BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–5

PROMO : A Method for identifying modules in protein interaction networks

Omer Tamuz[*], *Yaron Singer*[*], *Roded Sharan*

School of Computer Science, Tel Aviv University, Tel Aviv, Israel

ABSTRACT

Motivation: A major goal of systems biology is the dissection of protein machineries within the cell. The recent availability of genome-scale protein interaction networks provides a key resource for addressing this challenge. Current approaches to the problem are based on devising a scoring scheme for putative protein modules and applying a heuristic search for high-scoring modules.

Results: Here we develop a branch and bound approach to perform an exhaustive scan of the search space.We show that such a search is possible and enables detecting modules that are missed by previous approaches. The modules we identify are shown to be significantly coherent in their functional annotations and expression patterns. Our algorithm, PROMO, is shown to outperform the state-of-the-art MCODE and to provide results that are more in line with current biological knowledge.

Contact: omert@wise.tau.ac.il

1 INTRODUCTION

A major goal of systems biology is understanding how intricate networks of molecular interactions give rise to biological form and function. Recent technological advances enable a global mapping of protein-protein interactions (PPIs) within the cell, and provide an opportunity for addressing this challenge. A key step in interpreting such data is the elucidation of protein machinery, or *modules*, within the cell.

Many authors have studied the problem of identifying modules within a PPI network. The Molecular Complex Detection algorithm (MCODE) (Bader and Hogue, 2003) is the most commonly used method for module detection (see, e.g., LaCount *et al.* (2005); Rual *et al.* (2005)). It weighs vertices based on the densities of their neighborhoods and launches greedy local searches for dense network regions. A similar algorithm by Altaf-Ul-Amin *et al.* (2006) grows modules in a greedy fashion while ensuring that vertices that are added to the module are densely connected to it. The NetworkBlast algorithm of Sharan *et al.* (2005a) is based on a maximum likelihood scoring scheme. Each candidate set of proteins is assigned a likelihood ratio score that measures its fit to a protein complex model vs. the chance that its connections arise at random. A greedy algorithm is used for identifying high-scoring modules. Other proposed algorithms apply various clustering techniques for module detection (King *et al.*, 2004; Maciag *et al.*, 2006; Spirin and Mirny, 2003).

Our goal here is to provide a general strategy for identifying high scoring modules in a PPI network, given a linear scoring scheme of interest (i.e., a scheme that decomposes over the edges and non-edges of the network). The algorithm we propose, PROtein Module Optimizer (PROMO), is based on an efficient branch and bound search that scans through all possible protein modules that score above a certain threshold. While the problem we aim at solving is NP-complete, we show that on current PPI data it can be solved in minutes. We compare the performance of our exhaustive approach to the state-of-the-art MCODE algorithm, and show its superiority.

2 METHODS

2.1 Overview

We have developed an efficient algorithm for identifying optimal-scoring modules in a given network under a linear scoring scheme. The algorithm receives as input a PPI network, whose nodes represent proteins and whose edges represent protein-protein interactions and are assigned with confidence scores. The module identification problem is recast to that of identifying a subnetwork M within the weighted network such that the sum of weights of node pairs within M is maximum. A branch-and-bound approach is applied to identify the optimum solution.

2.2 Algorithm

Let $G = (V, E, w)$ be a PPI network, with V representing the set of proteins, E representing the set of PPIs, and $w : V \times V \rightarrow \mathbb{R}$ a weight function. The module identification problem can be formulated as that of finding a vertex subset $V' \subseteq V$ such that its *weight*, $W(V') = \sum_{u,v \in V'} w(u, v)$, is maximum. While this problem is known to be computationally hard (generalizing the problem of finding a maximum clique in a graph (Garey and Johnson, 1979)), we show that it can be efficiently tackled using a branch-and-bound approach.

[*]These authors contributed equally

Branch-and-bound is a well known exhaustive optimization technique (Lawler and Wood, 1966). It includes a *branching* mechanism, which divides the parameter space into subspaces, and a *bounding* mechanism, which calculates an upper bound to the score of the possible solutions in a subspace and compares it to a lower bound–the score of the best solution found so far.

In our case, the search space is the collection of all subsets of V. For $v \in V$, branching is naturally achieved by partitioning the set of all possible solutions to those that include or exclude v. A subset A of the parameter space is therefore a division of the set of vertices V into three subsets: V_{in}, V_{out} and V_{maybe}, where all the solutions in A include the vertices in V_{in}, do not include the vertices in V_{out} and perhaps include the vertices in $V_{maybe} = V \setminus (V_{in} \cup V_{out})$. Bounding can be naively performed by considering only the positive edges which might be in a solution, but can be significantly improved, as we show below.

A schematic description of the algorithm is appended below.

Key to the success of a branch-and-bound methodology are: (i) obtaining a good initial lower bound; (ii) tight estimation of the upper bound; and (iii) efficient branching strategy. To obtain an initial solution we use a greedy approach.

The upper bound can be made tighter through the following observation: given a vertex $v \in V_{maybe}$ the edges between v and vertices in V_{in} are "bound together" in the sense that either all of them are in the optimal solution, or all are not. Therefore, they can be treated as a single edge, weighted as the sum of the weights. Also, the total contribution of a vertex cannot be negative. Thus, for a vertex $v \in V_{maybe}$, a naive branch-and-bound approach quantifies its contribution to the upper bound by:

$$\sum_{u \in V_{in}} \max\{0, w(u,v)\} + \frac{1}{2} \sum_{u \in V_{maybe}} \max\{0, w(u,v)\} \quad (1)$$

In PROMO we use:

$$\max\left\{0, \sum_{u \in V_{in}} w(v,u) + \frac{1}{2} \sum_{u \in V_{maybe}} \max(0, w(v,u))\right\} \quad (2)$$

where the $1/2$ factor is due to the fact that each edge is accounted for by both its incident vertices. Since for all $v \in V_{maybe}, 1 \geq 2$ we obtain a tighter bound.

Finally, the branching strategy relies on the following insight: In calculating the upper bound, positive weights between vertices in V_{maybe} are the only ones considered, thus leading to overestimation of the contribution of vertices to the actual best score in a subspace A. Clearly, we wish to ensure that the algorithm processes the vertices in an order that will yield the fastest decrease in the upper bound, and yet the algorithm will not be trapped in subspaces for which the best score is lower than the optimum. This motivated us to sort the vertices by the difference between their contribution to the upper bound and to the best score. More precisely, since we cannot readily compute the best score, we resort to computing the expected difference in contributions. Clearly, such differences arise due to our uncertainty regarding which vertices in V_{maybe} will be eventually included in the optimum solution.

The contribution of each vertex $v \in V_{maybe}$ to the upper bound is given by Eq. 2. To compute the expected contribution to the best score we make a simplifying assumption: the probability that a vertex $v \in V_{maybe}$ is included in the best solution equals to the ratio of the sum of weights of positive edges incident to it and the overall sum of weights, in absolute values, of edges incident to it. Formally, let $w(v,v) = \sum_{u \in V_{in}} w(u,v)$ then:

$$p_v \equiv Pr(v \in V_{opt}) = \frac{\sum_{u \in V_{maybe}: w(u,v) > 0} w(u,v)}{\sum_{u \in V_{maybe}} |w(u,v)|}$$

It follows that the expected contribution of v to the best score is: $p_v \cdot \sum_{u \in V} p_u w(u,v)$. To save on running time, we use a simpler formula for sorting the vertices which is obtained by setting $p_v = 1$ for all $v \in V_{maybe}$.

Algorithm 1 The PROMO algorithm for module identification.

```
PROMO(V,L,U_best)
  return Recursion(φ,V,U_best)

Recursion(V_in, V_maybe,U_best)
  if V_maybe = φ
    if W(V_in) > W(U_best)
      return V_in
  else if UpperBound(V_in, V_maybe) > W(U_best)
    choose v ∈ V_maybe
    V_maybe ← V_maybe \ {v}
    U_best ← Recursion(V_in, V_maybe, U_best)
    U_best ← Recursion(V_in ∪ {v}, V_maybe, U_best)
  return U_best
```

2.3 Module scoring

We use the maximum likelihood scoring scheme presented in Sharan *et al.* (2005b). Briefly, a module is assigned a likelihood ratio score, which measures its fit to a protein complex model vs. the chance that the module's connections arise at random. The protein complex model assumes that every two proteins in a complex should interact, independently of all other pairs, with high probability β. The random model assumes that the PPI graph was chosen uniformly at random from the collection of all graphs with the same vertex degrees as the ones observed. This induces a probability of occurrence p_{uv} for each edge (u,v) of the graph. Under this model each

vertex pair receives a log likelihood ratio score which measures its fit to a protein module model vs. a random background. Given a module U, the likelihood ratio score factors over the vertex pairs in the module:

$$\mathcal{L}(V) = \sum_{(u,v)\in E} \log \frac{\beta Pr(O_{uv}|T_{uv}) + (1-\beta)Pr(O_{uv}|F_{uv})}{p_{uv}Pr(O_{uv}|T_{uv}) + (1-p_{uv})Pr(O_{uv}|F_{uv})}, \quad (3)$$

where O_{uv} denotes the set of experimental observations on the interaction between u and v, T_{uv} denotes the event that u and v truly interact, and F_{uv} denotes the event the u and v do not interact. The computation of $Pr(O_{uv}|T_{uv})$ and $Pr(O_{uv}|F_{uv})$ is based on the reliability assigned to the interaction between u and v (see Sharan *et al.* (2005b) for further details).

2.4 Module significance assessment and filtering

To assess the significance of the modules output by the algorithm, we compare their scores to those obtained on random networks. Specifically, we construct 100 random graphs with the same vertex degrees as in the original network, and apply our module discovery algorithm to each of them. For each root vertex $v \in V$ we record the best module obtained for it in each of the random runs. For each possible module size s, we collect all random modules of size s, and use their score distribution to determine a p-value for each real module of that size. We retain only modules with $p < 0.01$.

To avoid highly overlapping modules, we used an iterative procedure to filter the significant modules identified. Each iteration, the highest scoring module is output and all other modules that overlap it by more than 50% are removed. The amount of overlap is measured w.r.t. the smaller of the two modules compared.

2.5 Algorithmic speedups

We combined several speedups into the algorithm that are motivated by our assumptions regarding protein modules. First, we assume that a module should induce a connected subnetwork. Second, we assume that within a module every two proteins are at most l connections apart ($l = 2$ in our implementation). This restriction can be imposed by assigning $-\infty$ weights to vertex pairs that have distance$> l$. This assumption also allows us to focus the search on the l-neighborhood of each protein.

The method we have described guarantees the discovery of the highest-scoring module in the network. However, if two significant modules overlap, even by only a small amount, only one of them will be discovered. To circumvent this problem and allow the identification of a large number of significant modules we use the following heuristic: For each root vertex v, we apply the branch-and-bound algorithm in an iterative manner to its l-neighborhood. Each iteration, we identify the highest scoring module S in the current graph, remove the vertices in $S \setminus \{v\}$ and continue on the reduced graph.

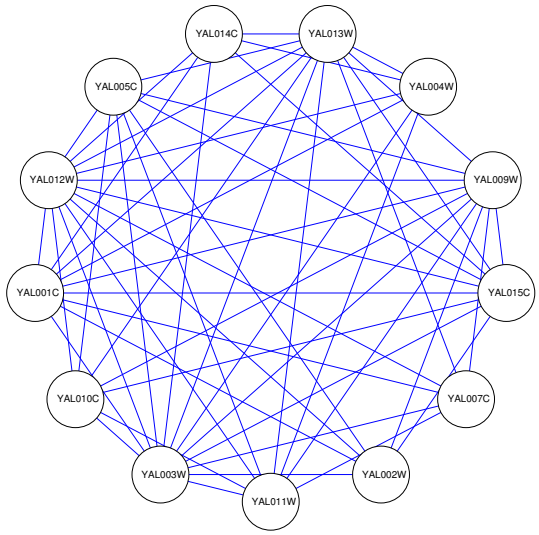

Fig. 1. Sample PROMO complex.

3 RESULTS

We applied PROMO to analyze the PPI network of yeast, which is one of the largest and most established networks in public databases (dip, 2006). In Fig. 2 we compare running times of PROMO vs. estimated running times (computed by extrapolation) of a full exhaustive search and a "naive" branch and bound algorithm employing the naive upper bound and no vertex sorting. It seems that while the naive branch and bound is exponential in the size of the searched graph, PROMO is exponential in the size of the optimal module (Fig 3).

Our application to the network yielded 65 significant, non-redundant modules. We considered only modules of size 8 or larger, but the results reported below remained relatively the same for higher thresholds. We compared our performance to that of the state-of-the-art MCODE approach (Bader and Hogue, 2003). MCODE was executed via its cytoscape plugin (cyt, 2006) with default parameters, except for the node score cutoff. For the latter we tried both the default value (0.2) and a smaller value (0.05) that avoids huge clusters. Fig. 4 shows the distributions of module sizes for PROMO and MCODE .

To evaluate the quality of the solutions we used information on protein cellular processes from the gene ontology (GO) (Consortium, 2000), protein complex association from the MIPS database (mip, 2006) and gene expression measurements. In each test, we calculated a score and compared it with those obtained for random sets of proteins of the same size as the module, and derived an empirical p-value for the

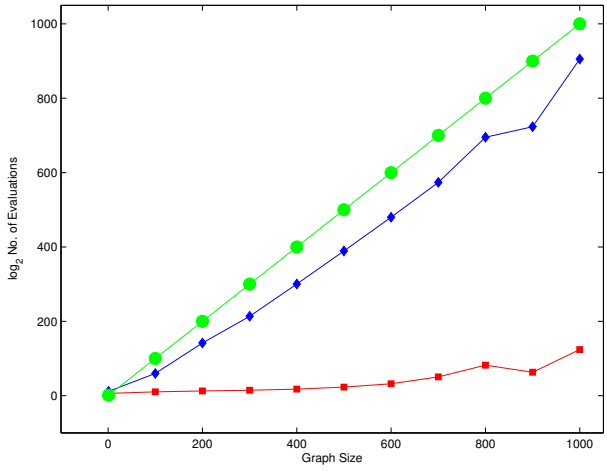

Fig. 2. Running times of naive search (circles), naive branch and bound (diamonds) and PROMO (squares) vs. the size of the searched graph.

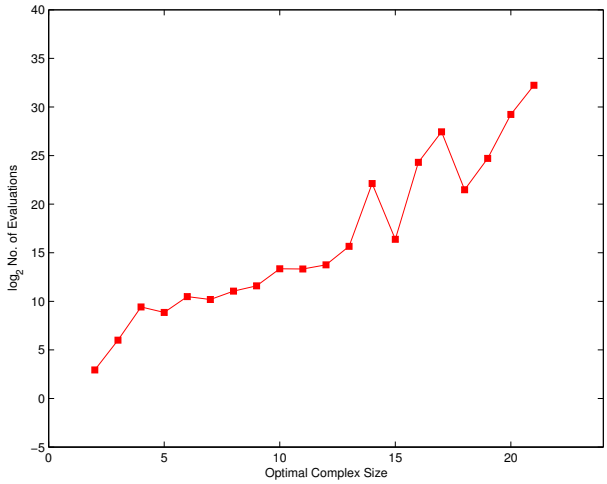

Fig. 3. Running time of PROMO vs. the size of the optimal module.

module. These p-values were further FDR corrected for multiple testing. Finally, we report the fraction of significant modules ($p \leq 0.01$).

To compute the functional enrichment of a module we scored it against each of the GO terms using a hypergeometric score. The lowest p-value obtained was used in the subsequent computations. The expression coherency of a module was measured as the mean pairwise Pearson correlation between the expression vectors of the module's genes.

To assess the quality of the modules w.r.t. known complexes in MIPS, we first computed an enrichment score for each module, in an analogous manner to the way functional enrichment was computed. We defined the specificity of the solution as the fraction of MIPS enriched modules. The sensitivity

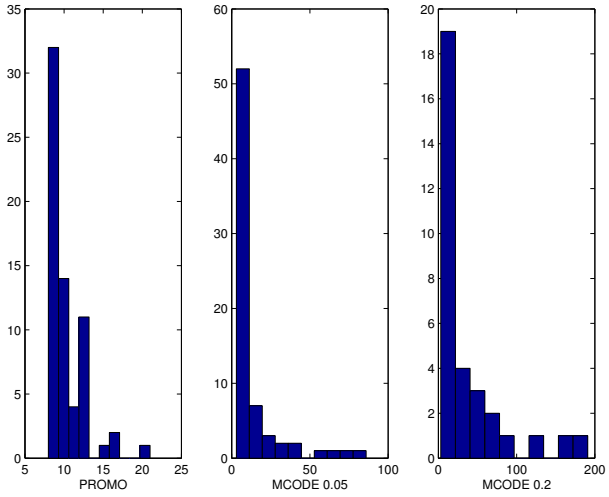

Fig. 4. Module sizes for PROMO and MCODE.

was measured as the fraction of MIPS categories for which a module was enriched with that category.

The performances of the two algorithms w.r.t. these measures are summarized in Table 1. Evidently, PROMO produces results that are more aligned with known biological annotations.

4 CONCLUSIONS

PROMO is an exhaustive, yet practical approach for exploring the landscape of protein modules in a network. Unlike previous approaches, it guarantees optimal solutions, and succeeds in uncovering modules that are missed by current approaches. The modules identified by the algorithm are shown to be highly functional and expression coherent, and to match known complexes.

REFERENCES

(2006). Cytoscape. http://www.cytoscape.org/.

(2006). The DIP database. http://dip.doe-mbi.ucla.edu/.

(2006). The MIPS database. http://mips.gsf.de/.

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

Bader, G. and Hogue, C. (2003). An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, **4**.

Consortium, T. G. O. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, **25**, 25–9.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco.

King, A., Przulj, N., and Jurisica, I. (2004). Protein complex prediction via cost-based clustering. *Bioinformatics*, **20**, 3013–3020.

Method	Functional enrichment	Expression coherency	MIPS specificity	MIPS sensitivity
PROMO	**0.97**	**0.64**	0.16	0.82
MCODE 0.05	0.87	0.31	**0.17**	0.71
MCODE 0.2 (default)	0.94	0.41	0.11	**0.89**

Table 1. A comparison of module identification algorithms.

LaCount, D. *et al.* (2005). A protein interaction network of the malaria parasite plasmodium falciparum. *Nature*, **438**, 103–7.

Lawler, E. and Wood, D. (1966). Branch-and-bound methods: a survey. *Operations Research*, pages 699–719.

Maciag, K., Altschuler, S., Slack, M., Krogan, N., Emili, A., Greenblatt, J., Maniatis, T., and Wu, L. (2006). Systems-level analyses identify extensive coupling among gene expression machines. *Molecular Systems Biology*, **2**.

Rual, J.-F. *et al.* (2005). Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, **437**, 1173–8.

Sharan, R., Suthram, S., Kelley, R., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R., and Ideker, T. (2005a). Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci. USA*, **102**, 1974–1979.

Sharan, R., Suthram, S., Kelley, R., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R., and Ideker, T. (2005b). Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci. USA*, **102**, 1974–1979.

Spirin, V. and Mirny, L. (2003). Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA*, **100**, 12123–12128.