Clustering with Overlap for Genetic Interaction Networks via Local Search Optimization

Joseph Whitney, Judice Koh, Michael Costanzo, Grant Brown, Charles Boone, and Michael Brudno

University of Toronto
jwhitney@cs.toronto.edu

Abstract. Algorithms for detection of modules in genetics interaction networks, while often identifying new models of functional modular organization between genes, have been limited to the output of disjoint, non-overlapping modules, while natural overlapping modules have been observed in biological networks. We present CLOVER, an algorithm for clustering weighted networks into overlapping clusters. We apply this algorithm to the correlation network obtained from a large-scale genetic interaction network of *Saccharomyces cerevisia*e derived from Synthetic Genetic Arrays (SGA) that covers ~4,500 non-essential genes. We compare CLOVER to previous clustering methods, and demonstrate that genes assigned by our method to multiple clusters known to link distinct biological processes.

Keywords: graph clustering, genetic interactions, local search.

1 Introduction

Recent developments in Synthetic Genetic Arrays have enabled the mapping of quantitative genetic interactions on a genome-wide scale [1]. A central computational task in the analysis of the genetic interaction networks is to cluster genes into coherent modules. The resulting modules provide a higher-level organization of the cell, and may be mined to make new predictions about gene functions and provide insights into cellular pathways. Such unsupervised organization of a network's nodes into highly-interconnected modules is termed *graph clustering*.

Several general-purpose graph clustering methods that use a number of diverse algorithmic approaches have been applied to genetic and protein interaction networks. [2][3][4][5][6][7]The RNSC algorithm[4] is based on stochastic local search. It was developed for clustering un-weighted general graphs and has been applied to the problem of predicting protein complexes in physical networks. MCL [5] is a general-purpose algorithm for clustering weighted graphs, and has been widely used to analyze both physical and genetic interaction networks. MCL uses stochastic matrix operations to simulate flow through a weighted network, and determines clusters as groups of nodes connected by a large number of high-weight paths. Graph summarization [6] is a paradigm for clustering graphs in which the output is a coarse-grained *summary* of the input graph, with super-nodes representing groups of nodes with similar interaction

T.M. Przytycka and M.-F. Sagot (Eds.): WABI 2011, LNBI 6833, pp. 326–338, 2011.

patterns and super-edges represent groups of interactions between members of super-nodes. The *cost* of this summary can be formulated in terms of the number of single-edge additions or deletions necessary to recover the input graphs. For module detection in genetic interactions specifically, the network of genetic *profile correlations*, rather than individual genetic interactions, has been found useful for determining cohesive modules even via hierarchical clustering [8][9]. In particular Ulitsky *et al.* achieved better results using clustering models biased toward strong profile correlation within modules versus strong intra-modular "monochromatic" alleviating interactions[10], despite the known enrichment of such interactions in protein complexes. All of these methods partition a network into non-overlapping clusters. However, as many genes participate in multiple biological processes, it has been recognized that a more general approach, allowing clusters to *overlap* (so that multiple genes may participate in any number of distinct clusters), would facilitate the identification of clusters of genes which are biologically more meaningful, and possibly exhibit higher functional coherence.

The need for clustering algorithms that explicitly model overlap was first proposed by Palla *et al.* [11]. Their algorithm for identifying overlapping clusters, CFINDER, is based on identifying overlapping *k*-connected subgraphs between maximal cliques, using a simple combinatorial definition of cluster which applies only to unweighted networks. The algorithm has exponential running time on dense graphs, making it impractical for analysis of genetic interaction networks, which are weighted, dense graphs. More recently Wang *et al.* proposed HACO[7], an algorithm that extends average-linkage hierarchical clustering to allow clusters to be re-used in multiple agglomerative iterations, producing a lattice which can be used to induce overlapping clusters which are similar in cohesion. It cannot be used to find overlaps between clusters of widely differing cohesion due to an exponential increase in the number of clusters which must be considered.

Some pre-existing methods designed to find modules with particular characteristics operate by repeatedly seeding a search from a single node (as in MCODE[2]) or interaction (as in the within/between-pathway model of Kelley and Ideker [12], using a similar search algorithm to Sharan et al[13]). Implicit overlaps may exist in the cumulative output and highly-overlapping modules are identified using a threshold of overlap with a higher-scoring module – overlaps are viewed as redundancies and not a targeted feature of optimization. Finally, the MCL algorithm can occasionally produce overlapping clusters around exact symmetries; on the real data used in this study no such overlaps were observed.

In this paper we present CLOVER (CLustering with OVERlap), a novel method for clustering weighted networks into overlapping clusters. CLOVER combines the ability to deal with large, weighted networks via local search optimization. We have applied our method to two large-scale genetic interaction networks [1][8] and CLOVER was able to identify many clear instances of overlapping modularity in this dataset. We demonstrate that the modules identified by CLOVER show significant enrichment for known functional annotations, on par with previously published results, while the overlaps between clusters identify genes which indeed link distinct biological processes.

2 Methods

CLOVER is a local search optimization algorithm. Like all local search algorithms, CLOVER optimizes a cost function – an evaluation of the quality of a solution – by perturbing an intermediate solution via local moves. In this section we first present the cost function utilized in the CLOVER algorithm, which captures objective criteria for good overlapping clusters. We then describe the local search method used to optimize the cost function on an input network.

2.1 CLOVER Cost Function

The CLOVER cost function represents a trade-off between two desiderata of a clustering

- *cohesion,* represented by many, or high-weight edges within each cluster, and
- *separation*, represented by few, or low-weight edges joining vertices assigned to different clusters

while accommodating *overlap* between clusters. The intuition behind the cost function may be best understood by considering the case of a simple, unweighted graph. We *penalize* (assign a cost to) any pair of nodes that violates the cohesion or separation desiderata. For each pair of vertices u, v that are connected by an edge, but lie in different clusters, a *cross edge* penalty is charged. Similarly, for each pair of vertices u, w lying in the same cluster, but not connected by an edge a *missing edge* penalty is charged (see Figure 1 (a)). By combining these penalties over all pairs of nodes, we have a standard clustering cost function for unweighted graphs: the "cost" of a clustering is the number of edges crossing between clusters, and the number of edges "missing" from within clusters (see, e.g. [4]).

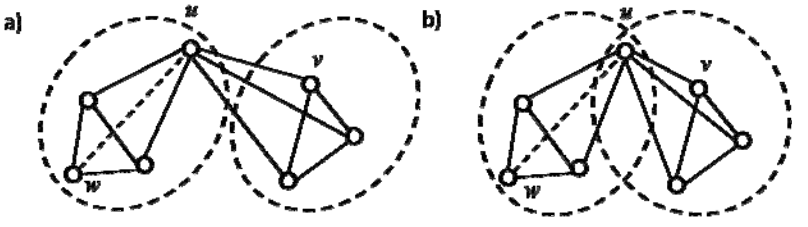

Fig. 1. a) in the non-overlapping case, there are two deviations from a perfect clustering: the edge $(\boldsymbol{u}, \boldsymbol{v})$ is a "cross edge" between vertices in different clusters while $(\boldsymbol{u}, \boldsymbol{w})$ is a "missing edge". b) $(\boldsymbol{u}, \boldsymbol{v})$ is still a cross-edge from the perspective of $\boldsymbol{v}$: $\boldsymbol{u}$ is not a cross edge from the perspective of v: u is in the same cluster. But $(\boldsymbol{u}, \boldsymbol{v})$ is a "half cross edge" from the perspective of $\boldsymbol{u}$, since it is in only one out of the two clusters of $\boldsymbol{u}$. Similarly $(\boldsymbol{u}, \boldsymbol{w})$ is a "missing edge" from the perspective of $\boldsymbol{w}$, since $\boldsymbol{u}$ is in the same cluster, but is only a "half missing edge" from the perspective of $\boldsymbol{u}$.

The CLOVER cost function generalizes this simple cost function in two ways. First, we deal with weighted networks by applying a penalty to missing and cross edges proportionate to their weights. For simplicity we assume that edge weights are scaled to lie in the interval [0,1]. For all edges within clusters we apply a penalty that is proportionate to 1 minus the weight: a cluster-internal edge with zero weight is assigned the "full" penalty while an edge with the "full" weight is assigned zero penalty. Conversely, for cross edges the penalty is directly proportional to the weight.

The second generalization deals with overlapping clusters, which requires modification of the cross-edge definition, as a single edge may be internal to one cluster, while crossing between several pairs of others. Our solution is to regard each penalty applying to an edge (u, v) as being calculated twice: once from the perspective of u and once from the perspective of v. Then for *each cluster* containing u, we apply a "cross-edge" penalty if v is not also in that cluster, and a "missing edge" penalty otherwise. The penalties are divided by the number of clusters containing u. Figure 1 (b) illustrates the application of this cost function to an example network.

The formal definition of the cost function is simplified by assuming a *complete* weighted graph where every pair of vertices u, v exists in E and is assigned a weight by ω, which may be 0. We further assume that while a clustering may assign any number of nodes to each cluster and vice versa, no node is allowed to be "unclustered" and thus not contribute to the clustering cost defined below. In practice, very small clusters (< 3 nodes) are pruned from the final results.

Formally, we define a *clustering* $\mathcal{C}$ of a weighted network $G = (V, E, \omega)$, where $\omega: E \mapsto \mathbb{R}$ assigns a weight in the range $[0,1]$ to each edge, as a collection of sets $C \subset V$, or *clusters*. We use the notation C_v to denote the set of clusters to which a vertex v is assigned by $\mathcal{C}$.

The cost function is defined as:

$$f(G, \mathcal{C}) = \sum_{u \in V} (\alpha \times x(u, \mathcal{C}, G) + (1 - \alpha) \times m(u, \mathcal{C}, G))$$

where

$$x(u, \mathcal{C}, G) = \frac{1}{|C_u|} \sum_{C \in C_u} \sum_{v \in V - C} \omega(u, v)$$

represents the total weight of edges (u, v) leading "out of" a cluster from vertex u (cross-edge cost), and

$$m(u, \mathcal{C}, G) = \frac{1}{|C_u|} \sum_{C \in C_u} \sum_{v \in C} (1 - \omega(u, v))$$

represents the total weight "missing" from each cluster containing u (missing edge cost). The real parameter α represents a trade-off between the two factors x and m, across all nodes.

The cost function calculates, for each vertex u and each cluster C, a penalty for each edge (u, v). If $v \in C$, the penalty is equal to $(1 - \alpha)(1 - \omega(u, v))$. This rewards higher-weight edges within the same cluster. If $v \notin C$, the penalty is just $\alpha \times \omega(u, v)$, penalizing higher-weight edges leading out of C from u. In the absence of overlap,

the same penalty will be applied symmetrically to v, in effect doubling all penalties. In the case of overlapping clusters, this cost is calculated *for each* cluster containing u – some of these clusters may include v, and some not. The sum of these cluster-wise penalties is then divided by the number of clusters to which the node u is assigned. An example of how this cost function promotes overlap is illustrated in Figure 2. Because both the x and m penalty factors are scaled by the number of clusters in which each vertex appears, increasing or decreasing the number of clusters to which a vertex is assigned does not increase its contribution to the cost.

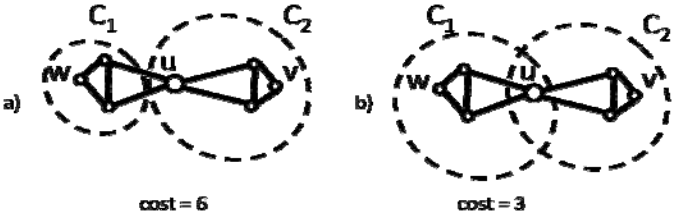

Fig. 2. Example of how the cost function promotes overlap. Edges drawn are assumed to have weight 1, edges not drawn are assumed to have weight 0. In the scenario at left, the total cost of this small clustering is 6: $\mathbf{1}$ for each of the two edges shown leaving $\boldsymbol{C_2}$ from vertex u, $\mathbf{1}$ for each of the (same) two edges leaving $\boldsymbol{C_1}$, 1 for the "missing" edge from $\boldsymbol{u}$ to $\boldsymbol{v}$, and 1 for the (same) "missing" edge from $\boldsymbol{v}$ to $\boldsymbol{u}$. In the scenario at right, the total cost is reduced to 3: $(\mathbf{2}+\mathbf{2})/\mathbf{2}$ for each edge leaving a cluster from $\boldsymbol{u}$, and $(\mathbf{1}+\mathbf{1})/\mathbf{2}$ for each "missing" edge from $\boldsymbol{u}$ to a neighbour in a shared cluster.

2.2 Local Search Algorithm

The CLOVER algorithm is an instance of the *local search* method, a type of discrete search which is typically applied to computationally difficult (e.g. NP-hard) combinatorial problems. Its goal is to minimize the value of an objective *cost function* that quantifies the fitness of a clustering as a real number, such that a "perfect" clustering has a cost of 0.

Like all local search algorithms CLOVER defines moves between *candidate solutions* – in this case, clusterings of an input network. Starting with the trivial clustering in which each vertex is assigned to one unique cluster, a new solution is constructed by applying a move, or transition operation, to the current one. There are three types of moves. The first is a *toggle* (reversal) of the inclusion of some particular vertex in some particular cluster. For a given current solution $\mathcal{C}$ for input network G, the set of possible toggle move transitions can be defined as $V \times \mathcal{C}$. The second type is an *all-in*: given an ordered pair of clusters (C, D), assign to D every vertex in C which is not already so assigned. The third type is an *all-out*: given clusters (C, D), remove from D every vertex in C which is currently assigned to D. CLOVER chooses the transition which will result in a new solution with lowest cost, b evaluating the cost of all clusterings that can be obtained by one transition from the current solution.

Any optimization search algorithm needs some mechanism for hill-climbing – that is, of moving the search away from local minima. To this end CLOVER follows a variant of *tabu search*. A list of recently moved nodes, the *tabu list*, is maintained,

and nodes on the tabu list are not moved again until a pre-defined number of moves τ has passed. We have used a τ value of 50, having determined experimentally that larger values up to 150 slow convergence to a solution without obtaining a better-scoring result.

Inherent to a local search algorithm is the need to determine a stopping condition, since it is impossible to know whether the best possible solution has been found. In some cases the stopping condition may simply be a bounded number of transitions or total running time. Either of these could be used with CLOVER, but for the results reported in this study we have used a convergence criterion: the algorithm will continue to iterate through solutions until some sufficiently large number of transitions has been made without encountering a better (lower-cost) solution than the best recorded so far. The convergence interval should be large relative to the tabu list length, so that the algorithm may return repeatedly to moves previously marked tabu. We used a convergence interval of 500 moves in all cases reported here. We discarded clusters containing fewer than three nodes – these nodes may still be reported in other, larger clusters.

The performance of CLOVER for a particular application depends greatly on the value of the α parameter. This parameter allows one to trade-off the size of the clusters and their connectivity: a lower α results in clusters which are smaller and more strongly connected, while a higher α results in clusters which are larger and more weakly connected. In Figure 3 (a) we plot the GO enrichment precision, recall, and F-measure of all the clusterings obtained with different values of α for the Costanzo et al [1] data. For comparison, we also plot the same metric for MCL with different values of the inflation parameter *I* in Figure 3 (b). Because the F-measure combines precision and recall into a single value, it is not as sensitive to cluster granularity. For this particular data set, the best results are achieved over a relatively wide plateau corresponding to α settings 0.55 - 0.75.

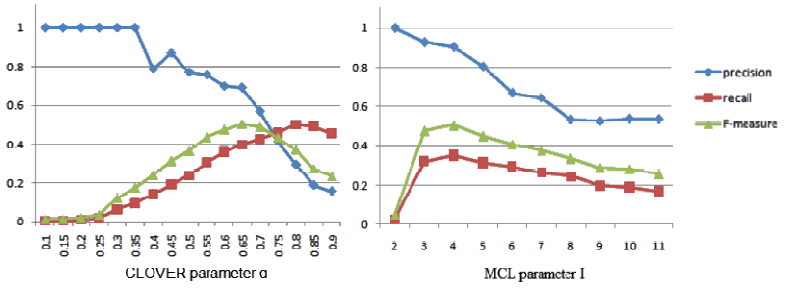

Fig. 3. Precision, recall and F-measure of GO enrichment for clustering obtained from CLOVER and MCL

3 Results

In this section we describe the application of CLOVER to the correlation networks of two high-throughput genetic interaction data sets, demonstrating CLOVER's

ability to identify functionally cohesive modules while identifying genes working at the interface of these functions.

3.1 Genetic Interaction Data Sets

Two genetic interaction data sets for *S. cerevisiae* were used in this study. The more recent and larger SGA (*Synthetic Genetic Array*) data set is obtained from [1] and comprises more than 5.4 million interactions involving 4443 genes. The raw genetic interactions from that work are modeled after the Fisher definition of epistasis as quantitative deviations from the expected multiplicative combination of independently functioning genes [14], which defines genetic interaction strength as follows:

$$\epsilon = f_{ab} - f_a f_b$$

f_a and f_b denote the single mutant fitness of gene a and gene b respectively, and f_{ab} is the fitness of the double-knockout mutant of genes a and b. The Collins *et al.* EMAP (*Epistatic Mini-Array Profile)* data set [9] is an earlier yeast double-knockout array, including 743 genes. While there is some overlap between the genes/interactions in these studies, they are independent and use different statistical techniques to compute interaction scores.

Both data sets consist of a matrix of scores, with each row corresponding to a query gene, and each column to one single-knockout library gene with which each query was crossed. A positive score indicates greater fitness in the resulting double mutant than expected according to a model based on single-knockout fitness measures for each of the two genes individually; such interactions are denoted *alleviating*. A negative score indicates lower fitness and is denoted *aggravating* [1]. In both data sets some individual scores are missing from the matrix.

Because the individual interaction scores are noisy and incomplete, we use the *correlation network* of the original interaction data, following the lead of previous studies (e.g. [15][9]). Each edge of the network corresponds to the similarity in the interaction profile between the two genes, obtained from the interactions by computing the Pearson correlation of each pair of genes' interaction profiles across all other genes.

Positions in the profile where either interaction value was missing were ignored. To obtain a complete graph in which all edge weights fall in the interval $[0,1]$, negative correlations were truncated to zero values, followed by scaling the remaining values linearly so that the highest correlation value is mapped to 1. We found this produced superior results to the two obvious alternatives: taking the absolute value of the correlation, and scaling all values into the range [0,1].

3.2 Evaluation on Genetic Interactions Networks

To validate the efficacy of CLOVER for the analysis of genetic interaction data, we clustered the correlation networks generated from quantitative interaction data from [9] and [1] respectively, as described above. We also obtained clusters using MCL for the same networks, over a range of the "inflation" parameter I which determines the granularity of the derived clusters. Additionally, a recent study [10] evaluated a number of other clustering techniques on the data from[9], and we compare our results to the best reported result achieved in that study by using genetic interactions alone.

3.3 Evaluation of Clusters

To evaluate the fitness of clusters identified using CLOVER, we tested the enrichment of clusters for known groups of functionally related genes. For this purpose we used all biological process annotations from the Gene Ontology (GO[16]). We considered a cluster to be "matched" to an annotation or complex if its enrichment was significant (P-value≤ 0.05) according to a Bonferroni-corrected hypergeometric test, following similar procedures in recent studies[10][6].

The number of matches based on this criterion, divided by the total number of clusters, gives a *precision* value P between 0 and 1 for each clustering compared to each benchmark. The number of matched benchmark groups, divided by the total number of groups, gives a *recall* value R. These values are combined into the harmonic mean *F-measure* defined as $F = P \times R/(P + R)$.

Table 1. Cluster Enrichment Summary for E-Map Network vs GO Annotations

Method	Precision (P)	Recall (R)	F-measure (P*R)/(P+R)	#clusters	#matches	#overlapping match pairs	#nodes in two clusters
Ulitsky et al	0.58	0.31	0.41	48	28	N/A	N/A
MCL $I = 4$	1.0	0.20	0.33	9	9	N/A	N/A
CLOVER $\alpha = 0.45$	0.72	0.32	0.44	53	38	Total: 12 Matched one-sided: 2 two-sided: 6	138

Table 2. Cluster Enrichment Summary for SGA Network vs GO Annotations

Method	Precision (P)	Recall (R)	F-measure (P*R)/(P+R)	#clusters	#matches	#overlapping match pairs	#nodes in two clusters
MCL $I = 4$	0.9	0.35	0.51	42	38	N/A	N/A
CLOVER $\alpha = 0.65$	0.69	0.4	0.51	189	131	Total: 40 Matched one-sided: 3 two-sided: 8	138

We compared the performance of CLOVER to MCL, a leading general-purpose clustering algorithm, on correlation networks derived from the Collins et al. EMAP and Costanzo et al. SGA data sets. We chose MCL because of its favourable performance relative to other methods for clustering of the EMAP data set[10]. For both algorithms we obtained the clusterings over a wide range of cluster granularity parameters (I for MCL and α for CLOVER), and present the best of these results. The performance of CLOVER over the full range of α parameters is described in Results. For the Collins EMAP dataset we also compare our results with the clustering obtained by the "Correlation" method of Ulitsky *et al.*[10], provided by the authors. Details of the best clustering obtained with each method, as determined by the corresponding F-measure, are presented in Tables 1 and 2. These results show that the CLOVER clustering captures co-annotated genes as well as MCL on the SGA correlation network, and better than both MCL and the "Correlation" method on the EMAP correlation network, with a significantly larger number of clusters in each case. Tables 1 and 2 additionally list the number of *matched overlaps* in each CLOVER clustering, as defined in the following section. The number of genes assigned to two clusters is also given; note that at the given parameter setting no nodes were assigned to more than two clusters.

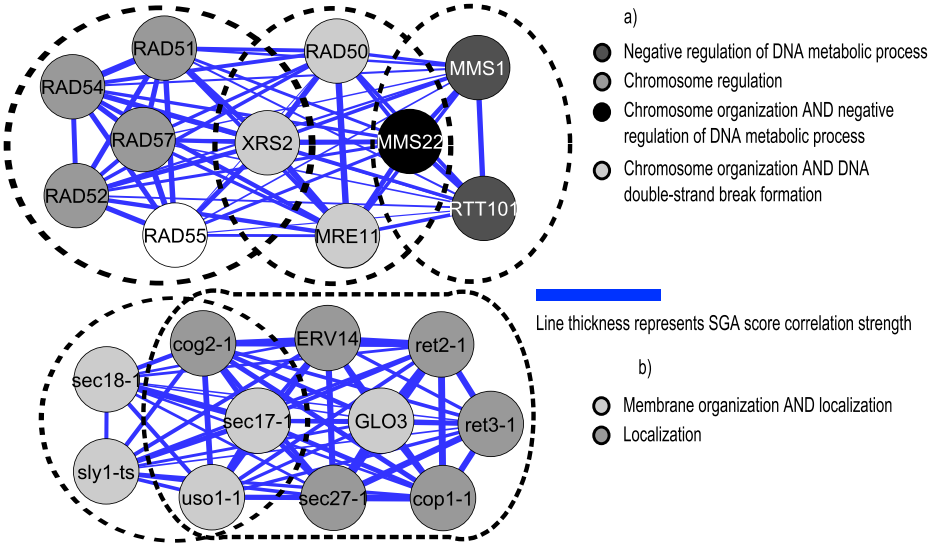

Fig. 4. Examples of double-sided matching pairs of clusters found by GO annotation mining. a) The genes *XRS2* and *MMS22* both participate in two of the three clusters shown. b) An example of three genes participating in two clusters.

3.4 Evaluation of Overlaps

We further assessed pairs of overlapping clusters with respect to the benchmarks according to the following criteria. A *two-sided matched overlap* is a pair of clusters C, D such that

- Both C and D are enriched for *distinct* biological processes (e.g. they are both "true positives")
- The overlapping region between C and D (i.e. the set of genes assigned to both clusters) contains genes which are annotated for both processes

Similarly, a *one-sided matched overlap* is a pair of clusters C, D such that

- Both C and D are enriched for *distinct* biological processes
- *The overlapping region between* C and D (i.e. the set of genes assigned to both clusters) contains genes which are annotated for at least one of these processes

In addition to these categories we also counted the number of simple overlapping matches – pairs of clusters, each matching some biological process, which overlap. The number of matched overlaps of each kind is included in tables 1 & 2.

In the remainder of this section, all specific examples of overlapping clusters are taken from the clustering of the SGA correlation network [1], using the parameter $\alpha = 0.65$. Figure 4(a) illustrates two two-sided overlaps. Each of the three clusters is

enriched for a distinct subset of the GO biological process categories "negative regulation of DNA metabolic process", "chromosome organization", and "DNA double-strand break formation". Both of the genes *XRS2* and *MMS2* are assigned to multiple clusters, and share the annotations for which those clusters are enriched. Figure 4(b) illustrates two clusters which share 3 overlapped genes. While all genes in both clusters are annotated with the broad annotation "localization", the cluster depicted at left is also enriched for "membrane organization". Two of the three genes shared by both clusters, *sec17-1* and *uso1-1*, are annotated with both biological process categories.

While this validation of overlaps participating in enriched biological functions is encouraging, the definitions of GO biological process annotations are coarse-grained, and cannot fully capture the fine distinctions evidenced by genetic interaction profiles and their modularity. Some genes are clearly and strongly connected to multiple clusters, and this can be validated by detailed analysis of the individual genes in each cluster. Figure 6 illustrates the intersection of two such clusters, with the *CTF4* gene assigned to both. One cluster includes genes involved in sister chromatid cohesion, while the other contains genes involved in DNA replication. In particular, the proteins encoded by *CTF18*,*CTF8*, and *DCC1* form a part of the replication factor C-like complex required for sister chromatid cohesion [18][19], while *MRC1* & *CSM3* gene products have been demonstrated present at replication forks [20][21]. POL32[22][23], *RAD27*[24], and *ELG1* [25][26][27] are all involved in lagging strand DNA synthesis. The *CTF4* gene product has also been found to act at replication forks[28][29][30][31], and in sister chromatin cohesion [19][32][18].

4 Future Improvements

Local search provides a great deal of control and flexibility due to the use of an explicit objective function. Recent work on special models of modularity in genetic interaction networks (e.g. [33][17]) suggest that more complex and domain-specific scoring schemes can be used to model specific types of modularity; we are interested in incorporating these models, possibly in combination, explicitly into our algorithm.

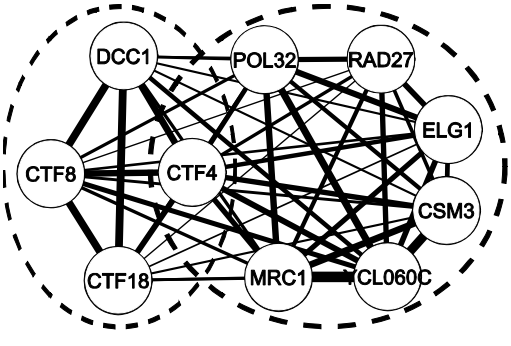

Fig. 5. Clusters overlapping at *CTF4*. The cluster depicted at left contains genes involved in sister chromatid cohesion. The cluster depicted at right contains genes involved in DNA replication.

This flexibility comes at a cost in computational complexity; currently our implementation takes roughly 6 hours to produce the reported results on our Dell Xeon-based servers, compared to e.g. under 20 minutes for MCL which does not perform explicit optimization. A common approach to this kind of intractability is to use a faster, unspecialized (and possibly non-overlapping) clustering method as a preprocessing step to reach a clustering which is "nearer" to optimal than a random or trivial starting point. Speeding up the algorithm in this way will allow faster exploration of different parameters and objective function variations.

References

1. Costanzo, M., Baryshnikova, A., Bellay, J., Kim, Y., Spear, E., Sevier, C., Ding, H., Koh, J., Toufighi, K., Mostafavi, S., Prinz, J., Onge, R., VanderSluis, B., Alizadeh, S., Bahr, S., Brost, R., Chen, Y., Cokol, M., Deshpande, R., Li, Z., Li, Z.-Y., Lian, W., Marback, M., Paw, J., San Luis, B.-J., Shuteriqi, E., Tong, A., van Dyk, N., Wallace, I., Whitney, J., Weirauch, M., Zhong, G., Zhu, H., Houry, W., Brudno, M., Ragibizadeh, S., Papp, B., Roth, F., Giaever, G., Nislow, C., Troyanskaya, O., Bussey, H., Bader, G., Gingras, A., Morris, Q., Kim, P., Kaiser, C., Myers, C., Andrews, B., Boone, C.: The Genetic Landscape of a Cell. Science 327(5964), 425–431 (2010)
2. Bader, G., Hogue, C.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4(2) (January 2003)
3. Hartuv, E., Shamir, R.: A Clustering Algorithm based on Graph Connectivity. Information Processing Letters 76, 175–181 (2000)
4. King, A., Przulj, N., Jurisica, I.: Protein complex prediction via cost-based clustering. Bioinformatics 20(17), 3013–3020 (2004)
5. van Dongen, S.: A Clustering Algorithm for Graphs. Technical Report 1386- 3681, Centrum voor Wiskunde en Informatica, Amsterdam (2000)
6. Navlakha, S., Schatz, M., Kingsford, C.: Revealing Biological Modules via Graph Summarization. Journal of Computational Biology 16(2), 253–264 (2009)
7. Wang, H., Kakaradob, B., Karotki, L., Fiedler, D., Shales, M., Shokat, K., Walther, T., Krogan, N., Koller, D.: A Complex-Based Reconstruction of the S. cerevisiae Interactome. Molecular and Cellular Proteomics 8(6), 1361–1381 (2009)
8. Tong, A.: Global Mapping of the Yeast Genetic Interaction Network. Science 303(5659), 808–813 (2004)
9. Collins, S., Miller, K., Maas, N., Roguev, A., Fillingham, J., Chu, C., Schuldiner, M., Gebbia, M., Recht, J., Shales, M., Ding, H., Xu, H., Cheng, B., Andrews, B., IngVarsdottir, K., Han, J., Boone, C., Berger, S., Hieter, P., Zhang, Z., Emili, A., Alic, C., Toczyski, D.P., Weissmann, J.: Functional dissection of protein complexes involved in yeast chromosome biology using a genetic interaction map. Nature 446, 806–810 (2007)
10. Ulitsky, I., Shlomi, T., Kupiec, M., Shamir, R.: From E-MAPs to module maps: dissecting quantitative genetic interactions using physical interactions. Molecular Systems Biology 4(209) (May 2008)
11. Palla, G.D.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)
12. Kelley, R., Ideker, T.: Systematic interpretation of genetic interactions using protein networks. Nature Biotechnology 23(5), 561–566 (2005)

13. Sharan, R., Ideker, T., Kelley, R., Shamir, R., Karp, R.: Identification of Protein Complexes by Comparative Analysis of Yeast and Bacterial Protein Interaction Data. J. Comp. Bio. 12(6), 835–836 (2005)
14. Fisher, R.: The correlations between relatives on the supposition of Mendelian inheritence. Trans. R. Soc. 52, 399–433 (1918)
15. Schuldiner, M., Collins, S., Thompson, N., Denic, V., Bhamidipati, A., Punna, T., Ihmels, J., Andrews, B., Boone, C., Greenblatt, J., Weissman, J., Krogan, N.: Exploration of the function and organization of the yeast early secretory pathway through an epistatic miniarray profile. Cell 123, 507–519 (2005)
16. Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S.: Gene ontology: a tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
17. Mayer, M., Gygi, S., Aebersold, R., Hieter, P.: Identification of RFC (Ctf18p, Ctf8p, Dcc1p): an alternative RFC complex required for sister chromatid cohesion in S. cerevisiae. Mol. Cell (May 2001)
18. Hanna, J., Kroll, E.S.: Saccharomyces cerevisiae CTF18 and CTF4 are required for sister chromatid cohesion. Molecular Cell Biology (May 2001)
19. Bando, M., Katou, Y., Komata, M., Tanaka, H., Itoh, T., Sutani, T., Shirahige, K.: Csm3, Tof1, and Mrc1 form a heterotrimeric mediator complex that associates with DNA replication forks. J. Biol. Chem. (October 2009)
20. Szyjka, S., Viggiani, C., Aparicio, O.: Mrc1 is required for normal progression of replication forks throughout chromatin in S. cerevisiae. Mol. Cell 19(5), 691–697 (2005)
21. Johansson, E., Majka, J., Burgers, P.: Structure of DNA polymerase delta from Saccharomyces cerevisiae. J Biol. Chem. 276(47), 43824–43828 (2001)
22. Stith, C., Sterling, J., Resnick, M., Gordenin, D., Burgers, P.: Flexibility of eukaryotic Okazaki fragment maturation through regulated strand displacement synthesis. J.Biol. Chem. 283(49), 34129–34140 (2008)
23. Liu, Y., Kao, H., Bambara, R.: Flap endonuclease 1: a central component of DNA metabolism. Annu. Rev. Biochem. 73, 589–615 (2004)
24. Bellaoui, M., Chang, M., Ou, J., Xu, H., Boone, C., Brown, G.: Elg1 forms an alternative RFC complex important for DNA replication and genome integrity. EMBO J 22(16), 4304–4313 (2003)
25. Kanellis, P., Agyei, R., Durocher, D.: Elg1 forms an alternative PCNAinteracting RFC complex required to maintain genome stability. Curr. Biol. 13(18), 1583–1595 (2003)
26. Ben-Aroya, S., Koren, A., Liefshitz, B., Steinlauf, R., Kupiec, M.: ELG1, a yeast gene required for genome stability, forms a complex related to replication factor C. Proc. Natl. Acad. Sci. 100(17), 9906–9911 (2003)
27. Lengronne, A., McIntyre, J., Katou, Y., Kanoh, Y., Hopfner, K., Shirahige, K., Uhlmann, F.: Establishment of sister chromatid cohesion at the S. cerevisiae replication fork. Mol. Cell (September 2006)
28. Gambus, A., van Deursen, F., Polychronopoulos, D., Foltman, M., Jones, R., Edmondson, R., Calzada, A., Labib, K.: A key role for Ctf4 in coupling the MCM2-7 helicase to DNA polymerase alpha within the eukaryotic replisome. EMBO J 28(19), 2992–3004 (2009)
29. Tanaka, H., Katou, Y., Yagura, M., Saitoh, K., Itoh, T., Araki, H., Bando, M., Shirahige, K.: Ctf4 coordinates the progression of helicase and DNA polymerase alph. Genes Cells 14(7), 807–820 (2009)
30. Gambus, A., Jones, R., Sanchez-Diaz, A., Kanemaki, M., van Deursen, F., Edmondson, R., Labib, K.: GINS maintains association of Cdc45 with MCM in replisome progression complexes at eukaryotic DNA replication forks. Nat. Cell Biol. 8(4), 358–366 (2006)

31. Warren, C., Eckley, D., Lee, M., Hanna, J., Hughes, A., Peyser, B., Jie, C., Irizarry, R., Spencer, F.: S-phase checkpoint genes safeguard high-fidelity sister chromatid cohesion. Mol. Biol. Cell 15(4), 1724–1735 (2004)
32. Segre, D., DeLuna, A., Church, G.M., Kishony, R.: Modular Epistasis in Yeast Metabolism. Nature Genetics 37(1), 77–83 (2005)
33. Ulitsky, I., Shamir, R.: Pathway redundancy and protein essentiality revealed in the Saccharomyces cerevisiae interaction networks. Mol. Syst. Biol. 3(104) (April 2007)
34. Brohee, S., van Helden, J.: Evaluation of clustering algorithms for proteinprotein interaction networks. BMC Bioinformatics 7(488) (November 2006)
35. Mewes, H., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schüller, C., Stocker, S., Weil, B.: MIPS: a database for genomes and protein sequences. Nucleic Acids Res. 28(1), 37–40 (2000)