BMC Bioinformatics

PROCEEDINGS Open Access

A novel subgradient-based optimization algorithm for blockmodel functional module identification

Yijie Wang*, Xiaoning Qian*

From The Eleventh Asia Pacific Bioinformatics Conference (APBC 2013)
Vancouver, Canada. 21-24 January 2013

Abstract

Functional module identification in biological networks may provide new insights into the complex interactions among biomolecules for a better understanding of cellular functional organization. Most of existing functional module identification methods are based on the optimization of network modularity and cluster networks into groups of nodes within which there are a higher-than-expectation number of edges. However, module identification simply based on this topological criterion may not discover certain kinds of biologically meaningful modules within which nodes are sparsely connected but have similar interaction patterns with the rest of the network. In order to unearth more biologically meaningful functional modules, we propose a novel efficient convex programming algorithm based on the subgradient method with heuristic path generation to solve the problem in a recently proposed framework of blockmodel module identification. We have implemented our algorithm for large-scale protein-protein interaction (PPI) networks, including *Saccharomyces cerevisia* and *Homo sapien* PPI networks collected from the Database of Interaction Proteins (DIP) and Human Protein Reference Database (HPRD). Our experimental results have shown that our algorithm achieves comparable network clustering performance in comparison to the more time-consuming simulated annealing (SA) optimization. Furthermore, preliminary results for identifying fine-grained functional modules in both biological networks and the comparison with the commonly adopted Markov Clustering (MCL) algorithm have demonstrated the potential of our algorithm to discover new types of modules, within which proteins are sparsely connected but with significantly enriched biological functionalities.

Introduction

Biomolecules interact with each other in a complex modular manner to maintain normal cellular functionalities [1,2]. Identifying recurrent functional modules may help better understand the functional organization of cells [3]. Many existing network clustering algorithms for functional module identification focus on identifying "modules" within which nodes are densely connected [4-6]. However, identifying modules using these existing computational approaches may have artificially introduced biases from their module definitions and corresponding optimization methods [2]. Topologically defined modules may simply originate from the evolution process but do not necessarily correspond to functional units in cells [7]. In addition to densely self-connected modules, there are other topological structures in biological networks which capture important functional relationships among biomolecules. For example, transmembrane proteins, including receptors in many signal transduction pathways, have a special structure in which they rarely interact with each other but have a similar interaction patterns with the rest of the network [2]. To identify functional modules with richer topological structures, blockmodel network clustering recently has been proposed for functional module identification in biological networks [2,4,8,9].

Blockmodel module identification problem has been investigated for years [2,7,10] and has recently been

* Correspondence: yijie@mail.usf.edu; xqian@cse.usf.edu
Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, USA

used for blockmodeling functional modules within biological networks [2]. However, the resulting optimization problem is NP hard with highly nonlinear and non-convex properties with many local optima in its objective function. This makes it computationally prohibitive to obtain the optimal modules, especially for large-scale biological networks. Simulated Annealing (SA) algorithm has been used to solve the optimization problem [2]. However, a slow cooling down procedure is required to guarantee the solution quality. Furthermore, its computational time escalates quadratically with the increasing number of modules to identify. Therefore, more efficient algorithms are needed for discovering fine-grained functional modules in genome-scale biological networks.

In this paper, an efficient optimization method–subgradient with path generation(SGPG) is proposed to solve this difficult non-convex combinatorial optimization problem. In order to achieve results close to global optima, SGPG combines the convex programming method, which uses subgradient (SG) to efficiently obtain the local optima, and a heuristic path generation (PG) strategy, which makes use of the obtained local optima to search for better solutions. We have applied our SGPG as well as SA for functional module identification in two large-scale protein-protein interaction (PPI) networks: *Saccharomyces cerevisia* (*Sce*) PPI network from the Database of Interacting Proteins (DIP) [11] and *Homo sapien* (*Hsa*) PPI network collected from the Human Protein Reference Database (HPRD version 9) [12]. The results demonstrate that our new SGPG method achieves competitive performance numerically and biologically comparing to SA but with significantly reduced computation time. Furthermore, we have implemented SGPG and the Markov Clustering (MCL) algorithm [13] to find fine-grained modules of these two PPI networks. The results reveal that SGPG can identify additional biologically meaningful modules that MCL may miss, which may provide us a better understanding of the functional organization of these PPI networks.

Blockmodel module identification

We first review the blockmodel module identification framework proposed in [2,7,10]. Any given network can be represented as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ denotes the set of N network nodes in $\mathcal{G}$, and $\mathcal{E}$ is the set of edges. The topology of the network $\mathcal{G}$ can be represented by an $N \times N$ adjacency matrix A, where each entry A_{ij} represents the interaction between nodes v_i and v_j. The blockmodel framework introduces the image graph $\mathcal{M} = \{\mathcal{U}, \mathcal{I}\}$ to abstract the function roles of the nodes in the original network and to outline the primary interactions among functional modules. In the image graph $\mathcal{M}$, $\mathcal{U} = \{u_1, \dots, u_q\}$ represents the set of virtual module nodes in the module space and $\mathcal{I}$ preserves the interactions within $\mathcal{U}$. The topology of $\mathcal{M}$ also can be represented by a $q \times q$ adjacency matrix B, where the entry B_{rs} denotes the interaction between modules u_r and u_s. For module identification, the mapping τ assigns N nodes in the original network $\mathcal{G}$ to q different modules in the image graph $\mathcal{M}$.

Assuming that we know the image graph adjacency matrix B, the optimization criterion of blockmodel framework is to minimize the mismatch between A_{ij} and B_{rs} by identifying an optimal mapping τ with resepct to the error function [2]:

$$E(\tau, B) = \frac{1}{M} \sum_{i \neq j}^{N} (A_{ij} - B_{\tau_i \tau_j})(w_{ij} - p_{ij}), \qquad (1)$$

in which w_{ij} denotes the weight of the corresponding edge in $\mathcal{G}$ (in this paper $w_{ij} = A_{ij}$); $M = \sum_{i \neq j}^{N} w_{ij}$ is used to restrict $E(\tau, B)$ between 0 and 1; and p_{ij} denotes the penalty of mismatching for the corresponding absent edges, which can be determined by $p_{ij} = \frac{\Sigma_{k \neq i} w_{ik} \Sigma_{l \neq j} w_{lj}}{\Sigma_{k \neq l} w_{kl}}$ [10].

In (1), we note that $E(\tau, B) = 0$ when the image graph preserves all the edges of the original network ($B_{\tau_i \tau_j} = A_{ij}$); Otherwise, $E(\tau, B) > 0$, meaning that the image graph does not preserve all the edges in the original network. When the image graph B has a mismatch for the absent edge between nodes v_i and v_j, it introduces error p_{ij}; It contributes w_{ij} - p_{ij} to the error function in (1) when it has a mismatch for the existing edge between nodes v_i and v_j. Because A_{ij} is constant, therefore, minimizing $E(\tau, B)$ is equivalent to maximizing $\frac{1}{M} \sum_{i \neq j}^{N} (w_{ij} - p_{ij}) B_{\tau_i \tau_j}$, which can be rewritten as $\max_{\tau, B} \frac{1}{2M} \sum_{i \neq j}^{N} (w_{ij} - p_{ij})(2B_{\tau_i \tau_j} - 1)$ by using the binary trick. With that, we can formulate the objective function as

$$Q(\tau, B) = \frac{1}{2M} \sum_{r,s} \sum_{i \neq j}^{N} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s} (2B_{rs} - 1),$$

where $\delta_{\tau_i r}$ is the indicator function that takes value 1 when $\tau_i = r$ and 0 otherwise. In order to maximize $Q(\tau, B)$, B_{rs} should be set to "1" when its corresponding term $\sum_{i \neq j}^{N} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s}$ is larger than 0, and 0 otherwise. Hence, the optimal solutions of τ and B are naturally decomposed. The optimal image graph B is a byproduct of the optimal mapping τ, which maximizes the following objective function:

$$Q^*(\tau) = \frac{1}{2M}\sum_{r,s}^{q}\left|\sum_{i\neq j}^{N}(w_{ij}-p_{ij})\delta_{\tau_i r}\delta_{\tau_j s}\right|. \tag{2}$$

The optimization problem (2) is NP hard [2,14]. In [10], SA has been proposed to solve the optimization problem, which has the time complexity escalating with the increasing q. In our biological application, the search space for annealing parameters also increases with the increasing q. To find a large number q of functional modules in large-scale networks, SA is a very time-consuming algorithm.

Subgradient with path generation (SGPG)

We propose to speed up the blockmodel module identification problem by convex programming combined with a heuristic path generation method. The basic idea is first to use the fast subgradient (SG) convex programming method to obtain the local optima, then use path generation (PG) to search for better solutions to reach global optima. We note that PG is originally proposed in this paper as a new useful heuristic to combine with subgradient algorithms to efficiently solve the hard combinatorial optimization problem. The combination of SG (time complexity $O(qN^2)$) and PG (time complexity $O(q^2N^2)$) can dramatically reduce the computational time with competitive performance compared to SA method.

Subgradient convex programming (SG)

Blockmodel module identification in matrix form

We now reformulate the module identification problem (2) into a matrix form by introducing an assignment matrix S corresponding to the module mapping τ. To identify q non-overlapping functional modules in $\mathcal{G}$, the assignment matrix S is defined as an $N \times q$ matrix with each entry $S_{ir} = 1$ when v_i is assigned to the module u_r or equivalently, $\tau_i = r$; and $S_{ir} = 0$ otherwise. In other words, $S_{ir} = \delta_{\tau_i r}$. Each column in S corresponds to an image graph module node in which all the assigned network nodes take the value "1". We further use W to denote the weight matrix with each entry as the corresponding edge weight w_{ij}, and P as the penalty matrix with each entry as the corresponding penalty p_{ij}. The objective function in (2) can be rewritten in the following equivalent matrix form:

$$Q^*(\tau) = f(S) = \left\|S^T(W-P)S\right\|_{L_1}, \tag{3}$$

in which the sum of each row in S has to be the unity and the columns of S are orthogonal to each other. In addition, if we assume that each node has to be assigned to one module, the assignment matrix S has to satisfy the normalization condition $S1_q = 1_N$, in which 1_q and 1_N denote the q-dimensional and N-dimensional vectors of all ones. Hence, the optimal solution for the assignment matrix S lies in the space $\phi = \{S \in \{0,1\}^{N\times q},\ S1_q = 1_N\}$, we have the convex programming formulation:

$$\begin{aligned}&\min_S : F(S) := -\left\|S^T(W-P)S\right\|_{L_1}\\ &s.t.\ \ S \in \phi.\end{aligned} \tag{4}$$

Note that we have converted our maximization problem into a minimization problem for the convenience of introducing subgradient methods in convex programming [15]. We denote $Q = S^T\ (W - P)\ S$ with its entries $Q_{rs} = S_r^T(W-P)S_s$, where S_r is the rth column of S. Again, with the optimal assignment matrix S, we can derive the topology of the image graph B: $B_{rs} = 1$ if $Q_{rs} > 0$, and 0 otherwise.

Subgradient

The optimization problem (4) is a non-smooth combinatorial optimization problem as the objective function involves the L_1 norm of the matrix Q. To solve this hard optimization problem, we first relax the binary constraints $S \in \{0,1\}^{N\times q}$ in (4) by the continuous relaxation $S \in [0,1]^{N\times q}$ and use γ to represent the relaxed constraint set, which is a convex hull after relaxation. To address the nonlinearity of the matrix L_1 norm objective function $F(S) = -\|Q\|_{L_1}$ with the relaxed linear constraints, we propose to use Frank-Wolfe algorithm [15] to iteratively solve the following optimization problem with a linear objective function from the approximation by the first-order Taylor expansion:

$$\begin{aligned}&\min_S : F(S^t) + < \nabla F(S^t), (S - S^t) >\\ &s.t.\ \ S \in \gamma,\end{aligned} \tag{5}$$

where S^t is the current solution, <, > is the inner product operator, and the new objective function is from the first-order Taylor expansion. The problem (5) at each iteration is a linear programming problem to search for the local extreme point along the gradient $\nabla F(S^t)$ as in steepest descent. However, as previously stated, $F(S^t)$ takes the matrix L_1 norm, which is non-smooth, and therefore non-differentiable. To address this last complexity, we apply subgradient methods [15] to replace $\nabla F(S^t)$ by a subgradient $\partial F(S^t)$ instead [16]:

Definition (Subgradient): A matrix $\partial F \in \mathcal{R}^{N\times q}$ is a subgradient of a function $F : \mathcal{R}^{N\times q} \to R$ at the matrix $X \in \mathcal{R}^{N\times q}$ if $F(Z) \geq F(X) + < \partial F, (Z - X) >, \forall Z \in \mathcal{R}^{N\times q}$.

In our case, the subgradient of the matrix L_1 norm can be presented by its dual norm–matrix L_∞ norm, which is used to derive the subgradient $\partial F(S^t)$. Similar to the derivation for the subgradient of the L_1 norm of vectors in L_1 regularization in [16], we show that the subgradient of the L_1 norm of any matrix $X_{N\times q}$ is

$$\partial||X||_{L_1} = \begin{cases}\left\{Y \in \mathbf{R}^{N\times q}; ||Y||_{L_\infty} \leq 1)\right\} & \text{if}\, X = \mathbf{0};\\ \left\{Y \in \mathbf{R}^{N\times q}; ||Y||_{L_\infty} \leq 1\ \ and\ < Y, X > = ||X||_{L_1}\right\}, & \text{otherwise};\end{cases} \tag{6}$$

where **0** is a $N \times q$ matrix of all zeros. For our module identification problem, we have the following proposition derived from (6):

Proposition: The subgradient of the objective function of our relaxed optimization problem $F(S)$ at the assignment S^t can be defined as: $\partial F(S^t) = 2(P - W)S^t\bar{Q}$. In our implementation, we choose

$$\bar{Q}_{rs} = \begin{cases} \alpha & Q_{rs} = 0; \\ 1 & Q_{rs} > 0; \\ -1 & Q_{rs} < 0; \end{cases} \tag{7}$$

where α is a number between [-1, 1].

Proof: From (6), there always exists a $\bar{Q}$ satisfying $\|\bar{Q}\|_{L_\infty} \le 1$ and $\|Q\|_{L_1} = <\bar{Q}, Q>$. As $\partial\|Q\|_{L1} = \partial <\bar{Q}, Q>$ and the subgradient of differentiable functions is equal to its gradient [16], we have $\partial F(S^t) = -\partial\,[\|Q\|_{L1}] = -\partial <\bar{Q}, Q> = -\partial \mathrm{tr}(\bar{Q}^T S^{t^T}(W-P)S^t) = 2(P-W)S^t\bar{Q}$ when S^t is close to the local minima. QED. □

Convex programming algorithm

Using Frank-Wolfe algorithm with the derived subgradient, we now have a conditional subgradient method [16] to iteratively solve the relaxed optimization problem as shown in the pseudo-code given in the following:.

```
Algorithm: Conditional Subgradient
Input: initial value $S^t$, $t = 0$.
Do:
    (i) Compute the subgradient $\partial F(S^t)$.
    (ii) Solve the minimization problem:
        $S^* = \arg\min_S : <\partial F(S^t), S> s.t.\ S \in \gamma$
    (iii) Linear search for the step in the direction $S^* - S^t$ found in (ii), update $S^t$, $t = t + 1$.
Until: $|\Delta F| + ||\Delta S^t|| < \zeta$
Output: $S^t$.
```

In this algorithm, step (ii) at each iteration can be solved using a generic linear programming solver in $O((qN)^{3.5})$. However, due to the special structure of the optimization problem, we instead solve it as a semi-linear assignment problem. As the assignment matrix $[\partial F(S^t)]_{N \times q}$ is not a square matrix, the optimization in step (ii) can be efficiently solved by assigning node i to module r, which is the index of the largest entry in row i of subgridient $\partial F(S^t)$, with the time complexity $O(qN)$.

To derive the solution to the original problem (4) from the results of the relaxed problem by the conditional subgradient algorithm, we recover from the relaxed solution to a closest feasible solution by a simple rounding up strategy. Finally, we note that the presented conditional subgradient algorithm converges to a local stationary point of the combinatorial optimization problem (4) due to the non-convex nature of the objective function (3) with the worst case complexity $O(qN^2)$ [15]. Hence, good initialization is critical to get high quality results. In our current implementation, we initialize S^t by a modified Expectation-Maximization (EM) algorithm presented in [8].

Path generation (PG)

In order to make use of the local optima found by the above fast subgradient method, we propose a novel path generation method for our combinatorial optimization problem. The path generation method aims to conserve the overlap between two local optima, and get improvement based on the overlap which contributes significantly to the objective function value. Our new path generation is inspired by the path relinking method which connects two combinatorial local optima and try to find better results along the linking path [14]. However, our method does not relink two local optimal results but creates new paths by extracting potentially useful overlap between them.

The essential idea of the path generation method is to construct new results by preserving the overlap between modules from two local optima that contributes significantly to the objective function. Given two solutions x_A and x_B from SG as the new path generators, PG generates new results and explores the search space while maintaining the current productive overlap between x_A and x_B. Let $N_r(x_A)$ denote the module u_r of x_A and $N_s(x_B)$ the module u_s of x_B. The contribution $S(r_A, s_B)$ from the overlap $Over(r_A, s_B) = \{d | d = N_r(x_A) \cap N_s(x_B)\}$ is defined as:

$$S(r_A, s_B) = \left\|s_{AB}^T(W-P)S_A\right\|_{L1} + \left\|s_{AB}^T(W-P)S_B\right\|_{L1}. \tag{8}$$

Here, s_{AB} is a binary vector, of which each entry is equal to 1 when the corresponding node is in both $N_r(x_A)$ and $N_s(x_B)$, and 0 otherwise. The value of $S(r_A, s_B)$ is the shared contribution to the objective function Q^* in (2) between $N_r(x_A)$ and $N_s(x_B)$ for two feasible solutions. S_A and S_B are assignment matrix of the two solutions. The most promising overlap between modules r and s are determined by

$$(r_A, s_B) = argmax\{S(r, s) : r, s \in \{1, \ldots, q\}\}. \tag{9}$$

The path generation based on (9) proceeds in the following manner: First, the most promising overlap $Over(r_A, s_B)$ between modules r_A and s_B of the initiating solution x_A and the guiding solution x_B is identified by (9), then r_A is locally adjusted to become $Over(r_A, s_B)$ by removing nodes. After the adjustment, a new solution x_1 is generated and $C_A = \{r_A\}$ and $C_B = \{s_B\}$, where C_A and C_B denote the sets of used modules in both solutions, respectively. Local search is then applied to find the improved x_1^*. Then we preserve x_1^* and let $x_A = x_1^*$. The above procedure is repeated until no overlap exists or it reaches other relaxed termination conditions, for example, we can set $N_{stop} = 5$ meaning that there are no larger than five nodes in the overlap of the modules from two solutions. Finally, we obtain the best solution along the generated paths. The whole procedure is illustrated in the following pseudo code:

```
Algorithm: Path Generation Method
Input: $x_A$, $x_B$, $x$, $x_{best}$, $N_{stop}$, $C_A = \emptyset$, $C_B = \emptyset$, $Over = +\infty$, $Q_{best} = -\infty$
While($Over > N_{stop}$)
    (1) $(r_A, s_B) = argmax\{S(r, s): r, s \in \{1, ..., q\}\}$ and find $Over(r_A, s_B)$;
    (2) modify nodes from $r_A$ in $x$ to make $N_r(x) = Over(r_A, s_B)$ and $C_A = \{r_A\}$, $C_B = \{s_B\}$;
    (3) $(Q_x^*, x^*) = \text{LocalSearch}(x)$;
    (4) If $(Q_x^* > Q_{best})$
    (5)     $Q_{best} = Q_x^*$ and $x_{best} = x^*$;
    (6) EndIf
    (7) $x_A = x^*$ and find the next $Over$ set using (9);
EndWhile
Output: $x_{best}$ and $Q_{best}$.
```

To illustrate how PG works, an example of the path generation procedure is shown in Figure 1. The module organization of the given network is shown in Figure 1A. Assume $x_A = \{\{1, 2, 4\}, \{5, 6\}, \{3, 7\}\}$ with $Q^*_{x_A} = 0.201$ and $x_B = \{\{1, 2\}, \{3, 4\}, \{5, 6, 7\}\}$ with $Q^*_{x_B} = 0.238$. Starting with $C_A = C_B = \emptyset$, a path is generated. At the first step, the most productive overlap between module $r_A = u_1^A$ in x_A and $s_B = u_1^B$ in x_B is identified, and a new solution x_1 is obtained by modifying $r_A = u_1^A$ to be the same as $Over(u_1^A, u_2^B)$ with $Q_1 = 0.201$. Update $C_A = \{u_1^A\}$ and $C_B = \{u_1^B\}$. Local search further improves the solution to obtain x_1^* with $Q_1^* = 0.374$ and then set $x_A = x_1^*$. Next, module $r_A = u_2^A$ and $s_B = u_2^B$ have the overlap with the largest contribution to the objective function. By similarly modifying $N_2(x_A) = Over(u_2^A, u_2^B)$, we then generate a path x_2^* with $Q_2^* = 0.374$ and update $C_A = \{u_1^A, u_2^A\}$ and $N_3(x_A) = Over(u_3^A, u_3^B)$. In the end, we make $N_3(x_A) = Over(u_3^A, u_3^B)$ and get the final path x_3^* with $Q_3^* = 0.374$ and update $C_A = \{u_1^A, u_2^A, u_3^A\}$ and $C_B = \{u_1^B, u_2^B, u_3^B\}$. The PG algorithm is executed as in Figure 1 with the time complexity $O(q^2N^2)$.

Experimental results

We have implemented our SGPG method to identify functional modules in two biological networks: *Saccharomyces cerevisia* PPI network from the Database of Interacting Proteins (DIP) [11] and *Homo sapien* PPI network from the Human Protein Reference Database (HPRD) [12]. We first show the efficiency of SGPG comparing to the previous algorithms based on SA for functional module identification in two networks with q = 10, 50 and 100. We further evaluate the potential of SGPG to identify biologically meaningful modules by contrasting the differences of the identified fine-grained modules (q = 500 for *Homo sapien* PPI network and q = 300 for *Saccharomyces cerevisia* PPI network) detected by MCL algorithm [13]. We show that SGPG can unearth certain kinds of biologically meaningful modules that may not be detected by MCL.

Performance comparison between SA and SGPG

We first compare SA and SGPG for module identification in two PPI networks with relatively small q = 10, 50, and 100 as SA requires very slow cooling down procedures to guarantee the solution quality when q >100. The *Homo sapien* PPI network has a largest component of 9,270 nodes and 36,917 edges. The upper bound of the objective function value in (2) $Q^*_{max} = 0.98$ when we consider the original network itself as the image graph with q =9,270. We also have implemented our algorithm to the *Saccharomyces cerevisia* PPI network, which has a largest component of 4990 nodes and 21,911 edges with the upper bound $Q^*_{max} = 0.97$ when q =4,990.

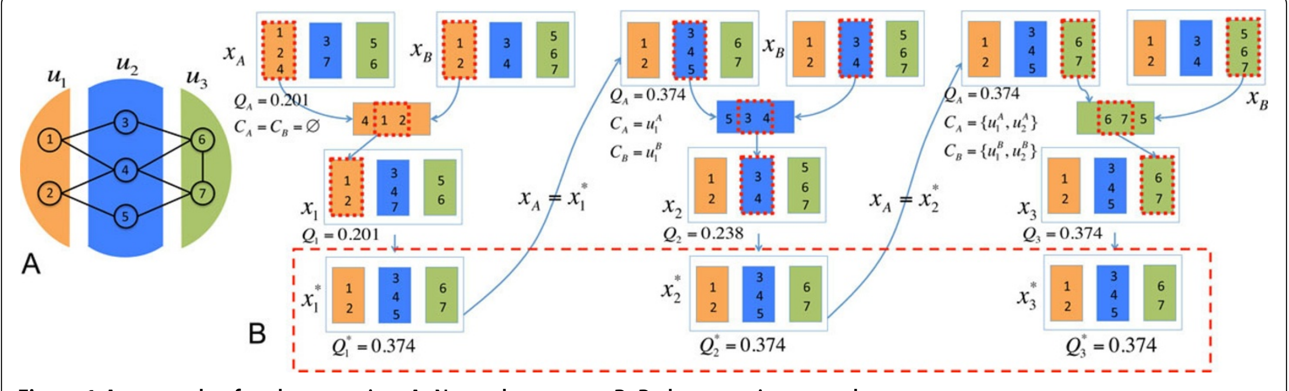

Figure 1 An example of path generation: A. Network structure; B. Path generation procedure.

The parameter settings for SA and SGPG are listed in Table 1. The starting temperature and cooling down procedure are two critical parameters that determine the performance of SA. In our implementation, the starting temperature is set high enough and the cooling down procedure is set slow enough to avoid freezing in metastable states. For SGPG, we set the number of local optima N_{set} to 10 and the terminal condition N_{stop} = 5 for the minimum requirement of the overlap set *Over* in path generation.

Table 2 shows the comparison of the fitting score Q^* computed by (2) and the running time between SA and SGPG. Because there is no ground truth of the exact functional modules in both networks, we use the fitting score Q^* as the criterion of the optimization performance. All the experiments are programmed in C++ and run on a MacPro Station with a 2.4 GHz CPU and 6 Gb RAM. From Table 2 we find that the quality of the solutions computed by SGPG is very competitive to the solutions of SA with the largest gap 3.9% in Q^* when q = 100 for *Homo sapien* PPI network. Meanwhile, SGPG is significantly faster than SA. Table 2 also reveals that the computation time of SA grows quadratically with increasing q, however, the computation time of SGPG grows sub-quadratically with q since the time complexity of SG and PG are $O(qN^2)$ and $O(q^2N^2)$ respectively. This makes a big difference when we need to identify a large number of modules. For example, SA takes more than two months for detecting q = 300 modules for the *Homo sapien* PPI network, while SGPG only requires around two days to obtain the results.

To further demonstrate whether these two different blockmodel methods have the potential of discovering biologically meaningful modules, we perform Gene Onotolgy (GO) enrichment analysis for the modules identified by both methods using GoTermFinder [17]. Figure 2 displays the comparison of the number of significantly enriched modules with different q detected by both SA and SGPG. From both figures, we find that SGPG achieves competitive performances on identifying GO enriched modules comparing to SA.

Comparison between SGPG and MCL

In order to verify the biological significance of the modules identified by blockmodel identification, we have implemented both SGPG and MCL to detect fine-grained modules for both *Saccharomyces cerevisia* and *Homo sapien* PPI networks. Because SA will take months to obtain results with q >200, we only have applied SGPG in this section. By analyzing the identified modules detected by two methods, we have found that SGPG can discover a comparable number of GO enriched modules as MCL detects. More importantly, SGPG discovers additional biologically meaningful modules in which proteins are sparsely connected but have the same interaction patterns to the rest of the network.

Saccharomyces cerevisia PPI network

We have identified fine-grained modules for the *Saccharomyces cerevisia* PPI network using SGPG and MCL. We set q = 300 for SGPG and the inflation parameter I = 1.5 for MCL, which identified 370 modules in total. Within these identified modules, 296 modules by SGPG and 307 modules by MCL have more than two nodes. From these, we have found 150 and 153 modules respectively with significantly enriched GO terms below 1% after Bonferroni-correction by GoTermFinder. SGPG performs competitively to MCL. But more importantly, we find that SGPG can detect sparsely connected modules with certain interaction patterns that MCL fails to detect.

In order to scrutinize the differences between the modules discovered by SGPG and MCL. We have annotated all modules by KOG categories [18]. For each module, the most KOG category assigned to the proteins in the module is annotated to that module. Figure 3A displays the percentage of the KOG annotated modules. Obviously, the percentages of modules annotated to KOG U, K, J and T are different (difference is larger than 2.5%) for the results from two methods. Specifically, SGPG detects more modules with KOG U, K and T annotations. To further examine the functionalities of different KOG categories, we discover that proteins annotated to KOG U play roles in intracellular trafficking, secretion, and vesicular transport; proteins annotated to KOG K have functionalities in transcription; and proteins in KOG T behave signal transduction functionalities. In [2], the results have already illustrated that proteins annotated to KOG T and K have the sparsely connected modules structures. Blockmodel based SGPG has successfully discovered more such modules than MCL. For proteins in KOG J (functions in translation, ribosomal structure and biogenesis), they are supposed to have densely connected modular structures which MCL tends to detect.

To verify that SGPG does detect sparsely connected modules, we investigate the network topology of the

Table 1 Parameter settings in SA and SGPG

Para.	C_β	T_{start}	T_{end}	T_{sweep}	T_{switch}	N_{set}	N_{stop}.
SA	0.99	40	0.001	100	20	-	-
SGPG	-	-	-	-	-	10	5

Table 2 Comparison of SA and SGPG on *Homo sapien* and *Saccharomyces cerevisia* PPI networks

PPI	Method	Q*(q=10)	Time(h)	Q*(q=50)	Time(h)	Q*(q=100)	Time(h)
Homo sapien	SA	0.5393	1.73	0.6530	45.07	0.7180	180.26
	SGPR	0.5346	0.5	0.6452	1.95	0.6898	6.35
Saccharomyces cerevisia	SA	0.5692	1.35	0.6834	25.02	0.7544	102.65
	SGPR	0.5690	0.3	0.6752	1.15	0.7292	3.34

proteins of all the modules annotated to KOG U, K, J and T. We count the number of sparsely connected modules, which have the interaction density of module less than 3%, annotated to KOG U, K, J and T. The specific comparison is in Table 3. Table 3 illustrates the difference of network topology by the average module density and the average clustering coefficient among proteins detected by both SGPG and MCL. The average clustering coefficients of the induced network, which is extracted from the original *Saccharomyces cerevisia* PPI network based on the proteins of modules annotated to certain KOG categories, are calculated by the definition in [19]. Larger average clustering coefficients indicate that proteins have densely connected modular structures [19]. From the table, we find there is a trend that the average module density and the average clustering coefficient of the modules identified by MCL are larger than those detected by SGPG, which means MCL detects more densely connected modules while SGPG can identify sparsely connected modules.

Figure 4 illustrates an induced subnetwork of sparsely connected modules discovered by SGPG from the *Saccharomyces cerevisia* PPI network. Only the interactions among the proteins in Figure 4 are displayed. As shown in Figure 4, modules A and B are both sparsely connected modules, in which there is no interaction among proteins. Module C is a topologically cohesive module. Modules A, B and C are all significantly enriched in GO terms related to KOG G (carbohydrate transport and metabolism), T (signal transduction mechanisms) and C (energy production and conversion) respectively. From the structure illustrated in Figure 4, we find that proteins in module B play roles in passing signal between proteins of hexokinase activity and nucleoside phosphate metabolism. Furthermore, we notice that the marked patterns I and II are two types of interaction patterns across these modules, which tend to be clustered into the same module when using MCL. Figure 4 clearly displays the advantage of SGPG, which is to identify modules by their interaction patterns and functional roles rather than their interaction density.

Additionally, Table 4 illustrates three sparsely connected module examples including module B in Figure 4, which are all detected by SGPG but missed by MCL. These three modules are annotated to KOG U and T respectively, which cannot be detected by MCL no matter what inflation parameter we choose.

Homo sapien PPI network

For the *Homo sapien* PPI network, we set SGPG to identify q = 500 modules with the same settings in Table 1. For MCL, we set its inflation parameter I = 1.5 and have found 450 modules. We have performed GO enrichment analysis for these identified modules with more than two nodes (478 from SGPG and 380 from MCL). Based on GoTermFinder, 269 modules from SGPG and 265 modules from MCL are significantly enriched with p-values below 1% after Bonferroni-correction. SGPG has discovered a competitive number of GO enriched modules compared to MCL. We also note that the modules identified by SGPG are relatively smaller than those from MCL and these modules have more

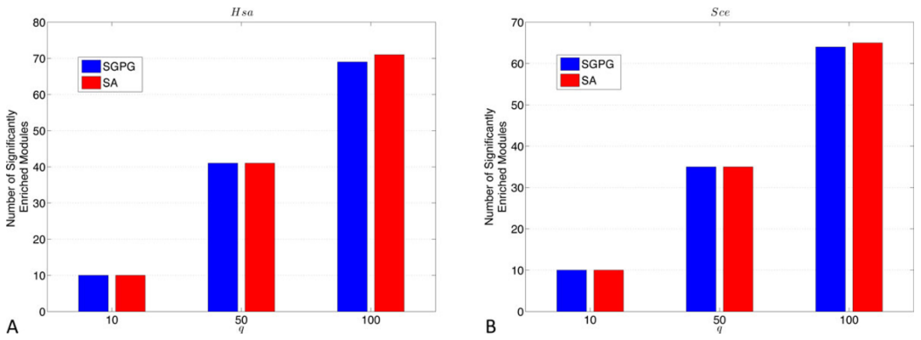

Figure 2 Comparison between SA and SGPG for the number of identified modules of *Homo sapien* PPI network (A) and *Saccharomyces cerevisia* PPI network (B) that have significantly enriched GO terms below 1% after Bonferroni-correction in either biological process, molecular function, or cellular compartment.

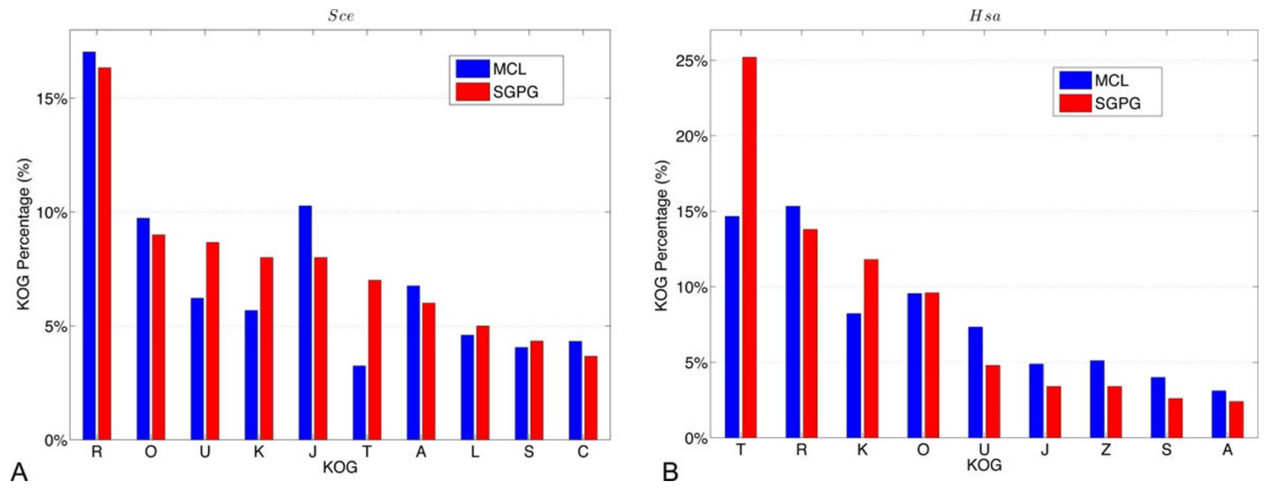

Figure 3 Percentage of different categories of modules detected by SGPG and MCL (annotated by KOG). A. KOG percentage of *Saccharomyces cerevisia* PPI network. B. KOG percentage of *Homo sapien* PPI network.

specific enriched functionalities and may provide more detailed information for future catalog of functional modules. More importantly, SGPG detects several modules with interesting functionalities that MCL has missed.

Following the same analysis method used in the previous section, we first annotate all the identified modules with KOG categories to scrutinize the differences between modules detected by SGPG and MCL. Figure 3B shows the percentages of the modules annotated to different KOG categories by both methods. Obviously, SGPG detects more modules annotated in KOG T and K categories, within which functional modules tend to have sparsely connected structures. However, MCL discovers more modules annotated in KOG U, within which functional modules tend to have a densely connected structure in the *Homo sapien* PPI network.

Table 5 further consolidates that the modules detected by SGPG can detect sparsely connected patterns that MCL may miss. The average density and average clustering coefficient both indicate that modules discovered by MCL have cohesive modular structure, while modules discovered by SGPG are sparsely connected.

Figure 5 illustrates an induced subnetwork discovered by SGPG from the *Homo sapien* PPI network. Only the interactions among the proteins in Figure 5 are exhibited. As shown in Figure 5, modules A, B and C are all sparsely connected modules, which have no interactions inside the modules. Proteins in module D only have a few connections. Modules A and B are annotated to KOG K (transcription). While module D is annotated to KOG T (signal transduction mechanisms). Module C is annotated to both KOG T and K. Module C contains proteins SMAD2 and SMAD3 which play an important role in tumor generation [20]. From the module structure in Figure 5, we find that SMAD2 and SMAD3 have intimate relationships to proteins of DNA binding, cellular response and kinase activity, which is useful to help us to obtain a better understanding of their functionalities and influence on other proteins. Furthermore, the two interaction patterns preserved in the module structure are detected by SGPG, which is difficult for MCL to identify.

Table 3 Topological analysis of different KOG categories in *Saccharomyces cerevisia* PPI network

KOG ID	Method	proteins	sparse modules/modules	Avg. density	Avg. clustering coef.
U	SGPG	353	15/26	2.98%	0.0814
	MCL	256	0/21	27.38%	0.2402
K	SGPG	359	6/24	6.68%	0.1352
	MCL	361	0/19	26.35%0	0.1834
J	SGPG	579	9/24	9.16%	0.0678
	MCL	358	0/25	37.90%	0.1429
T	SGPG	169	13/21	3.47%	0.0755
	MCL	94	0/12	31.31%	0.0912

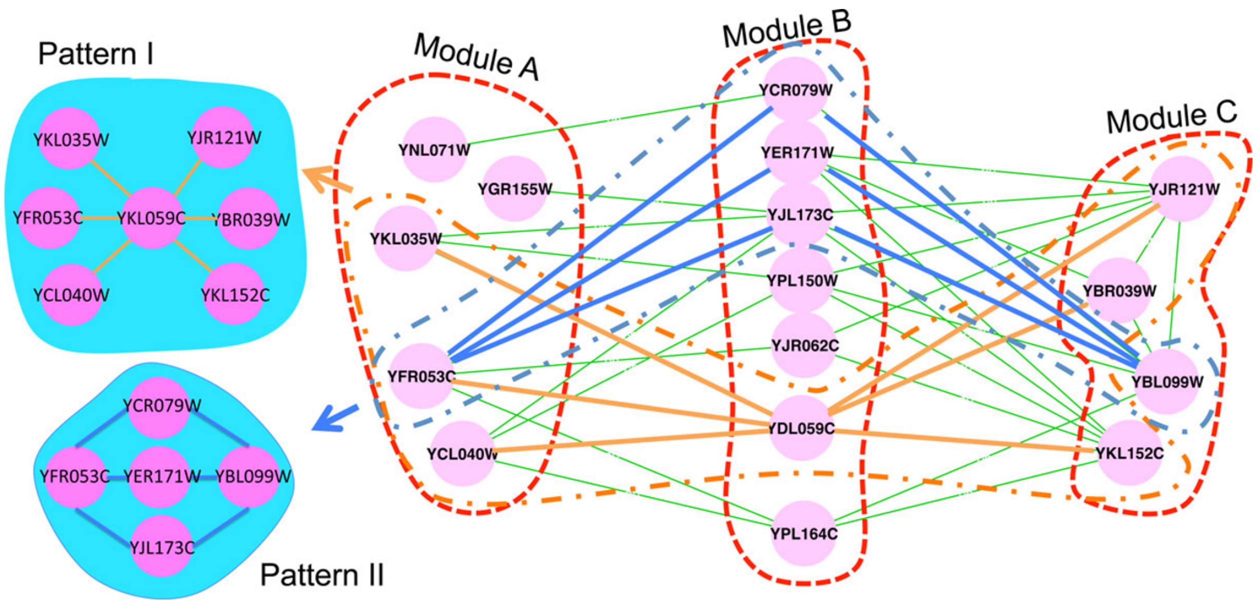

Figure 4 A subnetwork with sparsely connected modules detected by SGPG. Module A is enriched in hexokinase activity with p-value 1.71e-5. Module B is enriched in response to endogenous stimulus with p-value 4.77e-5. Module C is enriched in nucleoside phosphate metabolism with p-value 3.43e-6. Patterns I and II are two specific interaction patterns in the subnetwork.

Table 4 Sparse modules in U and T KOG categories for *Saccharomyces cerevisia* PPI network

KOG ID	Sparse module example	Enriched genes	Enriched GO Term	GO Level	p-value
U	YDR179C, YNL287W, YDL216C YCR099C, YIL004C,YAL026C YLR268W, YLR093C, YPR163C YPR148C, YOL064C, YOL117W YGL084C, YLR031W, YIL076W YPL179W, YKL191W, YPL010W	YOL117W, YDR179C, YDL216C	protein deneddylation	[+8, 0]	2.01e-5
T	YJL092W, YDR490C, YOR231W YJL005W, YPL074W, YPL083C YNL323W, YOL100W	YDR490C, YOL100W, YNL323W, YJL005W, YOR231W	signal transduction	[+3, -1]	6.09e-5
T	YDR076W, YDL059C, YJL173C YPL164C, YER171W, YPL026C YCR079W, YPL150W, YHR169W YJR062C	YDL059C, YPL026C, YER171W, YPL164C, YJL173C, YDR076W	response to endogenous stimulus	[+2, -1]	4.77e-5

Table 5 Topological analysis of different KOG categories in *Homo sapien* PPI network

KOG ID	Method	proteins	sparse modules/modules	Avg. density	Avg. clustering coef.
T	SGPG	1970	59/126	4.91%	0.0822
	MCL	2481	0/66	26.32%	0.1696
K	SGPG	878	27/59	3.15%	0.0779
	MCL	916	0/37	30.41%0	0.1928
U	SGPG	592	3/24	4.95%	0.0448
	MCL	517	0/33	31.42%	0.1359

Table 6 lists three sparsely connected module examples detected by SGPG but missed by MCL. These three modules are annotated to KOG T and K respectively, which cannot be detected by MCL no matter what inflation parameter we choose.

Discussion

At present, most of the module identification methods for biological networks aim to find densely connected modules but ignore sparely connected modules, which can be manifested in biological systems due to their

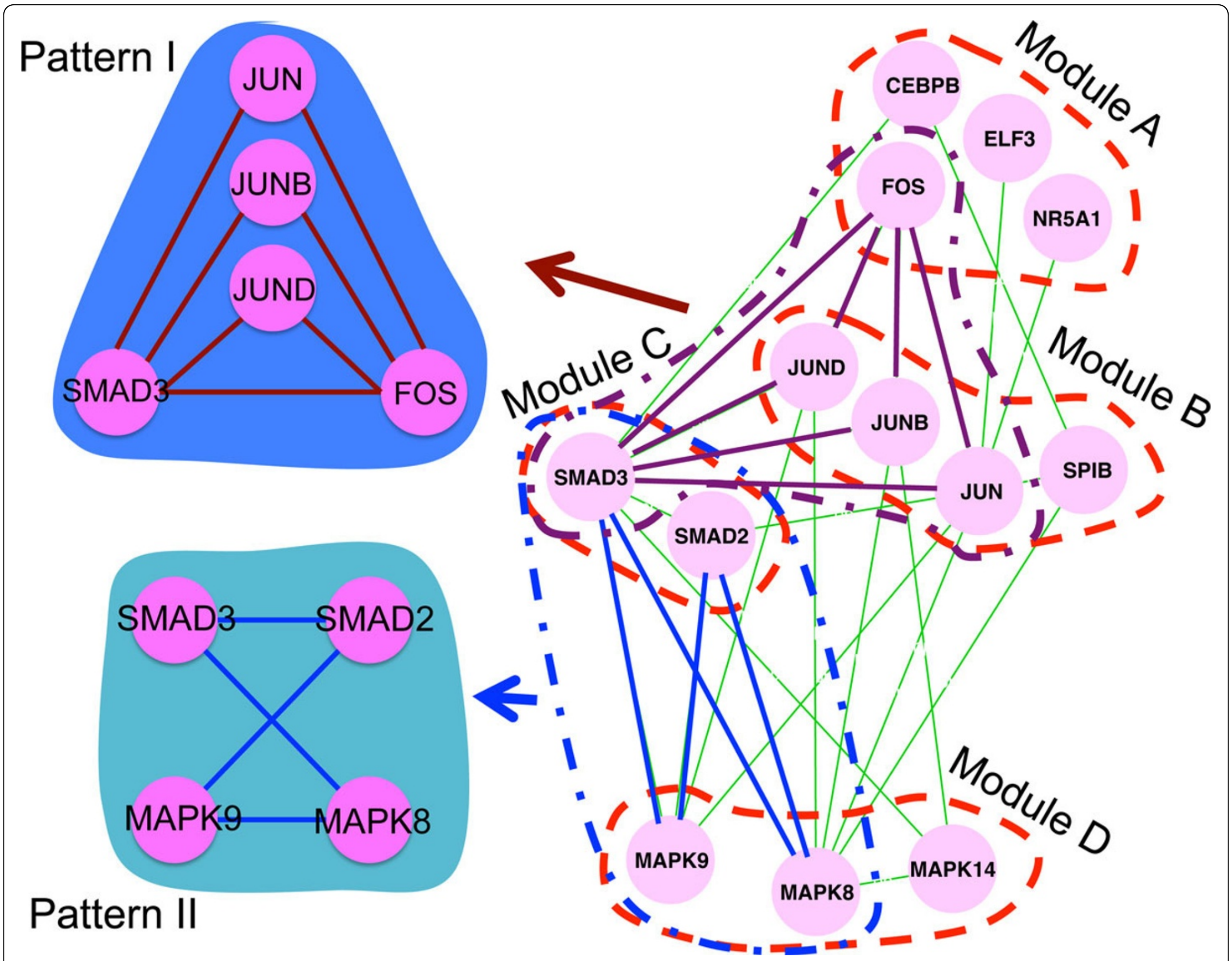

Figure 5 A subnetwork with sparsely connected modules detected by SGPG. Module A is enriched in sequence-specific DNA binding with p-value 9.91e-7. Module B is enriched in cellular response to calcium ion with p-value 4.04e-7. Module D is enriched in MAP kinase activity with p-value 8.60e-5. Patterns I and II are two specific interaction patterns in the subnetwork.

special functionalities. Here, in order to find more biologically meaningful modules with both types of modular structures, we adopt a blockmodel framework which detects densely connected modules and sparely connected modules simultaneously as it identifies modules by the interaction patterns. Our results indicate that the real world PPI networks, such as *Saccharomyces cerevisia* and *Homo sapien* PPI networks, do have the sparely connected modules, which may not be detected by the modularity based methods such as MCL.

We have proposed a novel efficient method SGPG that combines SG and PG to solve the blockmodel functional module identification problem. Our experimental results have proven that our SGPG method can achieve competitive performance numerically and biologically but with significantly reduced computation time compared to the original SA method in [2]. We have demonstrated that SGPG can identify biologically meaningful modules, specifically the ones with sparse interactions within them but with same interaction patterns to the rest of the network,

Table 6 Sparse modules in T and K KOG categories for *Homo sapien* PPI network

KOG ID	Sparse module example	Enriched genes	Enriched GO Term	GO Level	*p*-value
T	NTRK1, NTRK3, NTRK2 VAV1, VAV3	NTRK1, NTRK2, NTRK3	neurotrophin receptor activity	[+3, -1]	2.95e-9
T	PIK3R3, PIK3R2, PIK3R1	PIK3R3, PIK3R2, PIK3R1	phosphatidylinositol 3-kinase complex	[+5, -1]	4.77e-9
K	JUN, JUNB, JUND SPIB	JUN, JUNB, JUND	cellular response to calcium ion	[+6, -1]	4.04e-7

which behave important cellular functionalities. Our future research will focus on designing more efficient algorithms to detect functional modules in large-scale biological networks. Our method can be further improved with the potential to enhance the performance. For example, the number of modules q needs to be given in our current algorithm. In [21], the authors have introduced a Bayesian strategy based on a stochastic block model to identify the module assignments as well as the optimal number of modules. However, this Bayesian approach only guarantees that the final solution converges to the local optimum. We may be able to combine the strengths from our SGPG method and the Bayesian approach to efficiently determine the optimal q in SGPG by adopting this Bayesian strategy to further improve the proposed algorithm. Also, there are some other promising efficient heuristics for global optimization, such as differential evolution [22] and genetic algorithms [23], which may also be coupled with our PG strategy to further increase the efficiency of these algorithms.

Authors' contributions

Conceived the algorithm: YW, XQ. Implemented the algorithm and performed the experiments: YW. Analyzed the results: YW, XQ. Wrote the paper: YW, XQ.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

XQ was supported in part by Award R21DK092845 from the National Institute Of Diabetes And Digestive And Kidney Diseases, National Institutes of Health; and by the University of South Florida Internal Awards Program under Grant No. 78068.

Declarations

The publication costs for this article were funded by the corresponding author's institution.
This article has been published as part of *BMC Bioinformatics* Volume 14 Supplement 2, 2013: Selected articles from the Eleventh Asia Pacific Bioinformatics Conference (APBC 2013): Bioinformatics. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcbioinformatics/supplements/14/S2.

Published: 21 January 2013

References

1. Hartwell L, Hopfield J, Leibler S, Murray A: **From molecular to modular cell biology.** *Nature* 1999, **402**:47-52.
2. Pinkert S, Schultz J, Reichardt J: **Protein interaction networks: More than mere modules.** *PLoS Comput Biol* 2010, **6**:e1000659.
3. Zinman G, Zhong S, Bar-Joseph Z: **Biological interaction networks are conserved at the module level.** *BMC Systems Biology* 2011, **5**:134.
4. Newman M, Girvan M: **Finding and evaluating community structure in networks.** *Phys Rev E Stat Nonlin Soft Matter Phys* 2004, **69**:026113.
5. Ziv E, Middendorf M, Wiggins C: **Information-theoretic approach to network modularity.** *Phys Rev E Stat Nonlin Soft Matter Phys* 2005, **71**:046117.
6. Sharan R, Ulitsky I, Shamir R: **Network-based prediction of protein function.** *Mol Syst Biol* 2007, **3**:88.
7. Reichardt J, White D: **Role modules for complex networks.** *Eur Phys J B* 2007, **60**:217-224.
8. Newman M, Leicht E: **Mixture models and exploratory data analysis in networks.** *Proc Natl Acac Sci USA* 2007, **104**:9564-9569.
9. Wang Y, Qian X: **Functional module identification by block modeling using simulated annealing with path relinking.** *ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM-BCB 12)* 2012.
10. Reichardt J: *Structure in Networks* Springer-Verlag Berlin Heidelberg; 2008.
11. Salwinski L, Miller C, Smith A, Pettit F, JU JB, Eisenberg D: **The Database of Interacting Proteins: 2004 update.** *Nucleic Acids Research* 2004, **32**:D449-D451.
12. Prasad T, Goel R, Kandasamy K, *et al*: **Human Protein Reference Database–2009 update.** *Nucleic Acids Research* 2009, **37**:D767-D772.
13. Dongen S: **A cluster algorithm for graphs.** *Technical Report INS-R0010* 2000.
14. Mateus G, Resende M, Silva R: **GRASP with path-relinking for the generalized quadratic assignment problem.** *Journal of Heuristics* 2010, 1-39.
15. Bertsekas D: *Nonlinear Programming.* 2 edition. Athena Scientific; 1999.
16. Bach F, Jenatton R, Mairal J, Obozinski G: **Optimization with sparsity-inducing penalties.** *Technical report, HAL 00613125* 2011.
17. Boyle E, Elizabeth I, Weng S, *et al*: **GO::TermFinder–open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes.** *Bioinformatics* 2004, **20**:3710-3715.
18. Lander E, Linton L, Birren B, *et al*: **Initial sequencing and analysis of the human genome.** *Nature* 2001, **409**:860-921.
19. Kaiser M: **Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks.** *New J Phys* 2008, **10(083042)**.
20. Petersen M, Pardali E, *et al*: **Smad2 and Smad3 have opposing roles in breast cancer bone metastasis by differentially affecting tumor angiogenesis.** *Oncogene* 2010, **29(9)**:1351-1361.
21. Hofman J, Wiggins C: **A Bayesian Approach to Network Modularity.** *Phys Rev Lett* 2008, **100**:258701.
22. Storn R, Price K: **Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces.** *Journal of Global Optimization* 1997, **11**:341-359.
23. Akbari Z: **A multilevel evolutionary algorithm for optimizing numerical functions.** *International Journal of Industrial Engineering Computations* 2010, **2**:419-430.

doi:10.1186/1471-2105-14-S2-S23
Cite this article as: Wang and Qian: **A novel subgradient-based optimization algorithm for blockmodel functional module identification.** *BMC Bioinformatics* 2013 **14**(Suppl 2):S23.