A Hypergraph Model for the Yeast Protein Complex Network*

Emad Ramadan, Arijit Tarafdar, Alex Pothen
{eramadan, tarafdar, pothen}@cs.odu.edu
Computer Science Department
Old Dominion University
Norfolk, VA 23529 USA

Abstract

We consider a hypergraph model for the protein complex network obtained from a large-scale experimental study to characterize the proteome of the yeast. Our model views the yeast proteome as a hypergraph, with the proteins corresponding to vertices and the complexes corresponding to hyperedges. Previous work has modeled the protein complex data as a protein-protein interaction graph or as a complex intersection graph; both models lose information and require more space. Our results show that the yeast protein complex hyper-graph is a small-world and power-law hypergraph. We design an algorithm for computing the k-core of a hypergraph, and use it to identify the *core proteome*, the maximum core of the protein complex hypergraph. We show that the core proteome of the yeast is enriched in essential and homologous proteins. We implement greedy approximation algorithms for variant minimum weight vertex covers of a hypergraph; these algorithms can be used to improve the reliability and efficiency of the experimental method that identifies the protein complex network.

Keywords: protein complex network, hypergraph, small-world network, power-law network, graph core, vertex cover

1. Introduction

Proteins accomplish their tasks within a cell by forming multi-protein complexes, which are assemblies of groups of proteins. Recently a large-scale experimental study by Cellzome has attempted a systematic characterization of the multi-protein complexes in yeast [3]. This study is a part of a wider effort to characterize the *proteome*, all the proteins in an organism, with a view to understanding their sequence, structure, cellular localization, function, and role in cellular processes. (Identifying the proteome is a challenging task since alternate splicing and post-translational modifications potentially enable many proteins to be made from a gene. The proteome is also more dynamic than the genome, since the set of proteins made by a cell depends on the phase of the cell cycle and the environmental conditions.)

How should the data from large-scale protein complex studies be modeled in order to facilitate efficient algorithms to query the data? The data consists of the protein membership lists of the protein complexes. The authors of the Cellzome study and others have used graph models. However, graph models that have been proposed to date have represented the data either as protein-protein interaction graphs, or as complex intersection graphs, by projecting out either the complexes or the proteins. Unfortunately some of these graph models make assumptions about the structure of the protein complexes, for which there is insufficient data; other graph models lose some of the information in the data. In a protein-protein interaction graph, either all the proteins in a complex are considered to form a clique, or a distinguished protein (called the *bait protein* from the nature of the experiments) is considered to be connected to all the others. But both these assumptions are unphysical; in a large complex consisting of nearly hundred proteins, say, it would be sterically impossible for all the proteins to bind to each other, or for one protein to bind to all others. The other model used, the complex intersection graph, does not represent the proteins at all.

We propose a hypergraph model for the protein complex data. We characterize the properties of the yeast protein complex hypergraph from the Cellzome experimental study. We show that it is both a power-law (scale-free) network and a small world network. Then we discuss the concepts of the k-core of a graph and of a hypergraph, and describe an algorithm for computing the various cores. Using this algorithm, we compute the maximum core of the yeast protein complex hypergraph, and thereby characterize the *core proteome*. One of the problems with large-scale proteomic techniques when compared to smaller-scale methods is the relatively lower reliability of the experimental results.

*This research was partially supported by NSF grant CCR-0306334, by a Department of Energy grant DE-FC02-01ER25476, and by a grant B533529 from the Lawrence Livermore National Laboratory.

Hence we consider the problem of selecting a subset of the proteins to be bait proteins (that pull other proteins in complexes with them) in the Cellzome experiment. Minimum weight vertex covers (and multicovers) in hypergraphs are used to choose a set of candidate bait proteins that (1) are provably within a small factor of the optimum cover, and (2) identify the protein complexes with improved reliability.

1.1. The Cellzome experiment

In the Cellzome approach [3] for identifying multi-protein complexes, a set of *bait proteins* in the yeast is selected. The genes corresponding to the bait proteins are tagged with a tandem-affinity-purification (TAP) tag, and inserted into dividing yeast cells. In the yeast cells the tagged gene is expressed into tagged bait protein. The tagged bait protein and other proteins that are complexed with it are isolated from the lysed cells as described below. The TAP tag is used to purify a protein complex through two affinity column separations before the proteins in it are identified via mass spectrometry.

The TAP tag consists of three components. The first component is a protein that binds to immunoglobulin G beads in an affinity column, and the multi-protein complex is isolated from the rest of the cell lysate through this binding. The second component is a protease, which is used to remove the protein complex from the column by cleaving it from the immunoglobulin beads. The third component is a calmodulin binding peptide, which is used to purify the complex a second time by its binding to a second affinity column containing calmodulin beads. The complex is eluted from this column using a chemical that preferentially binds to the beads, and the proteins in the complex are identified through mass spectrometry.

The data from the TAP experiment thus consists of protein complexes identified from various bait proteins. Each complex is identified by the proteins that constitute its members.

1.2. Graph representations

Gavin et al [3] visualized the data from their experiments as a complex intersection graph. The vertices of this graph are the complexes, and an edge joins two complexes if they have one or more proteins in common. Edges in this network reflect common protein memberships in the complexes, and may represent common regulation, localization, turnover, or architecture. Each edge of the complex intersection graph could be weighted to represent the number of proteins two complexes have in common. Unfortunately, information about proteins is not represented in the complex intersection graph.

Another approach to representing the data from the TAP experiment is to use a protein-protein interaction graph, as has been used to represent binary protein interactions from experiments such as the yeast 2-hybrid system. In a binary protein interaction graph, the vertices correspond to proteins, and an edge joins two proteins that bind to each other. Note that the complexes are not represented in the protein interaction graph, and hence it is not powerful enough to represent both proteins and complexes.

There is a more serious difficulty with the use of protein interaction graphs to represent the protein complex data. Which proteins in a complex bind with each other? A range of answers to this problem is possible, and has been employed in graph representations.

All of the proteins in a complex could be considered to interact with each other, and then each complex is represented as a clique defined on the proteins in the complex. This approach leads to unusually high clustering coefficients in the protein interaction graphs [8]. Another viewpoint is to consider the bait protein as interacting with each other protein it pulls down in a complex, and thus each complex is represented as a 'star graph'. Both approaches are unphysical, as we have pointed out earlier.

Unless we know the set of binary interactions among the proteins in each complex, the approach of representing the data by protein-protein interaction graphs is bound to be inaccurate. It is also expensive in terms of storage, since a clique on n vertices can be represented in $O(n)$ space as a list of vertices, while representing each edge in the clique (as is done in protein interaction graphs) requires $O(n^2)$ space. A similar criticism is also true of complex intersection graphs, since an edge joins two complexes that share a common protein; a protein that belongs to m complexes generates $O(m^2)$ edges in the complex intersection graph, while the complexes can be represented in space proportional to the sum of the numbers of proteins in them.

1.3. Protein complex hypergraph

We propose a hypergraph model for the protein complex data. A hypergraph $H = (V, F)$ consists of a set of vertices V and a set of hyperedges F; each hyperedge is a subset of vertices. The difference between an edge in a graph and a hyperedge in a hypergraph is that the former is always a subset of two vertices (or a subset consisting of one vertex in the case of a loop), whereas in a hyperedge, the subset of vertices can be of arbitrary cardinality. In the protein complex data, we represent each protein by a vertex and each complex by a hyperedge.

The *degree* of a vertex is the number of hyperedges it belongs to, and the *degree* of a hyperedge is the number of vertices it contains. We denote the maximum degree of a vertex by Δ_V and the maximum degree of a hyperedge by Δ_F.

A *path* in the hypergraph consists of an alternating sequence of vertices and hyperedges v_1, f_1, v_2, f_2, ..., v_{i-1},

f_{i-1}, v_i, (where the v_j are vertices, and the f_j are hyperedges), with the following properties: each hyperedge f_j contains vertices to its left and right in the listing of the path, v_j and v_{j+1}; the path begins and ends with vertices, and no hyperedge or vertex can be repeated. The *length* of the path is the number of hyperedges in it. The *distance* between two vertices is the length of a shortest path that joins them. The *diameter* of a hypergraph is the maximum distance between any pair of vertices in it.

For drawing a hypergraph, it is helpful to represent both the vertices and the edges of the hypergraph as vertices in a bipartite graph. This graph $B(H) = (X, Y, E)$ has its vertex set partitioned into two sets X and Y, where X corresponds to the vertices of the hypergraph, and Y to the hyperedges of the hypergraph; an edge $(v, f) \in E$ joins a vertex $v \in X$ to vertex $f \in Y$ if the vertex v in the hypergraph belongs to the hyperedge f. For the protein complex data, the proteins correspond to one set of vertices, the complexes to the other, and an edge joins a protein to a complex if and only if the protein is a member of the complex.

Fig. 3 depicts a drawing of the yeast protein complex hypergraph from the Cellzome experiment as a bipartite graph. The figure is drawn using the Pajek software (URL: `vlado.fmf.uni-lj.si/pub/networks/pajek`).

2. Properties of the protein complex hypergraph

The protein complex hypergraph from the Cellzome experiments has 33 connected components, the largest of which consists of $1,263$ proteins (vertices) and 99 complexes (hyperedges). There are 846 proteins with degree 1 (belonging to only one complex), and the maximum degree of a protein is 21. This highest degree protein is ADH1, an alcohol dehydrogenase. The diameter of the yeast hypergraph is 6, and the average path length is 2.568, suggesting that the yeast protein complex hypergraph is a small world network.

The frequency of proteins with a given degree plotted against the degree shows that the yeast hypergraph satisfies a power law, $P(d) = cd^{-\lambda}$, where d is the protein degree, and $P(d)$ is the number of proteins with degree d. From a log-log plot, shown in Fig. 1, we estimate $\log(c) = 3.161$ and the exponent $\lambda = 2.528$; we assess the goodness of the linear fit to be excellent, since the value of $R^2 = 0.963$, where $R^2 = 1 - \frac{(r^T r)}{(y^T y)}$, r is the vector of residuals, and y is the dependent variable measured in deviations from the mean. Unlike proteins, the frequencies of complexes with a given degree do not satisfy a power-law or an exponential distribution.

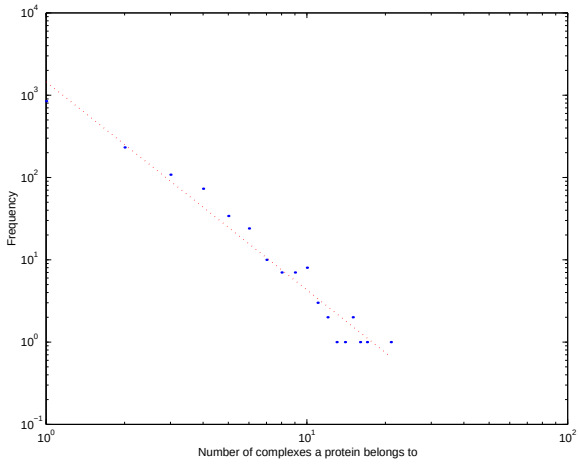

Figure 1. The frequency of proteins with a given degree plotted against the degree shows that the yeast protein hypergraph satisfies a power law degree distribution for the proteins.

3. The core of a hypergraph

Tong et al. [9] have considered graph models of binary protein interactions of a class of peptide-recognizing proteins. They have studied the k-cores of this proteomic network to identify significant interactions common to the computational network and an experimentally observed phage display network. Other authors, e.g., [7] have also studied the cores of protein-protein interaction graphs. Some of this work is done with an eye towards identifying biologically significant interactions, but some try to determine putative protein complexes in this manner. As we have pointed out earlier, the latter endeavor is error-prone since the proteins in a complex might have only few interaction partners. Others have conjectured the existence of a *core proteome*, a set of proteins with sequences, structures, and basic cellular roles that might be common to a set of organisms.

To identify the core proteome from a protein complex hypergraph, we need an algorithm for computing the maximum core of a hypergraph. We begin by considering the concept of a core in a graph, then extend it to a hypergraph, and finally design an algorithm for computing a core in a hypergraph.

The k-core of a graph G is a maximal subgraph of G in which every vertex has degree at least k in the subgraph. The k-core has been studied earlier in different contexts by several authors; e.g., see Sec. 3.7.2 in [6]. The maximum core of a graph corresponds to the maximum value of k for which the graph has a non-empty k-core. The maximum

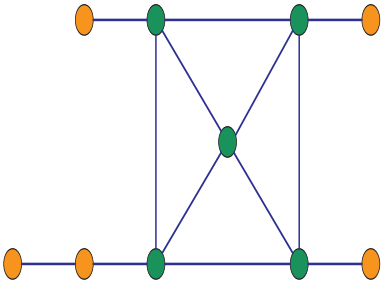

Figure 2. The k-core of a graph.

core in the graph shown in Fig. 2 is a 3-core (the subgraph consisting of vertices colored green and the subset of edges both of whose endpoints are green). The entire graph forms the 1-core, the 2-core is the same as the 3-core, and the 4-core is empty. Note that the k-core need not be a connected subgraph.

There is a linear-time algorithm (in the number of edges) for computing all the non-empty cores of a graph. The idea is to repeatedly remove a vertex v of minimum degree in the graph and all edges incident on it, updating the degrees of the neighbors of v in the residual graph as edges are deleted. The algorithm repeats this step until the graph is empty; the highest value of the minimum degree observed during the algorithm corresponds to the value of the maximum core. Note that the minimum degree in the residual graph could increase or decrease during the algorithm.

We now extend the concept of a k-core to a hypergraph. A *reduced* hypergraph is one in which every hyperedge is maximal, i.e., no hyperedge is contained in another. The k-core of a hypergraph H is a maximal subhypergraph that is reduced, and in which every vertex belongs to at least k hyperedges in the subhypergraph. Thus in a k-core, every hyperedge is maximal. When we delete a vertex from the hypergraph, a hyperedge that it belongs to gets deleted when it is no longer maximal. (This includes the special case of a hyperedge becoming empty due to vertex deletions.)

An algorithm for computing the k-core of a hypergraph is shown in Fig. 4. We can detect non-maximal hyperedges by counting overlaps among hyperedges instead of comparing set memberships, and will be discussed later.

Let $|V|$ denote the number of vertices, $|F|$ the number of hyperedges, Δ_V the maximum vertex degree, and Δ_F the maximum degree of a hyperedge. Let

$$\sum_{v \in V} d(v) = \sum_{f \in F} d(f) \equiv |E|,$$

denote the sum of the degrees of the vertices (or equivalently, the sum of the degrees of the hyperedges). Note that $|E|$ is the space needed to represent the hypergraph. Define the "degree-2" of a hyperedge f, $d_2(f)$, to be the number of hyperedges with which it has a vertex in common. (This is the number of hyperedges that f can reach by a path of length two (edges) in the bipartite graph $B(H)$.) Let $\Delta_{2,F}$ denote the maximum value of the degree-2 of the hyperedges in H.

We now establish the time complexity of the k-core algorithm. Initially we will consider all the steps in the algorithm excluding the maximality computation for the hyperedges.

A vertex v can be deleted from a hyperedge in $\ln \Delta_F$ time, and a hyperedge can be deleted from the adjacency set of a vertex in $\ln \Delta_V$ time, by maintaining these sets as balanced trees. The key observation to bounding the total number vertex deletions in the algorithm is that each vertex can be deleted from a hyperedge at most once. Hence the number of vertex deletions in the algorithm is $O(|E|)$. Similarly the number of times hyperedges are deleted from the adjacency lists of vertices is also bounded by $O(|E|)$. Hence the time required by vertex and hyperedge deletions in the algorithm can be bounded by $O(|E|(\ln \Delta_V + \ln \Delta_F))$.

Now we show that the maximality computations for the hyperedges can be performed without explicitly comparing the vertex lists of the hyperedges during the k- core algorithm. Initially we can compute the nonzero pairwise overlaps of the hyperedges by processing the adjacency lists of the vertices in time

$$O(\sum_{v \in V} d(v)^2) \ \ln \Delta_{2,F} = O(\Delta_V |E|) \ \ln \Delta_{2,F}.$$

The logarithmic factor comes from inserting into a balanced tree representation of the hyperedges that overlap with a given hyperedge. We need to check for non-maximality only those hyperedges whose degrees are decremented in the algorithm due to the deletion of a member vertex. The key observation is that a hyperedge is contained in another only when its current degree is equal to its current overlap with the latter hyperedge. Hence we can check for maximality without comparing set memberships, but by maintaining the current degrees and overlaps of the hyperedges. This cost can be bounded for the algorithm by

$$\sum_{v \in V} \sum_{f \ni v} d_2(f) \leq \Delta_{2,F} |E|.$$

The time complexity of the k-core algorithm is then bounded by

$$O(|E|(\Delta_{2,F} + \Delta_V \ln \Delta_{2,F})).$$

We have implemented this k-core algorithm for hypergraphs. On an Intel Xeon 2 GHz processor with 1GB memory, the k-core computation of the yeast hypergraph

took 0.47 seconds. While the Cellzome study is the largest such current study on protein complexes, it leads to a small hypergraph from a computational perspective. However, larger proteomic studies, e.g., ones that scale to the human proteome, or proteomic studies involving multiple organisms, will require high performance algorithms and software. Meanwhile, we have run the hypergraph core algorithm on larger hypergraphs obtained from scientific computing applications (from the Matrix Market, URL: `math.nist.gov/MatrixMarket`), and the results are included in Table 1. The results show that if the numbers of vertices and hyperedges in the core are large, then the run times can be substantial; hence for large hypergraphs, a parallel algorithm will need to be designed. The maximum core in the yeast protein complex hypergraph is a 6-core consisting of 41 proteins and 54 complexes.

It has been conjectured that the core of a protein-protein interaction network or a protein-complex hypergraph might represent the *core proteome*, a set of proteins and complexes performing essential cellular functions and that they might be common to many organisms. We tested this conjecture on the Cellzome yeast protein complex hypergraph.

We found that of the 9 of the 41 proteins in the 6-core are currently unknown or have unknown function; 22 of the 32 proteins that are known or have known function are essential; i.e., deleting the corresponding gene is lethal to yeast. Also, 24 have reported homologs in the Saccharomyces Genome Database (URL: `www.yeastgenome.org`), three of which belong to the subset of proteins that are unknown or have unknown function. The homologous proteins belong to organisms such as the human, mouse, E.coli, and bacillus. Since the Comprehensive Yeast Genome Database reports 878 essential and $3,158$ non-essential genes, essential proteins constitute a higher fraction of the proteins in the core.

We computed maximum cores in the *protein-protein interaction networks* for yeast and drosophila obtained from the Database of Interacting Proteins (DIP) circa. Nov 2003 (URL: `dip.doe-mbi.ucla.edu/dip/`). The maximum k-core for the yeast had $k = 10$ with 33 proteins, while the drosophila network had $k = 8$ with 577 proteins. The total number of proteins in the yeast network was 4746, while that in the fruitfly was 7065.

4. Vertex Covers

We now consider the reliability of the Cellzome experiment. Out of the total $1,361$ proteins in the study, 589 proteins were used as bait proteins. Each bait protein pulls down one or more complexes it belongs to. The large-scale identification of complexes from proteins associated with bait proteins is unfortunately more error-prone than smaller-scale studies (the Cellzome experiments report a reproducibility of 70%). Hence we consider the question of choosing the bait proteins from the protein complex hypergraph once it is partially known. We believe this could be useful in two situations: the first is when we wish to repeat the experiments to improve the reliability of the data; the second is when we wish to use one organism as a model to identify the protein complexes in a related organism. The bait selection problem can be formulated as the problem of choosing vertex covers in hypergraphs of minimum size or weight.

4.1. Vertex cover for hypergraphs

Given a hypergraph $H = (V, F)$, and non-negative weights on the vertices, the minimum weight vertex cover problem is to find a subset of vertices $C \subseteq V$ such that C includes at least one vertex from each hyperedge, and the sum of the weights of the vertices in C is minimum. The problem is NP-hard, but approximation algorithms for finding near-optimal vertex covers exist.

The vertex cover problem for a hypergraph can be reduced to the set cover problem for a collection of subsets of a ground set S. Given a set S and a collection of subsets of S with weights assigned to each subset, the set cover problem is to choose some of the subsets from the collection so that the union of the chosen subsets is S and the sum of the weights of the subsets chosen is minimum. The vertex cover problem for a hypergraph is equivalent to the set cover problem by viewing each vertex in the hypergraph in terms of its adjacency set.

The greedy algorithm for finding an approximate vertex cover of minimum weight in a hypergraph chooses a vertex with minimum *cost* (to be defined) for each vertex. During the course of the algorithm, the current cost $\alpha(v)$ of a vertex v is obtained by distributing its weight equally among the hyperedges it belongs to that are currently uncovered. Let F_i denote the set of hyperedges not yet covered by a partial vertex cover at the beginning of the ith iteration of the algorithm; then

$$\alpha(v) = w(v)/|\mathrm{adj}(v) \cap F_i|.$$

At each step, the algorithm chooses a vertex with minimum cost $\alpha(v)$ to include in the partial cover, and deletes all hyperedges it covers. The algorithm is described in Fig. 5.

The greedy algorithm for set cover is due to Johnson, Chvatal and Lovasz [10], and is an $H_m = O(\log m)$ approximation algorithm, where $m \equiv |F|$ is the number of hyperedges, and H_m is the mth harmonic number: $H_m = 1 + 1/2 + \ldots + 1/m$. We have implemented this algorithm in time

$$O(\sum_{v \in V} d_2(v)) \leq O(\Delta_F |E|).$$

(Here $d_2(v)$ is the "degree-2" of a vertex v; i.e., the number of distinct vertices other than v in all the hyperedges that v

belongs to. It is equal to the number of vertices that can be reached from v by a path of length two in the bipartite graph $B(H)$.)

A variation on the minimum weight vertex cover problem is the minimum weight multicover problem, where we wish to cover each hyperedge f with $r_f \geq 1$ vertices, where r_f is a given positive integer that depends on f. A simple modification to the algorithm given above solves this latter problem with the same approximation ratio. The change is that now when a vertex v_i is included in the cover, only those hyperedges $f \in \mathrm{adj}(v_i)$ whose multicover requirements have been met are deleted.

Dual and primal-dual algorithms with approximation ratios that depend on the maximum degree of a vertex can also be designed for the weighted vertex cover problem in hypergraphs. For the yeast protein complex hypergraph, the greedy algorithm yields a better approximation bound than these algorithms. However, it is not clear if these algorithms will be practically inferior or superior in quality to the greedy algorithm discussed here. This is the subject of current work.

4.2. Results

For the yeast protein complex hypergraph, the greedy algorithm finds an approximate minimum cardinality vertex cover consisting of 109 proteins, with the average degree of a protein in the cover being approximately 3.7. However, a protein belonging to many complexes might not unambiguously pull down all the complexes it belongs to; hence it is preferable to choose as bait proteins those of low degree. We accomplish this by weighting each protein by the square of their degrees. The greedy algorithm now finds an approximate minimum weight vertex cover with 233 proteins, with the average degree of a protein in the cover reduced to approximately 1.14. In comparison, the Cellzome experiment reports complexes pulled down from 459 bait proteins, with the average degree of a bait protein approximately 1.85.

Since the reproducibility of the Cellzome experiment is low, it would be more reliable to cover each complex more than once. We have also run a multicover algorithm to cover each hyperedge twice, excluding three complexes that consist of a single protein. The remaining 229 complexes are covered twice each by 558 proteins, with the average degree of a protein in the cover approximately 1.74.

The majority of the bait proteins of the Cellzome study (429) pull down only one complex, with 26 pulling down two complexes, and the remaining 4 pulling down 3 complexes. Of course, factors other than the degree influence the suitability of a bait protein in the TAP experiment. A proteomics expert could set preferences for each protein to be used as a bait, and our algorithms could work with those weights to make a meaningful choice of a candidate set of bait proteins for large-scale proteomics experiments.

5. Conclusions

We have defined the concept of a k-core of a hypergraph, designed an algorithm to compute it, and used it to identify the core proteome of a yeast protein complex hypergraph. The yeast core proteome is rich in both essential and homologous proteins. We have also implemented a greedy approximation algorithm for computing variations of minimum weighted vertex covers in a hypergraph, and used it to suggest candidate bait proteins in the Cellzome experimental methodology for obtaining the protein complex data. We believe this algorithm could be useful for proteomics experts to choose bait proteins to improve the reliability of the large-scale proteomics experiment, and to scale up to experimentation on proteomes of other organisms.

Acknowledgments. We thank Prof. Chris Osgood of the Biological Sciences Department at Old Dominion University for several helpful conversations.

References

[1] R. Albert and A. Barabasi, Statistical mechanics of complex networks, *Physics Review*, Vol. 74, pp. 47-97, 2002.

[2] S. Bornholdt and H. G. Schuster (eds.), *Handbook of Graphs and Networks*, Wiley VCH, 2003.

[3] A-C. Gavin, M. Bosche, R. Krause et al, Functional organization of the yeast proteome by systematic analysis of the protein complexes, *Nature*, 415, pp. 141-147, 2002.

[4] L. Giot, J. S. Bader, C. Brouwer et al., A protein interaction map of Drosophila melanogaster, *Science*, 312, pp. 1727–1736, 2003.

[5] Y. Ho, A. Gruhler, A. Heilbut et al, Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry, *Nature*, 415, pp. 180–183, 2002.

[6] D. S. Hochbaum, *Approximation Algorithms for NP-hard problems*, PWS Publishing Company, 1997.

[7] R. Jansen, H. Y, D. Greenbaum et al., A Bayesian network approach for predicting protein-protein interactions from genomic data, *Science*, 302, pp. 449–453, 2003.

[8] S. Maslov, K. Sneppen and U. Alon, Correlation profiles and motifs in complex networks, In [2], pp. 168–198, 2003.

[9] A. H. Y. Tong, B. Drees, G. Nardeli et al, A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules, *Science*, 295, pp. 321–324, 2002.

[10] V. Vazirani, *Approximation Algorithms*, Springer Verlag, 2001.

[11] A. Wagner and D. Fell, The small world inside large metabolic networks, *Procs. Royal Society*, pp. 1803-1810, 2001.

```
while there are vertices with degree < k do
    for each such vertex v →
        for each hyperedge f ∋ v →
            delete v from adj(f);
            decrement d(f) by one;
            if f is non-maximal then
                for each vertex w ∈ f →
                    delete f from adj(w);
                    decrement d(w) by one;
                    if d(w) < k then
                        include w in list of
                        vertices with degree < k;
```

Figure 4. An algorithm for computing the k-core of a hypergraph. The variable $d(.)$ denotes the degree of a vertex or a hyperedge.

```
Initialize:
i := 1; (iteration number)
C := ∅; (cover)
F_1 := F;
(hyperedges yet to be covered)
while F_i ≠ ∅ do
    for v ∈ V \ C →
        Choose a vertex v_i
        with min cost α(v);
        Add v_i to the cover C;
        F_{i+1} := F_i \ adj(v_i);
        i := i + 1;
```

Figure 5. The greedy algorithm for computing an approximate minimum weight vertex cover of a hypergraph.

hypergraph	$\|V\|$	$\|F\|$	$\|E\|$	Δ_V	Δ_F	$\Delta_{2,F}$	max core	core $\|V\|$	core $\|F\|$	core time
Cellzome	1361	232	2678	21	88	84	6	41	54	0.47s
bfw	62	62	450	21	17	47	4	31	23	0.06 s
fdp1	656	656	10038	62	62	191	3	276	28	0.5 s
stk13	2003	2003	42943	82	64	287	7	1159	659	35.7 s
utm	3060	3060	42211	29	20	129	12	160	220	4.5 m
fdp11	22294	22294	623554	43	43	256	13	20719	18578	6.8 h

Table 1. Statistics on hypergraphs and their maximum cores from Cellzome and scientific computing applications. Legends for times are h: hours, m: minutes, and s: seconds.

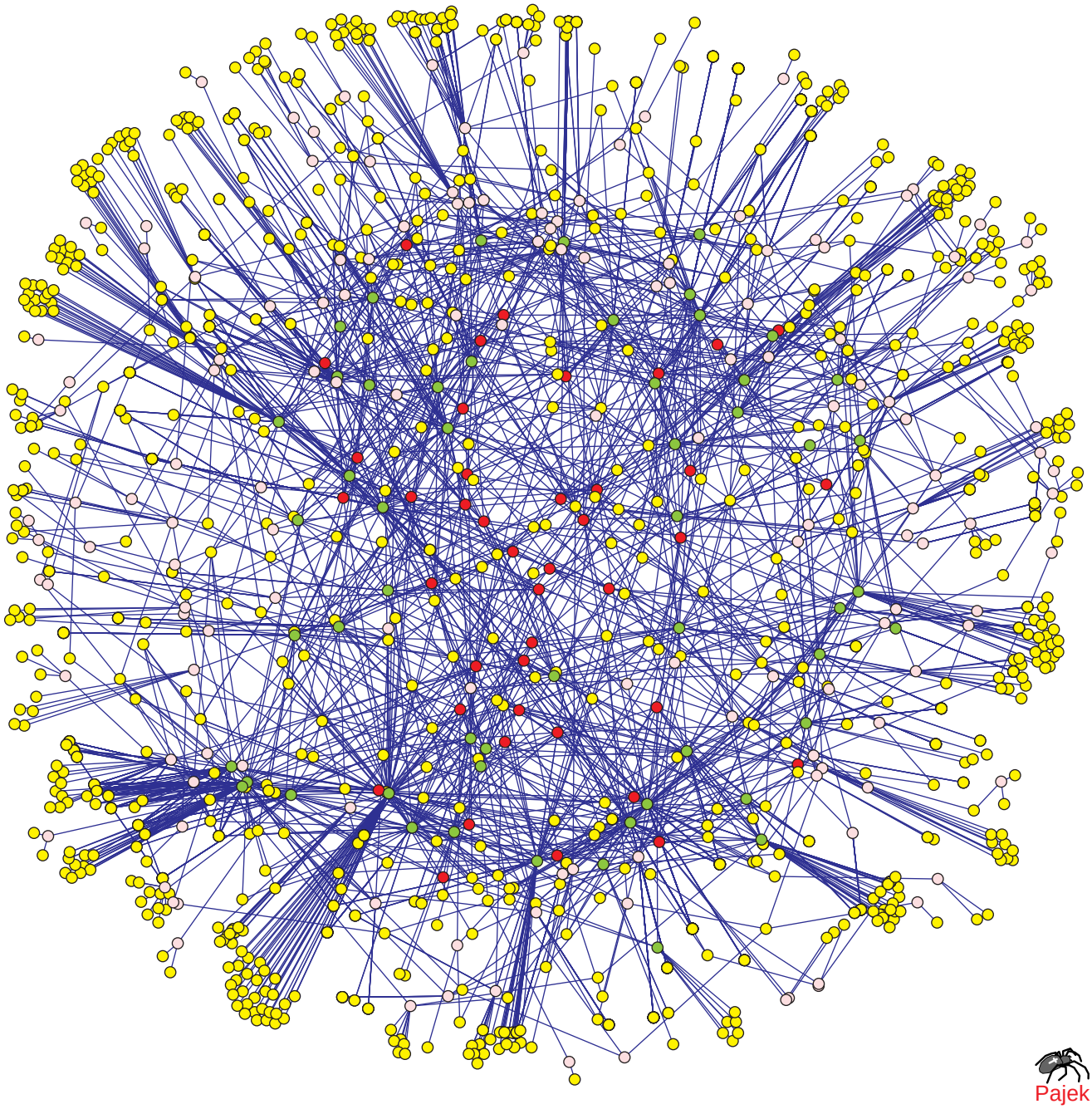

Figure 3. The yeast protein complex hypergraph and its maximum core. Yellow and red nodes correspond to proteins, and pink and green nodes correspond to complexes. Red nodes correspond to proteins and green nodes to complexes in the maximum 6-core.