Discovering Protein Clusters

Susan L. Epstein,[1, 2] Xingjian Li,[3] Peter Valdez,[2] Sofia Grayevsky,[2] Eric Osisek,[1] Xi Yun,[1] and Lei Xie[1, 2]

Department of Computer Science
[1]The Graduate Center and [2]Hunter College of The City University of New York
New York, NY 10065 USA

[3]Microsoft Online Services Division
Redmond, WA 98052 USA

susan.epstein@hunter.cuny.edu, Xingjian.Li@microsoft.com, lei.xie, pvald, sgrayevs@hunter.cuny.edu; eosisek, xyun@gc.cuny.edu

Abstract

As biological data about genes and their interactions proliferates, scientists have the opportunity to identify sets of proteins whose interactions make them worthy of further investigation. This paper reports on a knowledge discovery technique to support that work. Foretell is an algorithm originally designed to support search for solutions to constraint satisfaction problems. Recent adaptations enable Foretell to detect sets of genes that interact heavily with one another. We provide empirical results, and describe ongoing work on biological meaning and knowledge infusion from the user.

Introduction

This paper describes the integration of a local search metaheuristic with ongoing research in bioinformatics. The thesis of our work is that local search within a graph weighted by biological data can detect sets of closely interacting genes responsible for observable biological characteristics (e.g., cancer). The principal result reported here is that *Foretell*, an algorithm originally designed to support constraint solvers, hypothesizes meaningful relationships among sets of genes, relationships that are worthy of further biological investigation.

Systems biology seeks quantitative explanations for the simple principles that give rise to complex behaviors in biological systems. Although it is difficult to formulate mathematical equations to describe a biological system, boundary conditions are relatively easier to identify, state, and use. These *constraints* include evolutionary, chemical balancing, thermodynamic, capacity, temporal, and spatial restrictions. As a result, constraint-based modeling is an emerging paradigm for biological system models, particularly metabolic networks (Ruppin et al., 2010). Nonetheless, the implementation of such models remains challenging, and little work has thus far applied a rigorous, practical, constraint-based solution to model biological systems.

Omics data includes such knowledge bases as *genomes* (the hereditary information of an organism), proteomes (sets of expressed proteins), and metabolomes (small molecules that are intermediate products of metabolism). *Interactome* data includes experimentally determined and computationally predicted interactions. Such an interaction could be between a pair of proteins, between a protein and DNA, or between a protein and a ligand (small binding molecule). *Gene annotations* in databases such as FlyBase (Quilton et al., 2012) describe, to the best of current knowledge, the location of a gene on a chromosome and the role it plays in protein production and other cellular functions. The integration of gene annotations with interactome and omics data can provide important insights into noisy and incomplete biological data (Subramanian et al., 2005; Zhong et al., 2010).

The effects of genetic variants and molecular interactions that contribute to a trait are functionally channeled by the cell's regulatory or communication machinery. Thus, a *genetic event* (i.e., gene regulation and expression) implicated in a trait should address sets of closely interacting genes. One representation in which to search for such a set of genes is a context-specific protein-protein interaction (*PPI*) network, a weighted graph that represents known interactions between pairs of proteins in a genome (Califano et al., 2010).

To address the open problem of detecting strong relationships among sets of functional states, our innovation is to use variants of the algorithm used by constraint solvers to speed search for feasible solutions to a problem. Such search is often expedited by detection and prioritization of heavily-constrained subgraphs in the graph. Foretell was designed to detect such subgraphs in graphs associated

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

with constraint satisfaction problems (Li and Epstein, 2010).

The version of Foretell reported on here accepts a weighted graph that represents a PPI network for a particular organism as input, and searches within that graph for the genetic events it harbors, here called *clusters*. For Foretell, a cluster is a subgraph dense with high-weight edges. Although cluster detection is NP-complete, Foretell has scaled well on other real-world problems (Li and Epstein, 2010). Here we report for the first time on Foretell's adaptation to support the discovery of molecular and cellular mechanisms of fundamental biological processes.

In preliminary testing on the *drosophila melanogaster* (fruit fly) genome, Foretell quickly finds clusters comparable to those found by a state-of-the-art algorithm specifically designed for cluster detection in PPI networks. The next section of this paper provides relevant background on constraint satisfaction and PPI networks. Subsequent sections describe Foretell, the adaptations necessary to apply it to PPIs, empirical results, and a blueprint for ongoing work.

Background and related work

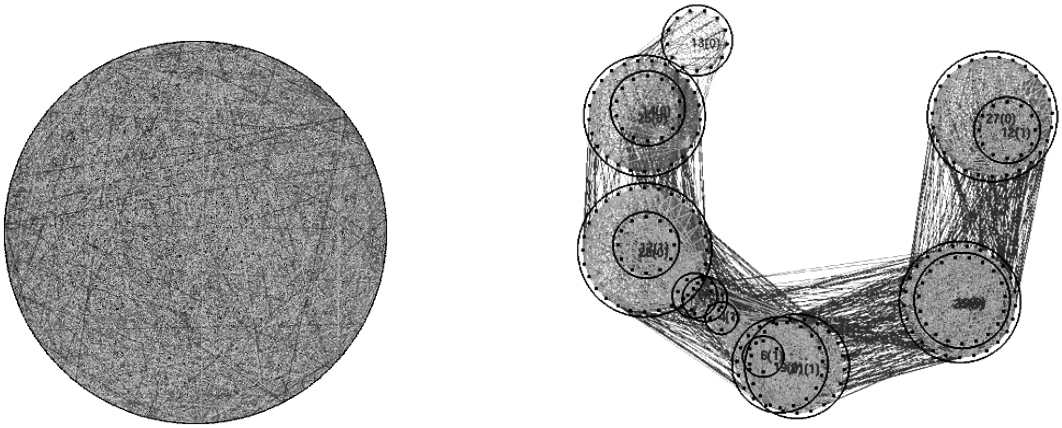
Constraint satisfaction

Intuitively, a constraint satisfaction problem (*CSP*) requires a set of objects to satisfy a set of constraints. In particular, a *binary* CSP $P = \langle X, D, C \rangle$ is a set X of variables, each with an associated discrete domain in D, and a set C of constraints that describes how pairs of variables may be assigned values from their respective domains simultaneously. For example, the n-queens puzzle seeks to place n chess queens on an $n \times n$ chessboard so that no two share the same row, column, or diagonal. Analogously, *protein-protein docking* seeks complementary interfaces between two proteins, A and B. It can be cast as an n-queens problem where the interface residues on protein A are viewed as the squares on the chessboard and the interface residues on B as the queens, with a set of energetic, evolutionary, and geometric constraints.

A *solution* to P assigns to each variable some value from its domain such that all of the constraints in C are satisfied. P can be represented by a *constraint graph* (as in Figure 1(a)), where each node represents a variable, and each edge between two nodes represents a constraint between them. How hard it is to find a solution to P depends both on the topological structure of P's constraint graph and on the interactions among its constraints.

Many modern constraint solvers do *systematic backtracking*, which sequentially assigns values to variables and validates the consistency of each assignment. *Propagation* removes, from the domains of as-yet-unassigned variables, some values that are inconsistent with constraints in C under the current assignments. When a variable's domain becomes empty under propagation (a *wipeout*), the solver backtracks. A *constraint weight* counts how often propagation along a constraint has led directly to a wipeout (Boussemart et al., 2004). Because many CSPs are intractable, solvers rely on search heuristics, many of which reference constraint weights.

Graph structure is important for propagation and decision making during search. Related work references elaborate structural graph features that might facilitate search for a solution to a CSP (e.g., (Cohen and Green, 2006; Dechter, 1990; Gottlob et al., 2000; Gyssens et al., 1994;



(a) (b)

Figure 1: Two visualizations of a package-routing constraint satisfaction problem. (a) All 650 nodes are plotted along the circumference of the circle, with 17,447 edges connecting them. Darker edges indicate more restrictive constraints. (b) Foretell finds (circled) clusters: dense, highly-constrained, disjoint subgraphs that include only part of the original problem. Although they are less restrictive than the intra-cluster edges, the density of the inter-cluster edges makes them appear more restrictive here.

Pearson and Jeavons, 1997; Samer and Szeider, 2006; Weigel and Faltings, 1999)). Many of these approaches are primarily theoretical; others prove powerful on some individual problems, but incur considerable computational overhead. All ignore the *tightness* of a constraint, the fraction of possible value pairs that it excludes. In contrast, Foretell considers tightness when it detects structure.

An important principle in systematic backtracking is to *fail first*, that is, to assign values first to those variables that will be most difficult to satisfy (Haralick and Elliott, 1980). Because they are dense with tight edges, the clusters that Foretell detects should more often experience wipeouts. Thus clusters are subgraphs where a solver should fail first as well. When a solver does so, it substantially reduces its total search time, in some cases by an order of magnitude (Epstein and Wallace, 2006). Extensive study has made clear, however, that the weights Foretell relies on are crucial to its success, and that search does best when the graph's constraint weights correctly anticipate difficulties later encountered during search (Li, 2011).

The link to biological systems

Constraint satisfaction is a feasible, novel, and elegant way to characterize allowable genetic, molecular, and cellular states under evolutionary, physical, genetic and environmental constraints (Tsang, 1993). In the constraint graph for such a CSP, the nodes are allowable functional states (e.g., three-dimensional coordinates of a protein structure, or a dysregulated gene regulatory network), the constraints are edges, and the weight on an edge represents how restrictive its associated constraint is. Once cast as a CSP, a biological problem can be explored effectively by incorporating constraints available from experiments and domain knowledge. Moreover, it is possible to specify biological knowledge for search and for optimization separately, to support the extraction of biologically-meaningful results in a large search space.

PPI networks, where each node represents a gene, have been widely used to extract closely-interacting genes (Gao et al., 2009). Thus far, however, graph-clustering algorithms for this purpose have been based purely on the connectivity (unweighted or weighted edges) of the PPI network, and therefore may not be biologically relevant. A *biologically meaningful* cluster is a set of genes that share similar pathways or functions, or physically interact with one another, much the way variables impact one another in CSP search. More meaningful clusters can be identified if a PPI incorporates other omics data (Chuang et al., 2007). Many algorithms have been developed to detect functional modules and integrate heterogeneous omics data (Califano et al., 2010). Nonetheless, three hurdles remain for effective, robust detection of functional modules that integrate such data. First, genetic regulation is a multivariable process; few existing methods consider coordinated regulation by a combination of genes. Second, the inherent noisiness and incompleteness of biological data seriously impact the performance of most algorithms (Yosef et al., 2009). Finally, biologists work from *phenotypic* data, descriptions of an organism determined both by its genome and by its environment. Phenotypic data is heterogeneous, context-specific, stochastic, and may vary significantly from one case of the same trait to another (Schadt, 2009).

Three features of Foretell make it particularly suitable for biological data analysis: its robustness, its tolerance for missing data, and its ability to incorporate domain-specific knowledge. Foretell can address noisy, incomplete, dynamic biological data, where not all existing constraints are known, and constraint tightness may be somewhat inaccurate. Moreover, Foretell finds not only *cliques* (complete subgraphs where every pair of nodes is connected by an edge) as conventional cluster-detection algorithms do (Wang et al., 2010), but also finds *near-cliques* (subgraphs a few edges shy of a clique). Thus Foretell can contend with the dynamic and stochastic nature of biological systems where constraints and tightness vary over time, space, and environmental conditions. Finally, it is straightforward to integrate heterogeneous biological data into Foretell, through edge weights, node weights, and its scoring and preference functions. Other cluster-detection algorithms are more difficult to modify with biological knowledge; they cannot accept both node and edge weights (Clauset et al., 2004; Newman, 2004).

Foretell

Foretell was originally embedded in a CSP solver, where it sped search for solutions by the detection of clusters such as those in Figure 1(b) (Epstein et al., 2005). In CSPs, Foretell quickly detected sets of interacting nodes in large constraint graphs, and its search metaheuristic (described below) made it robust to noise. We therefore have recently constructed a new, standalone, modular version of Foretell, one appropriate for research on large-scale PPI networks.

Foretell is based on Variable Neighborhood Search (*VNS*), a local search metaheuristic that explores increasingly large search subspaces called *neighborhoods* (Hansen and Mladenovic, 2003; Hansen et al., 2004). (Note that a VNS neighborhood is different from the neighborhood of a node in a graph.) VNS has been successfully applied to many problems in optimization and search. In the following discussion, the *size* of a cluster is the number of nodes it contains, and the *pressure* on a node is the sum of the weights on the edges on which it is incident.

Figure 2 provides high-level pseudocode for the new version of Foretell, which accepts graphs in GML (Graph Modeling Language), a standard graph file format. For

```
Foretell(graph, control)
while terminate(graph, cluster, control) is false
  cluster ← VNS(graph, control)
  output cluster
  remove cluster from graph

VNS(graph, control)
cluster ← seed(graph)
best-cluster ← SVT(graph, cluster)
best-score ← score(best-cluster)
local-optimum ← 0
shake-out ← 1
while terminate(graph, cluster, control) is false
          AND shake-out ≤ neighborhoods
  candidate ← copy(best-cluster)
  if shake-out ≠ 1
      candidate ← shake(candidate, shake-out)
  candidate ← VND(graph, control, candidate)
  local-optimum ← score(candidate)
  if local-optimum > best-score
      best-cluster ← candidate
      best-score ← local-optimum
      shake-out ← 1
      if size(candidate) ≥ neighborhoods
        neighborhoods ← ceiling(0.1 * size(candidate))
  else
      shake-out ← shake-out + 1
return (best-cluster)

VND(graph, control, candidate)
while terminate(graph, cluster ,control) is false
  candidate ← SVT(graph, candidate)
  best-score ← score(candidate)
  while terminate(graph, cluster, control) is false AND
          prefer(graph, control, possible) ≠ ∅
      possible ← copy(candidate)
      new-node ← prefer(graph, control, possible)
      possible ← include(possible, new-node)
      possible ← SVT(graph, possible)
      if score(possible) > best-score
        candidate ← possible
        best-score ← score(possible)
  interchanges ← likely(graph, control, candidate)
  if interchanges≠ ∅
      swap ← select(interchanges)
      possible ← copy(candidate)
      possible ←apply(swap, possible)
      if score(possible) > best-score
        best-score ← score(possible)
        candidate ← possible
return (candidate)
```

Figure 2: Pseudocode for cluster discovery.

simplicity, we describe the input here as a graph and *control*. The latter is a set of parameters, including designations of the supporting functions *seed, prefer, score,* and *terminate,* and values for such constants as *neighborhoods,* the maximum number of local search subspaces. During a single *run*, Foretell identifies one cluster at a time, and removes it from the graph before it searches for another. Thus, its clusters are disjoint.

Foretell calls VNS to detect a single cluster. A new cluster begins from a node selected by the *seed* function. VNS then applies *SVT* (Simplicial Vertex Test) to add to the cluster, one at a time, any node that has an edge to every node in the cluster and also has maximum pressure. VNS evaluates the worth of a cluster with the *score* function, records the top-scoring *best-cluster* and its *best-score*, and then iterates to try to improve upon *best-score*.

To combat the plateau effect common in local search and to explore different portions of the search space, each VNS iteration except the first *shakes* (i.e., removes) from the current *best-cluster* some number of nodes (*shake-out*). After shaking, VNS repeatedly calls *VND* (Variable Neighborhood Descent) to revise a copy of the current *best-cluster*. If the revision improves on *best-score*, VNS updates *best-cluster* and *best-score,* and resets *shake-out* to 1. While the size of the cluster is less than *neighborhoods, shake-out* counts how many consecutive times *shake* followed by VND has failed to produce a higher score. Although *terminate* requires *shake-out* to remain less than *neighborhoods*, VNS also revises *neighborhoods* so that larger clusters can continue to grow.

VND tries to improve the *best-score* of the *candidate* cluster it receives from VNS. To do so, VND applies SVT to *candidate*, and then does local search to try to improve *possible*, a copy of *candidate*. One at a time VND greedily selects a node chosen by its preference function *prefer*, adds the node, and reapplies SVT. When no further greedy addition is available, VND updates *candidate* and *best-score*, and assembles *interchanges*, actions that would replace a node in *candidate* with one or two nodes not in *candidate*. The *likely* function ensures that any interchange made to a cluster of n nodes will ensure a monotonically non-decreasing density. This is achieved if Foretell does not increase m (the number of missing edges that prevent that cluster from being a clique) by more than

$$\frac{n}{2}+\frac{m}{n-1} \qquad (1)$$

If a randomly-chosen *swap* from *interchanges* improves on *best-score*, VND revises *candidate* and *best-score,* and returns to SVT followed by its greedy loop.

Based on parameter values in *control*, the predicate *terminate* determines when to stop VND's local search, VNS's cluster search, and Foretell itself. During execution, graph nodes are partitioned into three sets: those in the cur-

rently developing cluster, those eligible to join it, and those excluded from it. Further details are available in (Li, 2011).

Adaptation for PPIs

Several changes to the original version of Foretell were required for PPIs. In a graph for a pairwise PPI network (henceforth, a *network*) each node represents a protein. An edge between two nodes indicates that the two proteins they represent are known to or predicted to interact with one another.

A fundamental difference between a CSP and a network is in the edge weights. For CSPs, Foretell did preliminary search to determine edge weights. In a network, each edge instead has an input *confidence level* based on the confidence score from STRING (Jensen et al., 2009). STRING is a database of known and predicted protein-protein associations, both physical and functional. These associations are derived from high-throughput experimental data, literature mining, and predictions based on genomic context analysis. To assign confidence scores to the associations, STRING benchmarks them against a common reference set. Foretell computes edge weights as the confidence level of the interaction the edge represents, normalized in [0, 1].

Other changes to Foretell for PPIs identified new supporting functions. Both *seed* and *prefer* are now user-specified through *control.* For the work reported here, the *seed* function selects a maximum degree node, and breaks ties first by pressure and then randomly. The *prefer* function maximizes node pressure*,* and breaks ties on maximum node degree. Finally, we developed and tested several new *score* functions adapted from the CSP formulation, and selected a simple and effective one. For a cluster with n nodes and total edge weight T:

$$score = \frac{2T}{n-1} \qquad (2)$$

Foretell also now supplies extensive data on the clusters it detects. This includes their contents (nodes and edges), *density* (percentage of possible edges that appear in the cluster), and average edge weight. Summary statistics across all clusters also provide data on *coverage*, the percentage of nodes and edges included in some cluster. This data is intended to support the user's understanding of the impact of her choices for parameters in the control file (e.g., *prefer* and *score*).

Table 1: PPI networks for Foretell.

Genome	nodes	edges	density
Yeast	4,757	97,617	0.86%
Fruit fly	11,408	759,580	1.17%
Human	13,757	1,013,580	1.07%

Experimental design and results

We have begun to explore three genomes with Foretell: *saccharomyces cerevisiae* (baker's yeast), *drosophila melanogaster* (fruit fly), and *homo sapiens* (human). Network details appear in Table 1. All three are considerably larger than the largest CSP tested with Foretell, which had 650 variables and 17,447 constraints. We report in detail here only on clusters detected by Foretell in the fruit-fly network.

The only *control* parameter explored here is *cutoff,* the maximum search time on an individual cluster. Parameters in *control* also include maximum total runtime to search for all clusters, minimum cluster size, and *neighborhoods*. There was no maximum total runtime enforced in these experiments, the minimum cluster size was set at 3, and *neighborhoods* was 10.

Cutoff, the time allotted under *terminate* to find a single cluster, impacted Foretell's consistency on CSPs (Li and Epstein, 2010) and proved to do so in this network as well. We tried a range of *cutoff* values from 1 to 15 minutes. One minute and two minutes produced unstable results, where different clusters and different numbers of clusters were produced from one run to the next. Values above eight minutes allowed Foretell to work harder on each cluster, but did not appreciably change the coverage. Identical or nearly identical sets of clusters were found for 10, 12, and 15 minutes, with results similar in size and number to those for 3 minutes. Here we analyze the 114 clusters found under a 3-minute cutoff in one run.

Foretell is selective; under any cutoff, it never clustered more than 15.45% of the nodes and 3.32% of the edges. Moreover, Foretell identifies dense clusters. Although the density of the full fruit fly network is 1.17%, many clusters are either cliques or near-cliques. Indeed, the average density of a Foretell-detected cluster in the fruit fly PPI was 91.14%. The clusters also contain high-weight edges; 23 clusters have average edge weight above 0.9 (on a scale, recall, from 0 to 1). Thus, in graphs based on known and predicted physical and functional protein-protein associations, Foretell found subsets of nodes that strongly interact with one another.

The salience of the clusters of variables found by Foretell in a constraint graph is demonstrated by the degree to which value assignment to those variables early in search improves performance — prescient clusters accelerate search. To evaluate the salience of a cluster of genes, however, we need metrics for the degree to which they are biologically meaningful. A gene ontology (*GO*) similarity score can be computed for three categories: biological process, cellular component, and molecular function. The GO similarity of a cluster measures the degree to which the genes in it describe known biological results. Figure 3 shows the GO similarity score in each category for the

clusters detected by Foretell in the fruit fly genome on a single run, using the Topological Clustering Semantic Similarity metric (Jain and Bader, 2010). It also shows the results for *SPICi*, a state-of-the-art clustering algorithm specifically designed for clustering in PPI networks (Jiang and Singh, 2010).

The results in Figure 3 overlap closely. Analysis indicates that SPICi finds somewhat larger clusters, but that Foretell's have higher average edge weights. Because biologically meaningful clusters are more likely to both be large and have high average edge weight, the results from Foretell and SPICi should together prove informative to biologists.

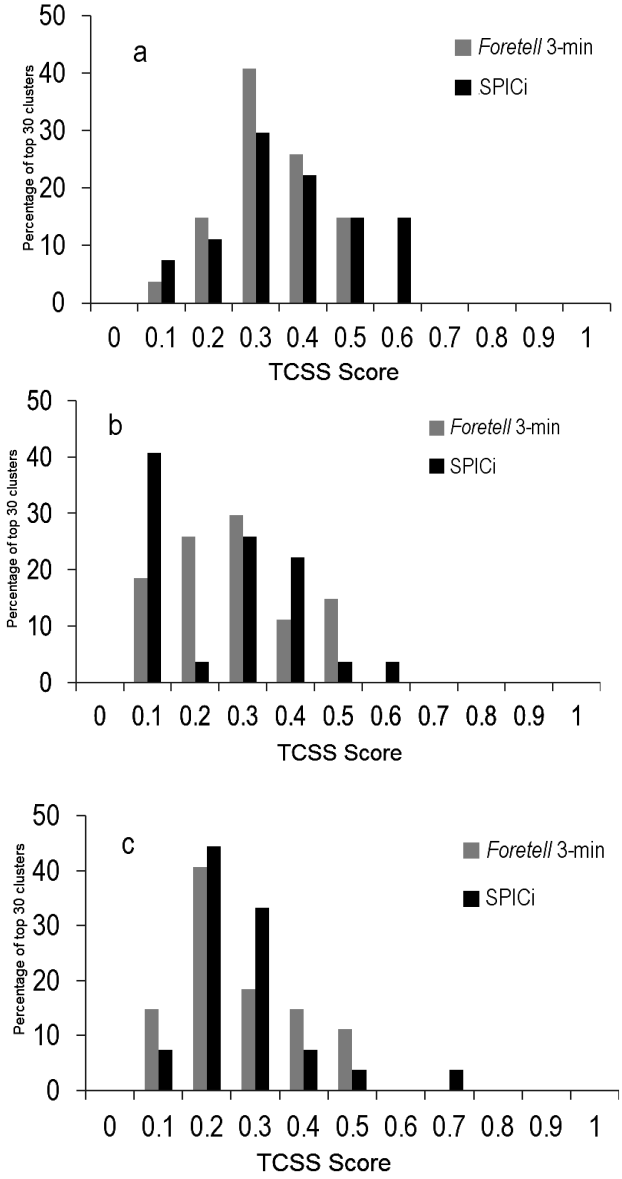


Figure 3: Distributions of cluster average gene ontology similarity in the fruit fly, as constructed by SPICi and by one Foretell run for (a) biological process, (b) cellular component, and (c) molecular function. The horizontal axis shows the distribution of similarity scores (higher is better); the vertical axis is the percentage of the top 30 clusters with that score.

Finally, Table 2 compares the 10 fruit-fly clusters with highest average edge weights from Foretell to the 10 from SPICi. The proteins in clusters top-ranked by Foretell are more functionally related than those from SPICi, as indicated by GO and biological pathway enrichment. Results on the clusters detected in the other genomes are currently under study.

Discussion

We emphasize that the results presented thus far are from a fully automated system, where we adjusted only a single parameter: the time to find one cluster. The selection mechanisms *seed, prefer,* and *score* that produced the results in the preceding section lack biological knowledge. They referenced only node degree and confidence level, a biological analog of search experience in a CSP. Here we simply cast a PPI network as a weighted graph, where nodes represent genes and edges represent relationships among them. Although Foretell is ready to accept biologically-meaningful versions of *seed*, *prefer, score,* and *terminate*, tests with such functions are only now underway.

The current version of Foretell is a more flexible tool for both skilled bioinformaticians and novice investigators. A significant change is that the human investigator can now exercise some control over what is valued during local search. Foretell provides a variety of options for the *seed, prefer, score,* and *terminate* functions; these options can be selected by name in the control file. Moreover, the investigator can specify a particular seed node (that cannot be shaken from the cluster), or a customized *seed* function that describes properties of the seed. For example, one could predefine a set of biologically-meaningful variables (e.g., disease-causing genes), and have Foretell use them as

Table 2: Enriched GOs and pathways in the top 10 clusters found by Foretell and by SPICi in a weighted PPI graph for the fruit fly. Clusters are ranked by average edge weight; those ranked 2, 3, 5, and 8 by both algorithms had no enriched GOs and pathways.

	Enriched GOs		**Enriched pathways**	
Cluster rank	Foretell	SPICi	Foretell	SPICi
1	39	0	1	0
4	3	0	0	0
6	16	0	0	0
7	0	0	1	0
9	32	0	2	0
10	79	0	0	0

seeds from which to build clusters of variables that are connected to them with high-weight edges. The investigator can also specify a *prefer* function that characterizes which nodes are of particular interest, and a *score* function that describes significant features of a cluster (e.g., minimum size, minimum or average edge weight).

Foretell now provides values likely to be employed in user-created versions of *prefer, score*, and *terminate,* including coverage, density, number of detected clusters, average cluster edge weight, and minimum degree. For example, a *seed* function could mandate some minimum degree, or a *score* function could enforce a minimum average edge weight and a minimum size. An investigator can also specify search time per cluster and per run, impose a limit on the number of nodes shaken out by VNS, and have *terminate* halt a Foretell run after it has found some number of clusters.

Foretell is non-deterministic. Because ties are broken at random, equal *prefer* values and equal *score* values may go unexplored. Foretell's results should therefore be analyzed over multiple runs for both accuracy and consistency. (Eventually a tie-breaking function will be another parameter, to facilitate further incorporation of biological data and knowledge.) Foretell is now embedded in an *experiment* loop, where each run identifies a full set of clusters. Metrics are currently under development to assess the consistency and statistical significance of sets of identified clusters from one run to the next.

Most existing PPI graph-clustering algorithms, including SPICi, cannot integrate multiple kinds of omics data directly; they use only edge weights to guide their search (Jiang and Singh, 2010). Foretell can use omics data as weights for nodes as well as for edges. A careful balance must be struck, however, between node weights and edge weights. Current work includes node weight computations based on gene expression profile data from heterogeneous samples using a signal-noise decomposition schema.

Our current investigations address heavily-interacting subsets of genes in the PPI network for yeast. The well-annotated biological pathway information for this organism should facilitate cluster evaluation. We have constructed several input graphs for yeast with different sets of weights, and have begun to search for and analyze clusters within them. We have begun to incorporate additional domain knowledge, such as differentially expressed genes, into the graph as node weights, and to address node weights in Foretell's computations.

Given Foretell's output, we expect researchers to generate novel, testable hypotheses that may lead to new biological discoveries. For example, the largest cluster Foretell found in a fruit fly run included 87 proteins highly involved in forming a protein complex (false discovery rate $p < 0.05$). The missing edges in these clusters represent unknown relationships worthy of investigation.

Conclusions

Foretell can infuse search with human knowledge, and has proved fast enough for graphs as large as the PPI for the human genome. Detected clusters that are not cliques are of particular interest — each missing edge hypothesizes a relationship between a pair of genes, and thereby motivates a biological study to determine if that pair also functionally or physically interacts. Moreover, Foretell's control mechanism makes it easy for relative novices in both computer science and biology to develop and test hypotheses about PPI networks.

Foretell could support human understanding of other weighted graphs as well, including the real-world CSPs for which it was originally developed. Since biological problems can be generally formulated as CSPs, Foretell can be applied to a broad range of challenging issues in biology beyond PPI analysis. These include protein-protein docking, prediction of functional sites (clusters of residues that perform a specific function), and the alignment of multiple sequences that represent DNA, RNA, or proteins. In particular, the Encode and modEncode projects seek to identify all the functional elements in the genome sequences of human and other model organisms, including the fruit fly (http://www.genome.gov/10005107). Recent remarkable advances there continue to generate vast amounts of data that represent a complex network among not only genes (as in the PPI networks here) but also non-gene fragments of DNA. Constraint modeling offers a powerful paradigm within which to construct and analyze the genetic circuits that control biological systems.

Meanwhile, Foretell integrates heterogeneous, noisy, and incomplete data into an effective and robust identification of functional modules: clusters. Its non-determinism provides a bulwark against noise. To the best of our knowledge, this is the first constraint-inspired approach to the challenging combinatorial problem of functional module detection in PPI networks.

Acknowledgements

This work was supported in part by the National Science Foundation under IIS-0811437, CNS-0958379, and CNS-0855217. Dr. Li's work was performed while at The Graduate Center of The City University of New York.

References

Boussemart, F., F. Hemery, C. Lecoutre and L. Sais 2004. Boosting systematic search by weighting constraints. In *Proceedings of ECAI-2004*, 146-149. IOS Press.

Califano, A., A. Butte, S. Friend, T. Ideker and E. E. Schadt 2010. Integrative Network-based Association Studies: Leveraging cell regulatory models in the post-GWAS era. *Nature Procedings* 713.

Chuang, H. Y., E. Lee, Y. T. Liu, D. Lee and T. Ideker 2007. Network-based classification of breast cancer metastasis. *Mol Syst Biol* 3: 140.

Clauset, A., M. E. J. Newman and C. Moore 2004. Finding community structure in very large networks. *Physical Review E* 70(6): 066111.

Cohen, D. A. and M. J. Green 2006. Typed Guarded Decompositions for Constraint Satisfaction. In *Proceedings of Principles and Practice of Constraint Programming -- CP2006*, 122-136. Nantes, Springer Verlag.

Dechter, R. 1990. Enhancement schemes for constraint processing: backjumping, learning and cutset decomposition. *Artificial Intelligence* 41: 273-312.

Epstein, S. L., E. C. Freuder and R. J. Wallace 2005. Learning to Support Constraint Programmers. *Computational Intelligence* 21(4): 337-371.

Epstein, S. L. and R. J. Wallace 2006. Finding Crucial Subproblems to Focus Global Search. In *Proceedings of ICTAI-2006*, 151-159. Washington, D.C., IEEE.

Gao, L., P. G. Sun and J. Song 2009. Clustering algorithms for detecting functional modules in protein interaction networks. *J Bioinform Comput Biol* 7: 217-242.

Gottlob, G., N. Leone and F. Scarcello 2000. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence* 124(2): 243-282.

Gyssens, M., P. G. Jeavons and D. A. Cohen 1994. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence* 66(1): 57-89.

Hansen, P. and N. Mladenovic 2003. Variable Neighborhood Search. *Handbook of Metaheuristics*. Glover, F. W. and G. A. Kochenberger. Berlin, Springer**:** 145-184.

Hansen, P., N. Mladenovic and D. Urosevic 2004. Variable neighborhood search for the maximum clique. *Discrete Applied Mathematics* 145: 117-125.

Haralick, R. M. and G. L. Elliott 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14: 263-314.

Jain, S. and G. D. Bader 2010. An improved method for scoring protein-protein interactions using semantic similarity within the gene ontology. *BMC Bioinformatics* 11(562).

Jensen, L. J., M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork and C. von Mering 2009. STRING 8--a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Res* 37D: 412-416.

Jiang, P. and M. Singh 2010. SPICi: A fast clustering algorithm for large biological networks. *Bioinformatics* 26(8): 1105-1111.

Li, X. 2011. Structure-based Search to Solve Constraint Satisfaction Problems. *Computer Science*. New York, The Graduate Center of The City University of New York. Ph.D. thesis.

Li, X. and S. L. Epstein 2010. Learning cluster-based structure to solve constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence.*

Newman, M. E. J. 2004. Analysis of weighted networks. *Physical Review E* 70(5): 056131.

Pearson, J. and P. G. Jeavons 1997. A Survey of Tractable Constraint Satisfaction Problems. London, Royal Holloway University of London.

Quilton, P., S. E. St. Pierre, J. Thurmond and F. Consortium 2012. FlyBase 101 – the basics of navigating FlyBase. *Nucleic Acids Res* 40: D706-14.

Ruppin, E., J. A. Papin, L. F. de Figueiredo and S. Schuster 2010. Metabolic reconstruction, constraint-based analysis and game theory to probe genome-scale metabolic networks. *Curr Opin Biotechnol* 21(4): 502-10.

Samer, M. and S. Szeider 2006. Constraint Satisfaction with Bounded Treewidth Revisited. In *Proceedings of Principles and Practice of Constraint Programming -- CP2006*, 499-513. Nantes, Springer Verlag.

Schadt, E. E. 2009. Molecular networks as sensors and drivers of common human diseases. *Nature* 461(7261): 218-23.

Subramanian, A., P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander and J. P. Mesirov 2005. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A* 102(43): 15545-50.

Tsang, E. P. K. 1993. *Foundations of Constraint Satisfaction*. London, Academic Press.

Wang, J., M. Li, Y. Deng and Y. Pan 2010. Recent advances in clustering methods for protein interaction networks. *BMC Genomics* 11 Suppl 3: S10.

Weigel, R. and B. Faltings 1999. Compiling Constraint Satisfaction Problems. *Artificial Intelligence* 115: 257-287.

Yosef, N., L. Ungar, E. Zalckvar, A. Kimchi, M. Kupiec, E. Ruppin and R. Sharan 2009. Toward accurate reconstruction of functional protein networks. *Mol Syst Biol* 5: 248.

Zhong, H., X. Yang, L. M. Kaplan, C. Molony and E. E. Schadt 2010. Integrating pathway analysis and genetics of gene expression for genome-wide association studies. *Am J Hum Genet* 86(4): 581-91.