. The experiments on two data sets and two benchmarks show that our algorithm has relatively high F-measure and exhibits better performance compared with some other methods.

1. Introduction

With the development of high-throughput methods (such as mass spectrometry [1] and two-hybrid [2]), more and more genome-scale protein-protein interaction (PPI) networks are now available, enabling us to systematically analyze the behaviors and properties of biological molecules. Among the various researches on PPI networks, a key topic is to predict protein complexes in PPI networks. A protein complex is a group of proteins that interact with each other at the same time and place, forming a single multimolecular machine [3]. These complexes are a cornerstone of many biological processes. PPI networks are modular [3, 4] and contain modules that are densely connected within themselves but sparsely connected with the rest of the network. These modules are called cluster and they may represent protein complexes [5, 6]. Thus, protein complexes can be detected by identifying clusters from PPI networks. Detecting protein complexes in PPI networks is of vital importance to the understanding of the structural and functional properties of PPI networks and can also help to predict the function of unknown proteins.

Due to the high level of noise as well as the topological features of PPI networks, traditional clustering techniques in a metric space cannot successfully predict protein complexes in PPI networks [3], thus various graph analysis approaches have been proposed to solve this problem. These approaches can be classified into the following three types. (1) Agglomerative methods: Bader and Hogue [6] proposed a graph theoretic clustering algorithm to detect molecular complexes in PPI networks. The method was based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed protein to isolate the dense regions according to given parameters. The problem of this method is that using only vertex weighting, it would find sparsely connected subgraphs (such as a rope-like subgraph or a mixed subgraph consisting of several connected clusters) instead of dense clusters. (2) Graph partition methods: King et al. [7] partitioned networks and found an approximate optimal solution in the space of partitions using a cost-based local search algorithm. This technique was nondeterministic and got different result in each run, and it consumed huge space. Chen and Yuan [8] extended a betweenness-based partition algorithm (Girvan-Newman algorithm [9],

GN for short) and used it to partition PPI networks into subgraphs and then obtained function modules by filtering these subgraphs. The chief drawback of GN algorithm is it is time consuming. Generally, graph partition methods can only find nonoverlapping clusters, while protein complexes tend to overlap with each other. (3) Methods based on clique (see Section 2.2 Concepts): Palla et al. [10] suggested clique percolation method (CPM) to find overlapping communities. The core concept of CPM is k-clique community which is defined as the union of all k-cliques (complete subgraphs of size k) that can be reached from each other through a series of adjacent k-cliques (where adjacency means sharing k-1 nodes). Some other researchers [11, 12] predicted protein complexes by identifying cliques or near-cliques in PPI networks. Clique-based approaches were too stringent with the topological structure and cannot detect protein complexes with other types of topological structure.

In this paper, we present a new topology-based algorithm, HKC (as it mainly uses two important concepts, highest k-core and cohesion), to detect protein complexes in genome-scale PPI networks. Our algorithm uses the concepts of highest k-core, and cohesion to predict protein complexes by identifying overlapping clusters. It first calculates the score for each node in the PPI network based on the concept of highest k-core then uses the nodes with high scores and high degrees as seeds; from each seed, gets a core according to the corresponding highest k-core of the direct neighborhood of the seed and expands this core to include nodes which are highly possible to form a cluster based on the criteria of node score and cohesion; finally, protein complexes can be predicted by filtering all the found clusters according to predefined features. We apply HKC on two data sets (MIPS and SGD-MC data set) and evaluate the results on two benchmarks (complexcat and Gavin benchmarks). The experiments show that our algorithm has relatively high precision and recall, that is, most of the predicted clusters match well with known protein complexes, and at the same time most of the known protein complexes have been recalled. HKC exhibits better performance compared with some other methods, and besides, it is general and can be used in any type of biological interaction networks and even in nonbiological networks.

2. Materials and Method

2.1. Materials. Protein-protein interactions of the model organism Saccharomyces cerevisiae (yeast) has been studied thoroughly, and the data of yeast protein complexes is the most comprehensive so far, so we test HKC on the yeast PPI data and use yeast complex data to validate its effectiveness.

We use the following two data sets as the input network: (1) MIPS data set, it is processed based on the Munich Information Center for Protein Sequences (MIPS) [13] yeast PPI data set (containing 15,456 records) downloaded from the Comprehensive Yeast Genome Database (CYGD) [14]; after removing those repetitive interactions and taking no account of the edge direction, the final input PPI network contains 4,554 proteins and 12,526 interactions. (2) SGD-MC data set, which is downloaded from the Literature Curation Data in SGD database [15]; the original data set contains 252247 records, including two types of interactions, high-throughput and manually curated interactions, and we only use the manually curated interactions. After deleting all repetitive interactions and taking no account of the edge direction, the final SGD-MC data set contains 4,448 proteins and 29,068 interactions.

To evaluate the algorithm, we collect two data sets as benchmarks: (1) complexcat benchmark, it is obtained by processing the MIPS yeast protein complex catalog in CYGD [14]. The MIPS yeast protein complex catalog was last modified in 2006, and has been manually curated from the literature; therefore, it is more realistic than other data obtained by high-throughput methods and has been used in many researches because of its quality [6, 7, 16]. However, it is not proper to use this data set directly as the benchmark, since it contains many complexes composed of a single protein which do not fit the definition of clusters and also contains the complexes by Systematic Analysis [17, 18] which are not so reliable as results by small-scale experiments. After removing those complexes by Systematic Analysis (the type of 550) and the complexes consisting of only one protein, the final obtained complexcat benchmark contains 217 protein complexes. (2) Gavin benchmark, it is processed from the experiment results by Gavin et al. [19]. They use affinity purification and mass spectrometry to get 491 complexes that differentially combine with additional attachment proteins or protein modules to enable a diversification of potential functions. We adopt the core proteins (those present in 2/3 of the isoforms) of the 491 complexes and 36 additional known complexes, and after removing those of size less than 3, finally we get 204 protein complexes in Gavin benchmark.

2.2. Concepts

2.2.1. PPI Network. PPI networks can be intuitively modeled as a static graph $G = (V, E)$, where V is the set of nodes (proteins), and E is the set of edges (protein-protein interactions).

An undirected edge is drawn between each pair of nodes for which there is evidence of a protein-protein interaction.

2.2.2. Basic Concepts in Graph Theory

Degree. The degree of the node n is the number of edges attached to it and is denoted as $\deg(n)$.

Graph Density. There is no standard graph theory definition of density, but definitions are normally based on the connectivity level of a graph [6]. For undirected simple graphs, the graph density is defined as the quotient of the number of real edges in the graph divided by the number of all possible edges:

$$\mathrm{den}(G) = \frac{|E|}{|V| * (|V| - 1)/2}, \quad (1)$$

where $\mathrm{den}(G)$ is the density of graph G, $|E|$ is the number of real edges in G, and $|V|$ is the number of nodes in G.

For undirected graph with loops (a loop is the edge that connects a node and the node itself), the number of all possible edges is $|V| * |V|/2$, thus the density of graph G is defined as

$$\text{den}(G) = \frac{|E|}{|V| * |V|/2}. \quad (2)$$

So, the range of graph density is between 0 and 1.

Clique. A clique is a fully connected subgraph, that is, a set of nodes that are all neighbors of each other [20]. For instance, Figure 1(c) is a clique consisting of 4 nodes.

k-Core. A k-core of a graph is a maximal subgraph such that each node in the subgraph has at least degree k and is denoted as $kc(G)$. For example, in Figure 1, the graph in (b) is a 2-core of the graph in (a).

Highest k-Core. The highest k-core of a graph is the one with the maximal k value among all the k-cores and is denoted as $hkc(G)$. It is the central most densely connected subgraph. The highest k-core of G can be found in the following way [6]: suppose the lowest degree of nodes in G is l, delete all nodes with degree l, if all remaining nodes have a least degree l_1 ($l_1 > l$), we will get the l_1-core of G; if some of the remaining nodes have degree lower than or equals to l, continue delete all nodes with the least degree until all remaining nodes have a degree higher than l, or until all nodes have been deleted. In this way we can find all k-cores and the one with the maximal k value is the highest k-core. For example, Figure 1(a) shows a graph G, if we delete the node of degree 1 (i.e., node 6), we can obtain a subgraph in which each node has a least degree 2, that is, the 2-core of G, as shown in Figure 1(b); if we continue delete the node of degree 2 (node 1), we can get a subgraph in which each node has a least degree 3, the 3-core of G, as shown in Figure 1(c), and it is also the highest k-core of graph G.

2.2.3. Cohesion. During the expansion of a core, only based on the score information, we cannot efficiently decide whether a node should be included into the core. We define a new concept *cohesion* to measure the connectivity between a node n and an existing cluster c, and we denote it as $\text{co}(n, c)$. Cohesion is calculated in the following way:

$$\text{co}(n, c) = \frac{E_{nc}}{N_c}, \quad (3)$$

where E_{nc} is the number of edges between node n and cluster c, and N_c is the number of nodes in cluster c. $\text{co}(n, c)$ is a real number between 0 and 1. The larger $\text{co}(n, c)$ is, the tighter node n connects to cluster c, and the more likely they belong to a larger cluster. Therefore, cohesion can be used as a criterion during the core expansion process. A node can only be included into a core when the cohesion between this node and the core is greater than a specified threshold.

2.3. The Algorithm. The algorithm HKC consists of the following three steps: scoring, cluster finding, and filtering.

2.3.1. Scoring. The first step of HKC is to score all nodes in the PPI network. For each node, firstly we find the highest k-core of its direct neighborhood (the subgraph consisting of all nodes connecting to the node, including the node itself), which we denote as H. Then we score H using the properties of highest k-core. Larger and denser cores will get higher scores, and it is computed in the following way:

$$\text{score}(H) = N_H * \text{den}(H) * k_{\max}, \quad (4)$$

where N_H denotes the number of nodes in H, $\text{den}(H)$ refers to the density of H, $k_{\max}$ is the maximal k value corresponding to the highest k-core H, and the larger these three values are, the larger and denser the corresponding highest k-core is.

As highest k-core is the most densely connected central core in the local area, in order to make sure that the node score gives better reflection of the connectivity in the local area, we assign score (H) to each node in H. For node n, it may be contained in more than one highest k-core, thus it may be given more than one score, and the final score of this node is defined as the maximal score of all highest k-cores in which node n is contained (see the pseudocode in line 14–16 in Pseudocode 1).

$$\text{score}(n) = \max\{\text{score}(H_i) \mid \forall H_i, n \in H_i\}. \quad (5)$$

In this way, we can make sure that all nodes in densely connected subgraphs will have high score and those with little neighbors will have low scores, and therefore we can distinguish the nodes in clusters from those not in clusters with the help of node score.

2.3.2. Cluster Finding. The second step of this algorithm is to find clusters in the scored graph. The process of finding a cluster is as follows: first choose the seed, then obtain the core based on the seed, and then expand the core to include the noncore nodes. The final cluster obtained in this way is thus a circular subgraph with densely connected core and less densely connected noncore, as shown in Figure 2, which fits our expectation of the topological structure of protein complexes.

The detailed process of finding a cluster can be divided into the following three steps.

(1) Choose the seed. As node scores represent the local density, if a node has very high score, it must have a very dense neighborhood; besides, for the nodes with the same score value (e.g., according to our scoring scheme, nodes in the same highest k-core may have the same score value), the one with the highest degree can be deemed as the most densely connected node among them. Therefore, among all the unseen nodes with the highest score we choose the one with the highest degree as the seed (see the pseudocode in line 2–4 in Pseudocode 2), and this way of seed selection can insure that the seed is in the center of the corresponding cluster.

(2) Get the core based on the selected seed. First get the highest k-core of the direct neighborhood of the seed,

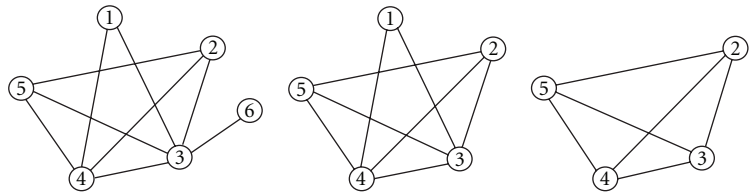

Figure 1: The illustration of k-cores of a graph. (a) Graph G; (b) 2-core of G; (c) 3-core of G, it is also the highest k-core of G.

```
Input:
   PPI network (an indirect simple graph): G = (V, E)
   Node score threshold: T1 and T2
   Cohesion threshold: T3
Output:
   The predicted protein clusters: Clusters
Call Scoring
Call ClusterFinding
Call Filtering

// step1: scoring
Procedure Scoring
   for all node n in V do
      score(n) = 0
   end for
   for all node n in V do
      compute the degree of n
      N = neighborhood (n) // neighborhood returns the
                           // direct neighborhood of n (including n)
      H = hkc(N) // hkc returns the highest k-core of N
      k_max = the maximal k value corresponding to H
      N_H = the number of nodes in H
      den(H) = the density of H
      score(H) = N_H * den(H) * k_max
      for all node m in H do
         score(m) = max{score(H), score(m)}     // compute the score of node
      end for
   end for
end procedure
```

Pseudocode 1: Procedure Scoring.

and after removing all nodes that have been already seen and those with too low scores we can get the core (see the pseudocode in line 8–13 in Pseudocode 2). In order to avoid repetitive computation, all nodes in the core (including the seed) would be marked seen and cannot be used as the core or the seed of another cluster.

(3) Expand the core. As the node score indicates the local connectivity in the neighborhood of this node, it can be used as a criterion during core expanding. When expanding a core, in order to guarantee the local density, nodes with too low scores could not be included into the core; on the other side, to avoid excessively expanding a little core to include a denser and larger cluster, nodes with extortionate scores could neither be included into the core. Furthermore, if we use node score threshold as the only criterion to decide whether a node should be included into the core, connected nodes with similar scores will be detected as one cluster when they do not actually make up a densely connected subgraph, such as those with rope-like shape. So we adopt cohesion as another criterion. The process of expanding a core c is as follows (see the pseudocode in line 15–26 in Pseudocode 2): for any unexpanded node n, looking in its neighbors for nodes such that satisfy the following conditions: (1) the score of the node is greater than or equal to score$(n) * T_1$ and less than or equal to score$(n) * T_2$; (2) the cohesion between the node and the core is greater than or equal to the threshold T_3. Where T_1 and T_2 are the node scores

```
Input:
   PPI network (an indirect simple graph): G = (V, E)
   Node score threshold: T1 and T2
   Cohesion threshold: T3
Output:
   The predicted protein clusters: Clusters
Call Scoring
Call ClusterFinding
Call Filtering

// step2: Cluster finding
Procedure ClusterFinding
   S = Sort(V) // sort all nodes descendingly according to node score, for nodes
               // with the same score put the one with higher degree ahead
   n = the first node in S
   Clusters = a empty list of cluster
   whilen is not the last node in S do
      if  n is not seen then
         C = hkc(neighborhood(n))
         for all node q in C do
            if q is seen or score(q) < score(n)*T1 then
               remove q in C
            end if
         end for
         mark all nodes in C seen
         for all node m in C do
            if m is expanded then continue
            for all i in neighbors of m do
               co = the cohesion between i and C
               if score(i) >= score(m)*T1 and
                  score(i) <= score(m)*T2 and
                  co >= T3 then
                     add i to C
               end if
            end for
            mark m expanded
         end for
         add C to Clusters
      end if
      n = the next node of n in S
   end while
end procedure
```

Pseudocode 2: Procedure Cluster Finding.

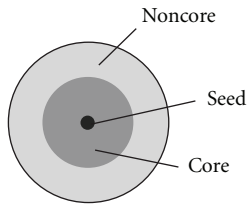

Figure 2: The illustration of a typical cluster with densely connected core and less densely connected non-core.

lower bound and upper bound, respectively, T_1 is a real number between 0 and 1, and T_2 is an integer greater than 1; T_3 is the cohesion threshold, a real number ranging from 0 to 1. Add the found nodes to the core and continue to expand the core until all nodes (including the new added nodes) in the core have been expanded.

After finding a cluster using the above method, choose another seed and repeat the above cluster-finding process until no satisfying nodes can be considered as seed, in this way we can get all clusters in the PPI network.

It is worth noting that while expanding a core, we never consider whether a node has been seen or not. As a result, a noncore part of a cluster can include nodes that have been seen as the core of another cluster, that is to say the different clusters detected by HKC can have overlaps between cores and noncores. In this way, we can find overlapping clusters, which better coincide with the fact that different protein complexes have overlaps.

```
Input:
   PPI network (an indirect simple graph): G = (V, E)
   Node score threshold: T1 and T2
   Cohesion threshold: T3
Output:
   The predicted protein clusters: Clusters
Call Scoring
Call ClusterFinding
Call Filtering

// step3: Filtering
Procedure Filtering
   for all cluster c in Clusters do
      if the size of c is less than 3 then
         remove c in Clusters
      end if
   end for
   for all cluster c in Clusters do
      den(c) = the density of c
      s = size of c
      score(c) = den(c) * s     // compute the score of cluster
   end for
   for all cluster c1, c2 in Clusters do
      o = the overlap ratio of c1 and c2
      if o > 0.95 then
         if score(c1) > score(c2) then delete c2 in Clusters
         else delete c1 in Clusters
         end if
      end if
   end for
    Clusters = sort Clusters descendingly according to cluster scores
end procedure
```

Pseudocode 3: Procedure Filtering.

2.3.3. Filtering. The clusters found in step two contain many clusters of size one or two, and these little clusters are insignificant, since they can be obtained by randomly select nodes in a PPI network. Thus we filter out the clusters that contain less than three nodes. Score all clusters using the product of cluster density and cluster size, and larger and denser clusters will get higher scores (see the pseudocode in line 7–11 in Pseudocode 3). As the algorithm allows overlaps between cores and non-cores, the results in step two may contain highly similar clusters, which must be filtered in the postprocessing. We use overlap ratio (OR, see Section 3.1 for more details) to measure the similarity between clusters; compare each two clusters, when their overlap ratio is higher than 0.95, delete the one with lower score (see the pseudocode in line 12–19 in Pseudocode 3). Finally, rank all remaining clusters in descending order according to cluster score.

2.3.4. Pseudocode

2.4. Implementation. The algorithm has been implemented in Java and we plan to convert it into a Cytoscape plug-in. Now the source code of the algorithm is available freely for noncommercial purposes upon request. All maps of networks were performed by Cytoscape [21].

3. Results and Discussions

3.1. Evaluation of the Algorithm. To evaluate the performance of our algorithm, we compare the predicted clusters with the protein complexes in two different benchmarks: complexcat and Gavin benchmarks. Each of the predicted clusters is compared with the benchmark complexes. The similarity between a predicted cluster and a benchmark complex is measured by overlap ratio (OR), which is defined as follows:

$$\mathrm{OR} = 2 * \frac{O}{(C_1 + C_2)}, \tag{6}$$

where O is the number of proteins shared by a predicted cluster and a benchmark complex, C_1 is the number of proteins in the predicted cluster and C_2 is the number of proteins in the benchmark complex. The scope of OR is between 0 and 1. $\mathrm{OR} = 0$ means the predicted cluster has no proteins in common with the benchmark complex; $\mathrm{OR} = 1$ means it is perfectly matched with the benchmark complex. The higher OR is, the more biologically meaningful the detected cluster would be. A detected cluster can be deemed as being matched with a benchmark complex only when their overlap ratio is above a given threshold. And we call a cluster an *effective cluster* as long as it has at least one benchmark complex matching with it. In the same way, a

matched complex refers to the benchmark complex that has a least one detected cluster that matching with it. A rational OR threshold should ensure that the detected cluster shares a large proportion of proteins with the matching benchmark complex, and meanwhile it could not be too stringent. In this paper, we adopt 0.4 as the OR threshold.

In [6] they use overlap score, defined as $O^2/(C_1 * C_2)$, to determine how effectively a predicted cluster matched to a known complex in the benchmark set, and it is assumed that a predicted cluster is more or less matches a known complex when is its overlap score is above 0.2. Here, we did not adopt this scoring scheme, because it is biased, that is, it would get a relatively high score when a small predicted cluster matching with a large known complex or a large predicted cluster matching with a small complex. For example, when a predicted cluster of size 2 shares 2 proteins with a known complex of size 10, its overlap score equals 0.2 and would thus be considered as matched with the known complex; actually, it is not so appropriate to deem such a small cluster as matching a known complex much larger that it. However, our definition of overlap ratio is more balanced and only gives high score when the matching cluster and complex have similar sizes. As for the above example, its overlap ratio is only 0.33, less than the threshold 0.4, and would not be deemed as a match.

To compare the performance of different algorithms, we define three criteria: *precision*, *recall,* and *F-measure*, defined as the following formulas:

$$\begin{aligned} \text{precision} &= \frac{\text{EC}}{\text{AC}}, \\ \text{recall} &= \frac{\text{MC}}{\text{BC}}, \\ F\text{-measure} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \end{aligned} \tag{7}$$

where EC is the number of effective clusters found by the algorithm, AC is the number of all clusters predicted by the algorithm, MC is the number of matched complexes in the benchmark set, and BC is the total number of benchmark complexes. Note that according to the overlap score threshold, EC may not equal MC, since one predicted cluster may match with several benchmark complexes as long as their overlap scores are higher than the given threshold, and in the same way one benchmark complex may correspond with several predicted complexes. *Precision* describes the accuracy of the algorithm result; *recall* denotes the percentage of benchmark complexes that are recovered by the algorithm. *F-measure*, which is the harmonic mean of *precision* and *recall*, shows a good balance of *precision* and *recall*, and thus can be used to measure the overall performance of algorithms.

3.2. Experiments and Comparison. We, respectively, use MIPS and SGD-MC data sets as the input PPI network and run HKC with 120 groups of parameter combination. The range of T_1 is between 0.3 and 0.7, with the step of 0.1, and the range of T_2 is between 5 and 20, with the step of 5, and the range of T_3 is between 0.4 and 0.9, with the step of 0.1. We evaluate the results using the two benchmarks: complexcat and Gavin benchmarks, and then choose the optimized parameters which enable *F-measure* to get the highest value. The best result and the corresponding optimized parameters for HKC are shown in Table 1.

To show the influence of different parameters on the algorithm performance, we draw the plot of average *F-measure* versus T_1, T_2, and T_3, respectively, as shown in Figure 3. Note that here the *F-measure* in y-axis is the average value of all *F-measures* with one parameter specified among the 120 groups of experiment results evaluated by the complexcat benchmark. From Figure 3, we can see that among the three parameters T_3 has the greatest influence on average *F-measure*, and for MIPS data set, the average *F-measure* gets the maximum value when $T_3 = 0.5$, while for SGD-MC data set, the average *F-measure* gets the maximum value when $T_3 = 0.7$. This is understandable, as the SGD-MC network (which contains 4,448 proteins and 29,068 interactions) is much denser than the MIPS network (which contains 4,554 proteins and 12,526 interactions), and during the core expansion process the nodes in SGD-MC network would have higher cohesion with the core than the nodes in MIPS network. Therefore, for SGD-MC network when expanding the core, the cohesion threshold T_3 should be higher than that for MIPS network. Furthermore, the figures show that a good range for T_1 is in the middle, between 0.4 and 0.6, the best value for T_2 is 10, and for T_3 the best range is between 0.5 and 0.8.

To show the performance of HKC, we compare it with MCODE [6], as shown in Table 1. We run the MCODE plugin in Cytoscape with 840 parameter combinations (the same with that used in [6]) on MIPS and SGD-MC data sets respectively, and then use the two benchmarks to evaluate the results. The optimized parameters (see Table 1) are chosen based on the highest *F-measure*. The result of HKC is the best one in 120 groups of parameters, and the corresponding optimized parameters are shown in Table 1. From this table, we can see that for MIPS data set, the recall of HKC is **0.429** corresponding to the complexcat benchmark, considerably higher than that of MCODE (**0.194**); for SGD-MC data set, HKC can recall as high as **58**% of protein complexes in complexcat benchmark, notably higher than that of MCODE (around **22**%). Experiment results show that whichever benchmark is adopted, for both MIPS and SGD-MC data set, the *recall* and *F-measure* of HKC are remarkably higher than that of MCODE, and the overall performance is substantially improved.

As shown in Figure 4, whatever the OR threshold is, HKC can extract much more effective clusters than MCODE in MIPS data set, and also the number of matched complexes by HKC is much higher than that by MCODE.

To show the overall performance improvement of our algorithm, we also plot *precision* versus *recall* for all results with different parameters in Figure 5. As can be seen from the figure, for all four cases, the data points resulted by HKC are located in the upper right portion of the plot, corresponding to high values of *F-measure*, while most of the data points resulted by MCODE are located in the lower left part of the plot. The figure illustrates that both precision and recall of

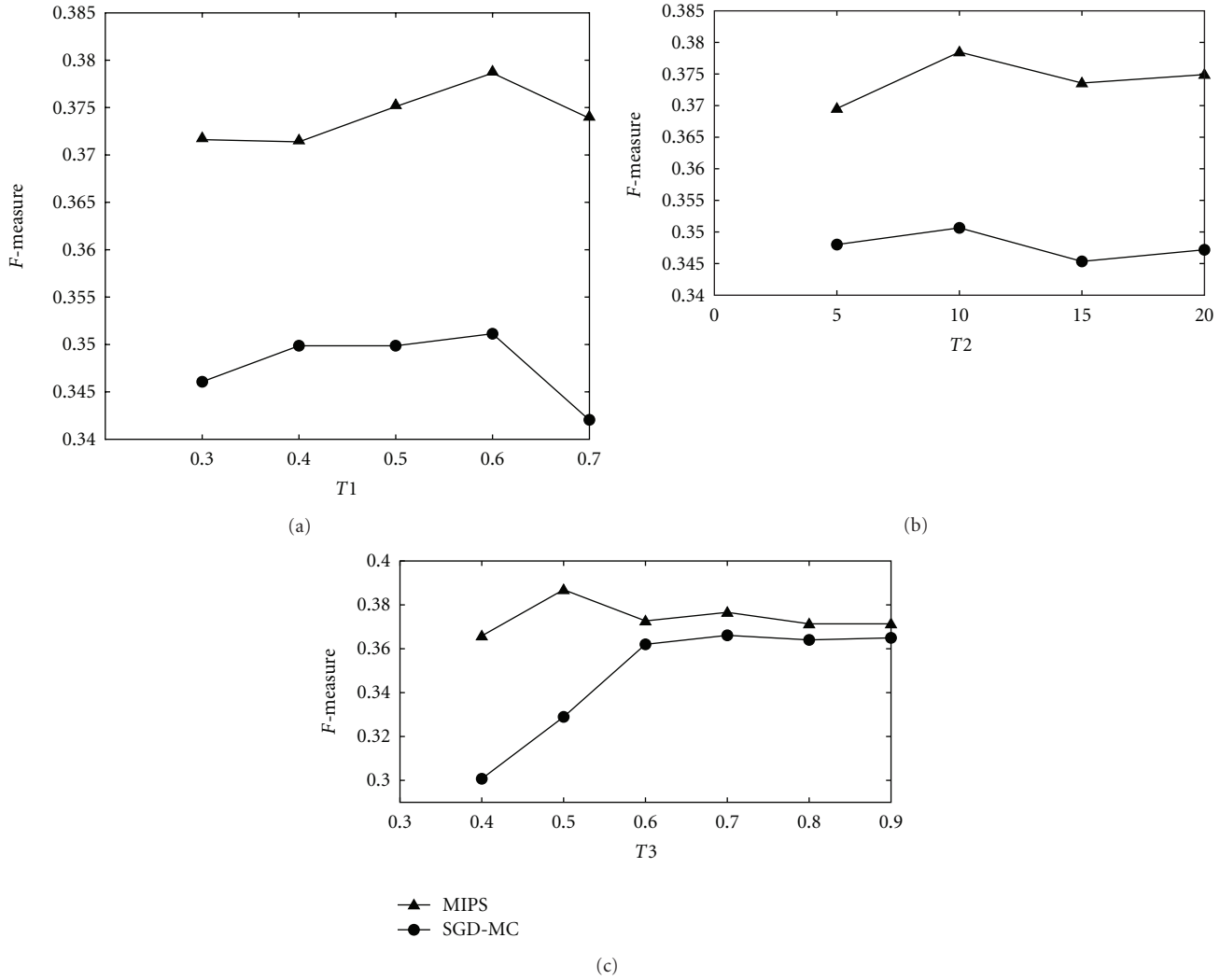

Figure 3: The influence of different parameters on algorithm performance. Note the *F-measure* in *y*-axis is the average value of all *F-measures* with one parameter specified among the 120 groups of experiment results evaluated by the complexcat benchmark, triangles mark the result on MIPS network, and circles mark the result on SGD-MC network.

Table 1: Comparison with MCODE. *P*, *R*, and *F* stand for *precision*, *recall,* and *F-measure,* respectively, and their definitions are given in Section 3.1. MIPS data set contains 4,554 proteins and 12,526 interactions, and SGD-MC data set contains 4,448 proteins and 29,068 interactions. AC is the number of all clusters predicted by the algorithm; EC is the number of effective clusters (with a least one matching complex above overlap ratio 0.4) found by the algorithm; MC is the number of matched complexes in the benchmark set. The sizes of complexcat benchmark and Gavin benchmark are 217 and 204, respectively. For HKC the optimized parameters are T_1, T_2, and T_3, respectively, and for MCODE the optimized parameters are NodeScoreCutoff, fluff (T for true, F for false), haircut (T for true, F for false), and other unspecified parameters adopt the default values.

Algorithm	Data set	Benchmark	*P*	*R*	*F*	AC	EC	MC	Optimized parameters
MCODE	MIPS	complexcat	0.455	0.194	0.271	66	30	42	0.05, F, F
HKC			**0.380**	**0.429**	**0.403**	**237**	**90**	**93**	**0.6, 10, 0.5**
MCODE	SGD-MC		0.213	0.221	0.217	197	42	48	0.05, F, T
HKC			**0.275**	**0.580**	**0.373**	**498**	**137**	**126**	**0.6, 10, 0.8**
MCODE	MIPS	Gavin	0.303	0.098	0.148	66	20	20	0.05, F, T
HKC			**0.237**	**0.235**	**0.236**	**245**	**58**	**48**	**0.6, 20, 0.5**
MCODE	SGD- MC		0.283	0.152	0.198	106	30	31	0, F, T
HKC			**0.271**	**0.402**	**0.324**	**487**	**132**	**82**	**0.5, 5, 0.5**

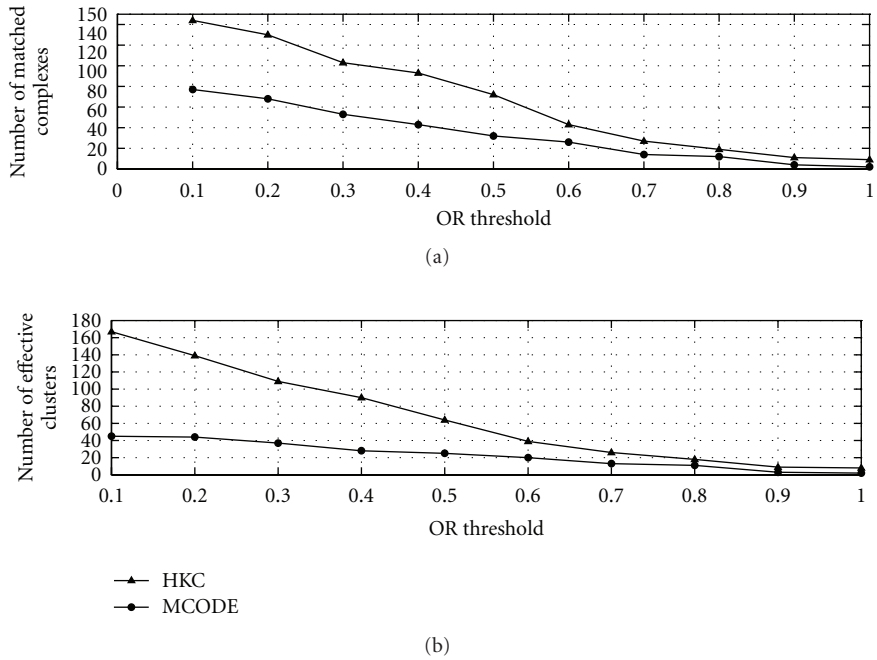

Figure 4: The number of effective clusters and the number of matched complexes by HKC and MCODE with respect to different OR thresholds. The result is corresponding to MIPS data set and evaluated on complexcat benchmark. Triangles mark the results of HKC and circles mark the results of MCODE.

HKC results for most parameter combinations are higher than that of MCODE, showing the overall improvement of the algorithm performance. From this figure, we can also see that the data points resulted by HKC are much more centralized than MCODE, indicating that our algorithm does not rely so severely on parameter selection.

3.3. Discussions. Among the 237 clusters found by HKC in MIPS data set, 8 clusters perfectly match with known protein complexes in complexcat benchmark. Figure 6(a) gives an example of one perfectly matched cluster: cluster 23 (consisting of 11 proteins and 54 interactions) perfectly matches with the TRAPP (transport protein particle) complex (catalog 260.60 in the complexcat benchmark), which plays an essential role in the vesicular transport from endoplasmic reticulum to Golgi.

Figure 6(b) shows an example of a containment match. Cluster 12 (consisting of 14 proteins and 92 interactions) is totally contained in a known complex of size 16, SAGA complex (catalog 510.190.10.20.10 in the complexcat benchmark), and their overlap ratio is 0.93. The two proteins YCL010c and YGL066w that are not recovered by cluster 12 have only one interaction with the cluster and do not exhibit good graph theoretic property. Actually, based on the available information currently, we cannot assert that YCL010c is contained in SAGA complex, and according to [22] it is only a probable subunit of SAGA complex.

Figure 6(c) gives an example of a well-matched cluster: cluster 103 matches with the complex of cytoplasmic translation initiation factor 3 (eIF3, catalog 500.10.40 in the complexcat benchmark). Each of them contains 7 proteins, their overlap is 6 and their overlap ratio is 0.857. As shown in the figure, protein YNL062c is contained in the benchmark complex, but is not included in cluster 103 predicted by HKC. Furthermore, it has only one interaction with cluster 103 and does not show an ideal topological property of belonging to a cluster. We searched it in Gene Ontology (GO) database [23] and found that according to the most updated GO annotation (release date 2011-05-14), YNL062c is not contained in the eIF3 complex (GO:0005852), but is a subunit of tRNA (1-methyladenosine) methyltransferase with Gcd14p required for the modification of the adenine at position 58 in tRNAs, especially tRNAi-Met. This indicates that the complexcat benchmark we use here may contain errors, because it was last modified in 2006 and many new protein complexes have been identified through experiments since then. In a way, it is possible to correct errors in the benchmark by carefully examining the difference between predicted clusters and their corresponding benchmark complexes with high overlap ratio.

Figure 6(d) shows a novel cluster (ranked 61) detected by HKC, and it involves 5 proteins and 10 interactions. Cluster 61 does not match with any known protein complex in the complexcat benchmark, but is highly homogenous in the cellular component ontology and biological process ontology. Search results on Gene Ontology database show that protein YGL153w, YLR191w and YNL214w form the docking complex that facilitates the import of peroxisomal matrix proteins, and YGL153w is a central component of

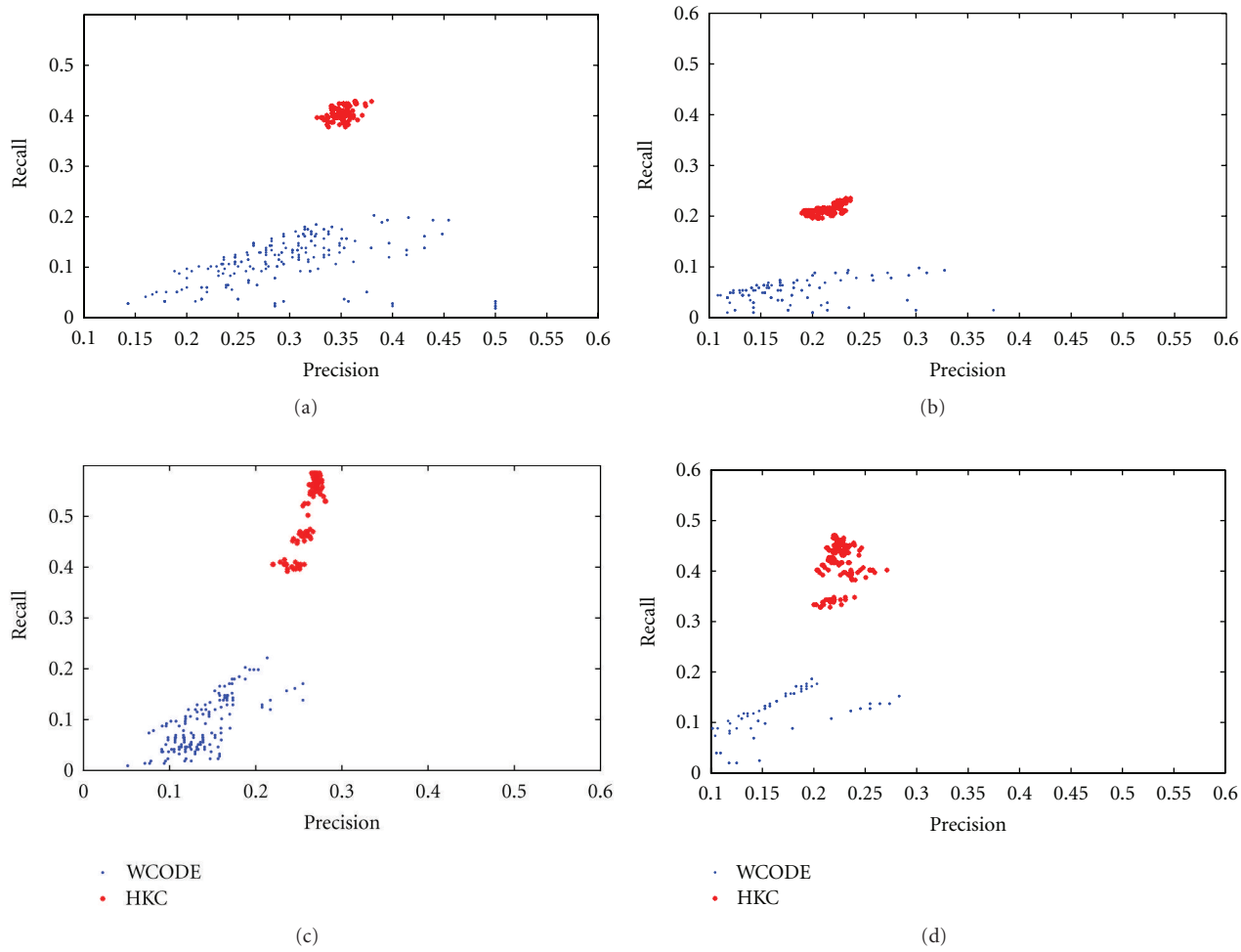

Figure 5: Precision versus Recall plots of all WCODE and HKC results with different parameters on various data sets. (a) Input data set: MIPS, benchmark: complexcat. (b) Input data set: MIPS, benchmark: Gavin. (c) Input data set: SGD-MC, benchmark: complexcat. (d) Input data set: SGD-MC, benchmark: Gavin. For all four cases, the data points resulted by HKC are located in the upper right portion of the plot, corresponding to high values of F-measure, while most of the data points resulted by MCODE are located in the lower left part of the plot. Furthermore, the data points resulted by HKC are much more centralized than MCODE.

the peroxisomal protein import machinery. The other two proteins in this cluster, YDR244w, and YDR142c, are the PTS1 signal recognition factor and the PTS2 signal recognition factor, respectively, and they also participate in the same biological process *protein docking during peroxisome matrix protein import* (GO: 0016560) as the docking complex.

The above illustrative examples show that HKC can not only effectively detect protein complexes in genome-scale PPI networks, but also through the comparison of predicted clusters and their matching benchmark complexes, it may help to correct the errors in the benchmark. Furthermore, HKC can discover novel protein complexes which can be used as candidates for experimental verification, and thus greatly helps to reduce the time consumption and cost of experiments. Among the 237 clusters resulted by our algorithm on MIPS data set, 147 clusters do not match with known complexes in complexcat benchmark, and we give all 49 clusters with score ≥ 4 and size ≥ 5 as novel predictions in Table 2, which would be a starting point for experimental validation in the future.

4. Conclusions and Future Work

A genome-scale PPI network is usually very large, consisting of thousands of proteins and tens of thousands of interactions, for example, the SGD-MC data set we use as the input PPI network in this paper contains 4,448 proteins and 29,068 interactions. It is a challenging task to extract protein complexes in such a large and complicated network. To solve this problem, many computational methods have been proposed, including the graph theoretic clustering algorithm. In this paper, we presented a new topology-based algorithm, HKC, which mainly used the concepts of highest k-core and cohesion to predict protein complexes by identifying overlapping clusters. The experiments on two data sets and two benchmarks showed that HKC can

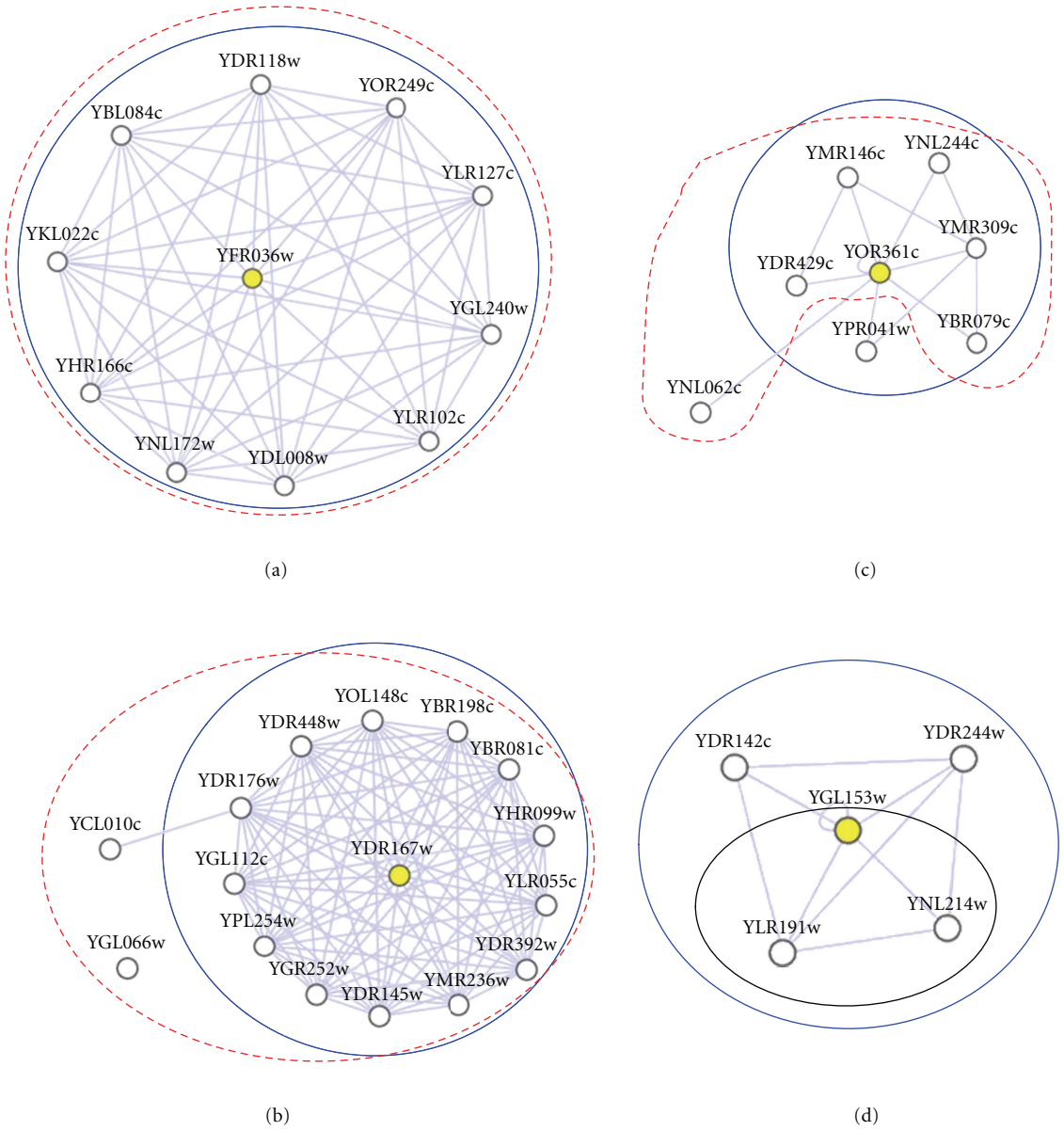

Figure 6: Examples of clusters predicted by HKC in MIPS data set. The nodes encircled by the red-dotted line are known complex in the complexcat benchmark, the nodes contained within the blue circle are clusters predicted by HKC, and the yellow node in each cluster denotes the seed of the cluster. (a) A cluster of size 11 perfectly matches with the TRAPP complex. (b) A cluster of size 14 shares 14 proteins with the SAGA complex (size 16), and their overlap ratio is 0.93. The two proteins YCL010c and YGL066w that are not contained in the predicted cluster are isolated nodes with only one edge connecting with the cluster. (c) An example of a well-matched cluster, involving 7 proteins, among which 6 is in common with the complex of cytoplasmic translation initiation factor 3 (eIF3). (d) A novel cluster detected by HKC, which does not match with any known protein complexes in the complexcat benchmark, and the proteins in the black circle form the docking complex that facilitates the import of peroxisomal matrix proteins according to GO annotation.

effectively extract protein complexes from genome-scale PPI networks and exhibited better performance compared with some other methods. Besides, HKC is general and can be used in any type of biological interaction networks.

There is huge amount of work to be done in PPI network analysis. As for protein complex prediction, there are also a lot of researches to be done in the future. One of the problems with the current PPI networks is that they are consisting of interactions that do not necessarily happen at the same time and space. Instead, the interactions in PPI networks may be unstable, transient or conditional, and may also happen in different subcellular locations. However, by definition, a protein complex is a group of proteins that interact with each other at the same time and place. As a result, to increase

Table 2: All novel predictions by HKC on MIPS data set. Each column gives the original ID of clusters (ID), cluster score (S), number of nodes (N), number of edges (E), and the protein names in each cluster.

ID	S	N	E	Protein names
1	21.5	38	406	YLR200w,YNL153c,YDR318w,YOR349w,YHR191c,YEL003w,YJL013c,YLR085c,YPL269w,YHR129c,YOR026w,YMR055c,YPL155c,YCL029c,YMR048w, YML094w,YJL030w,YGR188c,YCL016c,YPR135w,YML124c,YPR141c,YER016w,YOR058c,YER007w,YDR150w,YGR078c,YJR053w,YEL061c, YMR294w,YMR299c,YPL008w,YMR078c,YGL086w,YPL174c,YFL037w,YOL012c,YPL241c
2	20.1	33	328	YPL008w,YMR078c,YGR188c,YGL086w,YCL016c,YPL241c,YMR294w,YER007w,YOR058c,YER016w,YPR141c,YPR135w,YJL030w,YPL155c,YOR026w, YHR129c,YPL269w,YJL013c,YDR254w,YDR318w,YFL037w,YPL174c,YEL061c,YGR078c,YLR200w,YML124c,YML094w,YCL029c,YEL003w,YOR349w, YNL153c,YMR138w,YOR265w
3	19.7	27	260	YGL086w,YMR078c,YCL016c,Y*GR188*c,YMR048w,YOR026w,YJL013c,YHR191c,YOR349w,YDR318w,YPL008w,YEL061c,YGR078c,YLR200w,YER016w, YPR141c,YPR135w,YJL030w,YML094w,YEL003w,YNL153c,YDL003w,YDR254w,YJR135c,YPR046w,YPL018w,YLR381w
4	18. 9	28	260	YGR188c,YLR381w,YPL018w,YDR254w,YDR318w,YOL012c,YGL086w,YPL008w,YGR078c,YDL003w,YLR200w,YML094w,YMR048w,YLR085c, YEL003w,YOR349w,YNL153c,YNL298w,YMR078c,YEL061c,YER016w,YPR141c,YPR135w,YCL016c,YJL030w,YHR191c, YOR195w,YGL216w
5	15.2	24	179	YNL271c,YNL322c,YBL061c,YBR023c,YGR229c,YBL007c,YER155c,YLR330w,YHR030c,YNL298w,YDL029w,YJR075w,YGR078c,YLR200w,YLR337c, YML094w,YDR388w,YCR009c,YBR234c,YEL003w,YJL095w,YNL153c,YJL020c,YMR109w
6	15.2	25	185	YER155c,YEL003w,YGR078c,YML094w,YNL271c,YGR229c,YNL298w,YLR337c,YNL153c,YLR200w,YHR030c,YBL007c,YCR009c,YJL095w,YDR245w, YBL061c,YHR142w,YLR330w, YDL029w,YDR388w,YBR023c,YBR234c,YJR075w,YNL322c,YDR129c
7	14.8	27	196	YBL007c,YEL003w,YGR078c,YLR200w,YML094w,YNL153c,YNL298w,YJL095w,YHR111w,YGR229c,YBR234c,YCR009c,YER111c,YBR023c,YDR388w, YHR030c,YDL029w,YLR330w,YHR142w,YDR424c,YNL271c,YFR019w,YBL061c,YPL031c,YBR200w,YER155c,YOR326w
8	14.6	25	178	YLR337c,YNL322c,YNL271c,YBL007c,YLR200w,YBL061c,YHR142w,YLR330w,YML094w,YBR023c,YNL233w,YCR009c,YJR075w,YNL153c,YNL298w, YDL029w,YHR030c,YJL183w,YDR388w,YBR234c,YGR229c,YLR342w,YJL095w,YJL099w,YJR118c
9	14.3	21	146	YNL250w,YER016w,YPR141c,YHR191c,YGL163c,YMR078c,YMR190c,YKL113c,YOR144c,YCL061c,YER173w,YPR135w,YCL016c,YMR048w,YLR234w, YJL092w,YLR103c,YML032c,YPL194w,YNL273w,YJR043c
10	14.3	21	147	YML032c,YER016w,YJR043c,YMR078c,YNL273w,YLR103c,YPR135w,YCL016c,YMR048w,YHR191c,YGL163c,YMR190c,YDR004w,YOR144c,YER095w, YCL061c,YDR076w,YER173w,YNL250w,YJL092w,YKL113c
11	14.2	21	146	YNL298w,YHR030c,YJL095w,YMR109w,YBR023c,YDL029w,YBR234c,YGR078c,YLR200w,YBL061c,YML094w,YDR388w,YJL020c,YCR009c,YGR229c, YEL003w,YNL153c,YLR337c,YEL031w,YBL007c,YJR075w
13	13.9	23	155	YLR200w,YNL271c,YNL153c,YBR234c,YNL322c,YHR142w,YER111c,YBL007c,YCR009c,YGR229c,YJL095w,YNL298w,YLR330w,YBR023c,YJR075w, YBL061c,YDL029w,YHR030c,YJR118c,YDR388w,YBL047c,YNL233w,YLR342w
14	13.7	23	153	YHR191c,YML032c,YHR031c,YMR078c,YCL061c,YCL016c,YMR048w,YNL273w,YNL250w,YPR135w,YMR190c,YKL113c,YLR234w,YJR043c,YOR144c, YLR103c,YJL092w,YOL006c,YPL024w, YBR098w,YDR386w,YLR235c,YDR363w
15	13.6	23	154	YER173w,YKL113c,YML032c,YNL273w,YOR144c,YMR048w,YJR043c,YMR078c,YCL061c,YLR103c,YPR135w,YCL016c,YHR191c,YMR190c,YPL024w, YDR052c,YNL250w,YHR031c,YLR235c,YLR234w,YJL092w,YDL017w,YHR154w
16	13.3	21	136	YPL194w,YML032c,YNL250w,YJL092w,YLR103c,YMR078c,YCL016c,YHR191c,YGL163c,YJR043c,YMR190c,YKL113c,YNL273w,YCL061c,YER016w, YER173w,YPR135w,YMR048w,YOR144c,YDL013w,YER116c
17	12.9	22	138	YER016w,YGL163c,YHR191c,YNL273w,YML032c,YLR103c,YMR078c,YKL113c,YOR144c,YCL061c,YPR135w,YNL250w,YCL016c,YHR031c,YMR048w, YLR234w,YJL092w,YJR043c,YMR190c,YDR363w,YBR228w,YLR135w
20	10.8	13	66	YER146w,YDR378c,YNL147w,YLR438c-a,YER112w,YJL124c,YCR077c,YBL026w,YJR022w,YOL149w,YGL173c,YNL118c,YEL015w
21	10.8	11	54	YBL026w,YER146w,YDR378c,YJR022w,YNL147w,YLR438c-a,YER112w,YOL149w,YJL124c,YCR077c,YGL173c
24	10	17	82	YBL061c,YDR388w,YBR023c,YJL095w,YNL298w,YNL271c,YLR330w,YHR030c,YER111c,YGR229c,YER155c,YPL031c,YMR307w,YOR008c,YLR342w, YLR371w,YLR332w
25	9.9	17	81	YJL095w,YDL029w,YCR009c,YNL298w,YBL061c,YDR388w,YER111c,YGR229c,YNL322c,YLR330w,YHR030c,YBR023c,YMR307w,YGL027c,YML115c, YGL200c,YPR159w
26	9.5	17	78	YNL298w,YDR388w,YER111c,YNL271c,YLR330w,YHR030c,YBL061c,YBR023c,YNL322c,YMR307w,YLR039c,YLR262c,YGR229c,YLR342w,YJR073c, YJL183w,YKL190w
27	8.6	14	56	YLR039c,YPL051w,YLR262c,YJL154c,YDR126w,YGL005c,YOR132w,YOR069w,YNL051w,YHL031c,YNL041c,YOR070c,YML071c,YBR164c

Table 2: Continued.

ID	*S*	*N*	*E*	Protein names
28	8.5	14	55	YOL012c,YLR039c,YLR262c,YLR085c,YLR418c,YAL011w,YAL013w,YMR263w,YJL168c,YML041c,YGL244w,YPL181w,YDR334w,YOR123c
29	7.7	13	46	YNL051w,YHL031c,YPL051w,YOR070c,YNL041c,YLR262c,YML071c,YDR126w,YKL190w,YLR039c,YBR164c,YKR001c,YMR004w
30	7.6	11	39	YBR023c,YKL190w,YNL322c,YHR030c,YGR229c,YMR307w,YJR073c,YLR262c,YDR162c,YLR039c,YDL006w
31	7.5	13	45	YNL051w,YML071c,YPL051w,YOR070c,YNL041c,YLR262c,YOL018c,YDR126w,YHL031c,YBR164c,YLR039c,YNL238w,YLL040c
39	6	17	51	YBR200w,YNL298w,YOR127w,YHR061c,YHL007c,YER149c,YGR152c,YLL021w,YPL115c,YLR319c,YDR309c,YBL085w,YAL041w,YER114c,YOR188w, YMR273c,YLR229c
40	6	8	25	YCR088w,YOR284w,YGR268c,YDR388w,YBL007c,YNL094w,YNL243w,YHR016c
42	6	7	18	YER155c,YAL040c,YDL047w,YKR028w,YJL098w,YFR040w,YKR072c
43	5.8	10	27	YMR263w,YOR123c,YLR085c,YGL244w,YML041c,YJL168c,YAL013w,YLR418c,YBL008w,YOR038c
48	5.5	8	26	YIL144w,YDR201w,YFL008w,YOL069w,YFR031c,YEL043w,YDL074c,YJL074c
50	5.3	17	55	YLR347c,YNL189w,YML064c,YLR082c,YLR245c,YDR321w,YPL070w,YBR176w,YNL044w,YNL331c,YPL124w,YER023w,YLR423c,YJR056c,YEL066w, YJL218w,YNR012w
51	5.3	9	26	YOR284w,YJL199c,YNL189w,YML064c,YBR176w,YLR245c,YPL070w,YLR291c,YHL018w
55	4.9	8	18	YHR066w,YPL093w,YER006w,YPL211w,YMR049c,YMR290c,YNL002c,YGR103w
56	4.9	8	17	YGL244w,YLR039c,YLR262c,YOR216c,YLR085c,YLR418c,YIR005w,YGL174w
59	4.5	17	49	YML064c,YNL189w,YLR347c,YGR010w,YKL067w,YFR047c,YGL175c,YOR020c,YLR328w,YLR335w,YGL040c,YGL037c,YNL333w,YFL059w,YPL111w, YGR267c,YLR377c
60	4.5	5	11	YFL059w,YNL333w,YMR322c,YMR095c,YMR096w
61	4.5	5	10	YNL214w,YDR244w,YGL153w,YDR142c,YLR191w
65	4.4	6	12	YLR310c,YLL016w,YAL024c,YNL098c,YAR019c,YOR101w
70	4.3	7	16	YML064c,YPL124w,YLR423c,YPL070w,YDR148c,YER086w,YDL239c
71	4.3	7	14	YDL226c,YGR172c,YLR324w,YGL198w,YDR425w,YGL161c,YPL095c
73	4.3	7	14	YLR347c,YMR047c,YKL068w,YBR216c,YML007w,YER107c,YDL207w
74	4.3	7	14	YLR039c,YPL234c,YKL190w,YLR262c,YER081w,YHR060w,YGR020c
77	4	5	11	YER093c,YJL058c,YJR066w,YKL203c,YNL006w
78	4	5	12	YKL203c,YJR066w,YNL006w,YHR186c,YPL180w
82	4	5	8	YLR039c,YLR262c,YCR020c-a,YPR051w,YEL053c
90	4	5	8	YAR019c,YGR092w,YDL047w,YPR111w,YIL106w
93	4	5	8	YLR039c,YOR070c,YBR036c,YMR272c,YPL057c
94	4	5	8	YMR197c,YHL031c,YLR026c,YKL006c-a,YKL196c

the precision and recall of complex prediction algorithm, further information about the time, space, or conditions of interactions should be taken into consideration.

Acknowledgments

This work was supported in part by the Grants from the National Natural Science Foundation of China (no. 60835005) and the scientific research project of National University of Defense Technology (jc09-03-04).

References

[1] Y. Ho, A. Gruhler, A. Heilbut et al., "Systematic identification of protein complexes in *Saccharomyces cerevisiae* by mass spectrometry," *Nature*, vol. 415, no. 6868, pp. 180–183, 2002.

[2] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, "A comprehensive two-hybrid analysis to explore the yeast protein interactome," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 8, pp. 4569–4574, 2001.

[3] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 21, pp. 12123–12128, 2003.

[4] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray, "From molecular to modular cell biology," *Nature*, vol. 402, no. 6761, pp. C47–C52, 1999.

[5] 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*, vol. 7, article 207, 2006.

[6] G. D. Bader and C. W. V. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, article 2, 2003.

[7] A. D. King, N. Pržulj, and I. Jurisica, "Protein complex prediction via cost-based clustering," *Bioinformatics*, vol. 20, no. 17, pp. 3013–3020, 2004.

[8] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein-protein interaction network," *Bioinformatics*, vol. 22, no. 18, pp. 2283–2290, 2006.

[9] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[10] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[11] H. N. Chua, K. Ning, W. K. Sung, H. W. Leong, and L. Wong, "Using indirect protein-protein interactions for protein complex predication," *Life Sciences Society: Computational Systems Bioinformatics Conference*, vol. 6, pp. 97–109, 2007.

[12] G. Cui, Y. Chen, D. S. Huang, and K. Han, "An algorithm for finding functional modules and protein complexes in protein-protein interaction networks," *Journal of Biomedicine and Biotechnology*, vol. 2008, Article ID 860270, 10 pages, 2008.

[13] H. W. Mewes, S. Dietmann, D. Frishman et al., "MIPS: analysis and annotation of genome information in 2007," *Nucleic Acids Research*, vol. 36, no. 1, pp. D196–D201, 2008.

[14] U. Güldener, M. Münsterkötter, G. Kastenmüller et al., "CYGD: the comprehensive yeast genome database," *Nucleic Acids Research*, vol. 33, pp. D364–D368, 2005.

[15] J. M. Cherry, C. Adler, C. Ball et al., "SGD: saccharomyces genome database," *Nucleic Acids Research*, vol. 26, no. 1, pp. 73–79, 1998.

[16] S. Brohée and J. van Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC Bioinformatics*, vol. 7, article 488, 2006.

[17] A. C. Gavin, M. Bösche, R. Krause et al., "Functional organization of the yeast proteome by systematic analysis of protein complexes," *Nature*, vol. 415, no. 6868, pp. 141–147, 2002.

[18] N. J. Krogan, W. T. Peng, G. Cagney et al., "High-definition macromolecular composition of yeast RNA-processing complexes," *Molecular Cell*, vol. 13, no. 2, pp. 225–239, 2004.

[19] A. C. Gavin, P. Aloy, P. Grandi et al., "Proteome survey reveals modularity of the yeast cell machinery," *Nature*, vol. 440, no. 7084, pp. 631–636, 2006.

[20] J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari, "Modular decomposition of protein-protein interaction networks," *Genome Biology*, vol. 5, no. 8, p. R57, 2004.

[21] S. Killcoyne, G. W. Carter, J. Smith, and J. Boyle, "Cytoscape: a community-based framework for network modeling," *Methods in Molecular Biology*, vol. 563, pp. 219–239, 2009.

[22] S. L. Sanders, J. Jennings, A. Canutescu, A. J. Link, and P. A. Weil, "Proteomics of the eukaryotic transcription machinery: identification of proteins associated with components of yeast TFIID by multidimensional mass spectrometry," *Molecular and Cellular Biology*, vol. 22, no. 13, pp. 4723–4738, 2002.

[23] M. Ashburner, C. A. Ball, J. A. Blake et al., "Gene ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.