Transitive Closure and Metric Inequality of Weighted Graphs – Detecting Protein Interaction Modules Using Cliques

Chris Ding, Xiaofeng He, Hui Xiong, Hanchuan Peng, Stephen R. Holbrook
Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720.

June 2, 2006. Updated October 1, 2007

Abstract

We study transitivity properties of edge weights in complex networks. We show that enforcing transitivity in affinity graphs leads to maximal transitive affinities which satisfy transitivity inequalities equivalent to ultra-metric and harmonic triangle inequalities. This can be used to define transitive closure on weighted graphs, and can be computed efficiently using a modified Floyd-Warshall algorithm. From this, we extend the clique concept from unweighted graph to weighted graph. These new concepts and theorems are extended to dissimilarity graphs. We outline several applications and present results of detecting protein functional modules in a protein interaction network.

Keywords: transitive closure, ultrametric, triangle inequality, weighted graph, protein interaction

1 Introduction

Complex network studies include the internet and the Web, social networks, biological networks, etc. Much progress has been made in degree distribution, network topology, growth models, small-world effect, etc.

In this paper, we study transitivity properties of edge weights, reflecting affinity between two proteins, relatedness between two persons or similarity between two webpages. Generally speaking, these complex networks exhibit some level of transitivity. For example, for 3 persons A, B, C, if A is related to (a friend of) B and B is related to C, A is likely to be related to C, at least in comparison with two randomly selected persons. In the protein interaction network, if protein p_i interacts with protein p_j and protein p_j interacts with protein p_k, it is likely that protein p_i also interacts with protein p_k. Another way to characterize the same feature is to say that the average clustering coefficient of the network is large (in the language of small world(Watts and Strogatz, 1998)).

Beyond this general understanding, not many studies on network transitivity exist so far. In the social network field, there are some studies focusing on statistical evidence of this being true (Wasserman and Faust, 1994). We wish to push this further and in a somewhat different direction. We ask: Over a large number of different networks, what kind of transitivity we should expect? What are the consequences of transitivity of network affinity (edge weights)?

At present, most network models use unweighted graphs (edge weights are either 1 or 0). This is a simplification convenient for understanding large scale properties. A detailed study of transitivity requires a "quantitative" analysis, i.e., the edge weights in the network are more realistically assigned on a gradual scale. For example, in a protein interaction network, the weight (the interaction strength) should be the

probability that two proteins have a synergistic interaction. In a social network data (Freeman's electronic information exchange system data(Wasserman and Faust, 1994)), the relationship between two persons has 5 scales: (1) don't know each other; (2) have heard of the other, but never met; (3) have met ; (4) are friends; (5) are close friends. Therefore, we focus on weighted graphs.

We approach the transitivity problem by seeking maximum mathematical consistency, and relate it to the well-known properties of metric distances, such as the triangle inequality and the ultra-metric inequality. Note that we are not "asserting" that complex networks "must" follow mathematical consistency. Our goal is to see how far we can go by assuming mathematical consistency.

In §2 we introduce a new concept of transitive affinity and discuss its properties. We show that transitive affinity satisfies an inequality which is shown to be equivalent to the ultra-metric inequality of a distance metric. Through this, we introduce the concept of transitive closure of a weighted graph. An efficent algorithm is also presented. In §3- §4, the transitive closure on affinity graphs is extended to dissimilarity graphs and to situations where triangle inequalities hold.

In §5, we outline several applications of transitive closure of weighted graphs. In §6, we extend the clique concept from unweighted graph to weighted graph. In §7 - §8, we present results of detecting protein functional modules from protein interaction networks. A preliminary version of this work has appeared in Ding et al. (2005).

2 Transitive affinity of a weighted affinity graph

2.1 Transitivity and associativity

We study transitive properties of affinity graphs where the edge weights measure pairwise affinities between nodes. We use the protein interaction as example.

Suppose protein p_1 strongly interacts with p_2, say $w_{12} = 0.7$. Suppose protein p_2 strongly interacts with p_3, say $w_{23} = 0.9$. Given these, there is a reasonable probability that protein p_1 also interacts with p_3, i.e., there is a nonzero *transitive* affinity between p_1 and p_3. There are three probable quantitative ways to characterize the transitivity:

$$\mathbb{T}_{max}(w_{12}, w_{23}) = \max(w_{12}, w_{23}) = 0.9, \tag{1}$$

$$\mathbb{T}_{avg}(w_{12}, w_{23}) = (w_{12} + w_{23})/2 = 0.8, \tag{2}$$

$$\mathbb{T}_{min}(w_{12}, w_{23}) = \min(w_{12}, w_{23}) = 0.7. \tag{3}$$

Let $P_{13} = (1, 2, 3)$ be the path connecting p_1 to p_3. We say that $\mathbb{T}(w_{12}, w_{23}) = \mathbb{T}(P_{13}) = \mathbb{T}(1, 2, 3)$ is the transitive affinity along path P_{13}, which is a 2-hop path (passing through 2 edges). The question is: among the three possible choices, which one best captures the transitivity and satisfies many other consitency principles?

Suppose protein p_3 interacts with another protein p_4. Thus a nonzero transitive affinity $\mathbb{T}(P_{14}) = \mathbb{T}(1, 2, 3, 4)$ exists between p_1 and p_4. For the transitive affinity to be consistent along a path of $(1, 2, 3, 4)$, we require it have the associativity:

$$\mathbb{T}(\mathbb{T}(w_{12}, w_{23}), w_{34}) = \mathbb{T}(w_{12}, \mathbb{T}(w_{23}, w_{34})) \tag{4}$$

With this associativity, $\mathbb{T}(w_{12}, w_{23}, w_{34})$ is uniquely defined and is denoted as $\mathbb{T}(P_{14})$. One can easily extend this to longer paths. It is clear that $\mathbb{T}_{avg}$ does not satisfy associativity; this rules out $\mathbb{T}_{avg}$. Both $\mathbb{T}_{max}$ and $\mathbb{T}_{min}$ satisfy associativity. In general, transitivity is regulated by the weakest link on the path. Thus we choose $\mathbb{T} = \mathbb{T}_{min}$. In the rest of this paper, we study the transitivity associated with $\mathbb{T}_{min}$. The transitive affinity on the path $P_{ij} = (i, k_1, \cdots, k_m, j)$ is therefore

$$\mathbb{T}(P_{ij}) = \min\ (w_{i,k_1}, w_{k_1,k_2}, \cdots, w_{k_{m-1},k_m}, w_{k_m,j}). \tag{5}$$

2.2 Maximal transitive affinity

Transitive affinity is defined on a specific path. In general, fixing nodes i, j, transitive affinities on different paths connecting i, j are different. For this reason, we define maximal transitive affinity between i, j as

$$h_{ij} = \max_{P_{ij}} \mathbb{T}(P_{ij}), \tag{6}$$

where P_{ij} is any possible path between i, j. Given a graph, the maximal transitive affinity between any pair of vertices is uniquely defined. We can show that

$$h_{ij} \geq w_{ij}, \forall i, j. \tag{7}$$

This implies the maximal transitive affinities bring nodes of the network more close than the original affinity metric. Furthermore, we can prove that

Theorem 1. For any weighted graph, the maximal transitive affinity between any pair of vertices satisfies the following transitive affinity inequality relationship

$$h_{ij} \geq \min(h_{ik},\ h_{kj}) \quad \forall i, j, k. \tag{8}$$

Proof. Let(P_{ik}, P_{kj}) be a path starting at i, going through k, and ending at j. P_{ij} is a path starting at i and ending at j. Clearly $\{(P_{ik}, P_{kj})\}$ is a subset of $\{P_{ij}\}$. Thus,

$$\begin{aligned}
h_{ij} &= \max_{P_{ij}} \min\left[W(P_{ij})\right] && (9)\\
&\geq \max_{(P_{ik},P_{kj})} \min\left[W(P_{ik}, P_{kj})\right] && (10)\\
&= \max_{(P_{ik},P_{kj})} \min\Big[\min\left[W(P_{ik})\right], \min\left[W(P_{kj})\right]\Big] && (11)\\
&= \min\Big[\max_{P_{ik}}(\min\left[W(P_{ik})\right]), \max_{P_{kj}}(\min\left[W(P_{kj})\right])\Big] && (12)\\
&= \min(h_{ik},\ h_{kj}). && (13)
\end{aligned}$$

□

2.3 Metric inequality

Here we show that the transitive affinity inequality is identical to the ultra-metric of a distance metric, and therefore, a general principle.

Recall the definition of the metric space. A function $d(x_i, x_j) = d_{ij}$ on all pairs of objects in the space, i.e., pairwise dissimilarities, is a metric, if (m1) nonnegativity: $d_{ij} \geq 0 \; \forall i, j$. (m2) identity: $d_{ij} = 0$ if $x_i = x_j$. (m3) symmetry: $d_{ij} = d_{ji}$. (m4) triangle inequality

$$d_{ij} \leq d_{ik} + d_{kj}. \tag{14}$$

A metric function preserves the important notion of *distance*. Metric space has a large number of properties and is useful for many problems. A special case of metric space is ultra-metric space, where the triangle inequality is replaced by a stronger ultra-metric inequality

$$d_{ij} \leq \max(d_{ik}, d_{kj}). \tag{15}$$

A dissimilarity function satisfying the ultra-metric inequality also satisfies the triangle inequality; however, not all metric functions satisfy ultra-metric inequality.

First, we note that *similarity* is a decreasing function of *distance*: the more similar two objects are, the smaller their distance is. Thus $s_{ij} = f(d_{ij})$, where $f(\cdot)$ is a monotonic decreasing function (more precisely, a nonincreasing function).

Theorem 2. The distance-based ultra-metric inequality is identical to the similarity-based transitive affinity inequality, assuming that $s_{ij} = f(d_{ij})$ is a nonincreasing function.

Proof. By construction, $s_{ij} = f(d_{ij}) \geq f(\; \max(d_{ik}, d_{kj}) \;) = \min(f(d_{ik}), f(d_{kj}))$ □

The ultra-metric inequality Eq.(15)] can be viewed as a general principle (or a highly desirable property for distance functions) from the aximatic point of view, similar to the triangle inequality. Therefore, we may consider the transitive affinity inequality of Eq.(8) as a general principle that affinity functions should obey.

2.4 Transitive closure of a weighted graph

If we repeat the same line of discussions of transitive affinity in §2.1 - §2.2 on an unweighted graph, it is clear that the maximal transitive affinity h_{ij} will be 1 between any two nodes in a connected component. This process is very similar to the concept of transitive closure of a directed unweighted graph. Thus the transitive closure of an undirected unweighted graph is the complete graph on the nodes of any connected component.

Now we further extend this concept to an undirected weighted graph with weight $\{w_{ij}\}$. We replace the original weight w_{ij} by the maximal transitive affinity h_{ij}. The new graph weights are thus consistent with the idea of transitivity of affinity relationship, because it satisfies the transitive affinity inequality. We therefore have

Definition 1. The transitive closure of weighted graph G is formed by their maximal transitive affinity h_{ij}.

Remark. This definition can be extended to the transitive closure of a weighted directed graph, by generalizing the maximal transitive affinity h_{ij} to directed graphs in a straightfoward manner. For unweighted graphs, this definition reduces to the standard definition of transitive closure.

2.5 Computing transitive closure of a weighted graph

Computing maximal transitive affinity h_{ij} by its original definition Eq.(6) is farily complicated. Instead, we compute them using the dynamical programming approach similar to the Bellman-Ford algorithm for computing the shortest path problem Cormen et al. (1998). In Bellman-Ford algorithm, the deviation from the sub-optimal condition on each pair of nodes are iteratively tightened (*relaxed* in the original word).

Bellman-Ford algorithm is used to compute the shortest paths from a fixed node to the rest of the graph nodes. Generalizing Bellman-Ford algorithm to compute shortest path between all pair of nodes is the Floyd-Warshall algorithm which can be viewed as iteratively tightening the deviation from the sub-optimal condition. In shortest path problem, the sub-optimal condition is the triangle inequality Eq.(14). Coming back to our minimal transitive affinity h_{ij}, the sub-optimal condition is the transitive affinity inequality Eq.(8) for which h_{ij} must satisfies.

Theorem 3. The transitive closure of weighted graph G can be equivalently computed by iteratively increasing edge weights such that the transitive affinity inequality is satisfied.

This theorem is the consequence of Theorems 4A and 4B below.

Theorem 4A. Suppose the edge weights of given graph satisfy the transitive affinity inequality. The transitive affinities as defined in Eq.(6) are equal to edge weights: $h_{ij} = w_{ij}$. (This theorem plays the role of Lemma 24.14 convergence property in Cormen et al. (1998)).

Proof. We first show that for 2-hop paths, maximal transitive weight is equal to the edge weight. Fixing i, j, we consider a 2-hop path $i - k - j$. Since weights satistify transitive affinity inequality, w_{ij} must be larger or equal to 2-hop transitive weight $t(P_{ikj})$ for any k. Thus the 2-hop maximal transitive weight between $i, j.$ must be w_{ij}. This is valid for all i, j pairs.

Now consider 3-hop paths between i, j. Consider a 3-hop path $i - k - l - j$. Since for 2-hop paths, maximal transitive weight is equal to the edge weight, we can replace $i - k - l$ by $i - l$, or replace $k - l - j$ by $k - j$ when computing maximal transitive weight. Thus we need consider only 2-hop paths $i - l - j$ and $i - k - j$. Apply the 2-hop path equivalence property, these two paths have the same maximal transitive weight as path $i - j$. Thus the maximal transitive weight for the path $i - k - l - j$ is w_{ij}. This is valid for all possible i, k, l, j.

The proof for 4-hop paths, 5-hop paths, etc. are the same. □

Theorem 4B. Given a pair of nodes (i, j). Let $(i, k_1, k_2, \cdots, k_m, j)$ be the path with the eventual minimal transitive affinity h_{ij}. After successive relaxation (tightening) of edges $(i, k_1), (k_1, k_2), \cdots, (k_{m-1}, k_m), (k_m, j)$ in order, the transitive affinity h_{ij} achieves the final optimal minimal transitive affinity. This holds no matter what other edges relax occur. (This is Lemma 24.15, path relaxation property in Cormen et al. (1998)).

Proof. Given the optimal path, the length-2 transitive affinity is easily determined and this is obviously the optimal solution. The length-3 transitive affinity is easily determined based on the the length-2 transitive affinity and the weight of the 3rd edge. This is extended to the last edge on the path. □

Theorems 4A and 4B are the basis for the algorithm to compute minimal transitive affinity.

2.6 Modified Floyd-Warshall algorithm

Using Theorems 4A and 4B, we see that for computation of maximal transitive affinities on multi-hop paths, it is sufficient to consider 2-hop paths only and simultaneously tightening all h_{ij}'s. Eventually h_{ij} converges to the optimal solution. This 2-hop path (or triangle) only property is crucial to for an efficient algorithm implementation.

Since to verify whether a given graph weights satisfy transitive closure, we need to go through all possible triangles $\binom{n}{3} = n(n-1)(n-2)/3! = O(n^3)$ to check the transitive affinity inequality. This is the minimal computational cost.

Implementation of Theorem 4A and 4B can be shown to be a slight modification of the Floyd-Warshall algorithm for all-pair shortest paths in $O(n^3)$. Since $O(n^3)$ is the the minimal computational cost, the Floyd-Warshall algorithm is optimally efficient.

Given input W, the weight matrix of a network, the algorithm computes the maximal transitive affinity as the following:

Floyd-Warshall to compute transitive closure of graph with weights W.

```
1  H = W
2  for k = 1 to N
3       do for i = 1 to N
4            do for j = 1 to N
5                 h_ij = max(h_ij, min(h_ik, h_kj))
6  return H
```

The modification from standard the Floyd-Warshall algorithm is on Line 5, updating h_{ij}, which uses Eq.(8) for satisfying the transitive affinity inequality.

The correctness of this algorithms is proved by the following. (1) From Theorem 4B, for any optimal eventual path between any pair of nodes (i, j), the loop of $k = 1, \cdots, N$ ensure that h_{ij} converge to the final correct solution. This happens for all pair of node simultaneously, thus all h_{ij} are computed simultaneously. (2) If at some point before the loop of $k = 1, \cdots, N$ is finished, some of h_{ij} are already converged. Theorem 4A guarenttes that continued tightening does not cnange h_{ij} any more.

We note that although minimal transitive affinity is uniquely defined, the edge function (the function defined on the edges) which satisfy the transitive affinity inequality is not unique: for any $\{h_{ij}\}$ who satisfy the transitive affinity inequality, $\{2h_{ij}\}$ also satisfies the inequality. This is not a problem, however. Starting from w_{ij}, our algorithm compute the minimal increases necessary for $\{h_{ij}\}$ to satisfy the transitive affinity inequality. This gives the unique maximal transitive affinity.

3 Transitive dissimilarity and closure of a dissimilarity graph

The concepts and results in §2 for affinity graphs can be extended to dissimilarity graphs where the edge weights $D = (d_{ij})$ measure the "dis-similarity" or "distance". The "transitive dissimilarity" (or "transitive distance") on path P_{ij} is defined as

$$f(P_{ij}) = \max\,(d_{i,k_1}, d_{k_1,k_2}, \cdots, d_{k_{m-1},k_m}, d_{k_m,j}). \tag{16}$$

Fixing i, j, transitive dissimilarity depends on the particular path. Thus we define minimal transitive dissimilarity (analog of shortest path) between i, j as

$$g_{ij} = \min_{P_{ij}} f(P_{ij}). \tag{17}$$

We can easily show

$$g_{ij} \leq d_{ij}. \tag{18}$$

For affinity graphs considered in §2, the maximal transitive affinity there satifies the ultra-metric inequality (see Theorem 1). For dissimilarity graphs considered here, the minimal transitive dissimilarity satifies the similar ultra-metric inequality:

Theorem 5. For any weighted dissimilarity graph, the minimal transitive dissimilarity between any pair of vertices satisfies the ultra-metric inequality

$$g_{ij} \leq \max(g_{ik}, g_{kj}). \tag{19}$$

The proof is identical to the proof for Theorem 1. Therefore, $\{g_{ij}\}$ is consistent with the transitivity of dissimilarity relationship. We therefore define

Definition 2. The transitive closure of weighted dissimilarity G is formed by the minimal transitive dissimilarity $\{g_{ij}\}$.

Computing g_{ij} by its original definition, Eq.(17), is farily complicated. Instead we can efficiently compute them using the ultra-metric inequality (following the same analysis in §2.4):

Theorem 6. The transitive closure of a dissimilarity graph can be equivalently computed by reducing edge dissimilarities such that the ultra-metic inequality is satisfied.

Intuitively, we can see this from Eq.(18). The transitive closure of a dissimilarity graph can be computed using the same Floyd-Warshall algorithm in §2.6, with Line 1 replaced by $G = D$, and Line 5 replaced by $g_{ij} = \min(g_{ij}, \max(g_{ik}, g_{kj}))$.

4 Transitive closure based on the triangle inequality

In §3, the transitive closure of a weighted dissimilarity graph is formed by the minimal transitive dissimilarity which obeys the ultra-metric inequality Eq.(19). The ultra-metric inequality is a stronger form of the triangle inequality. Indeed, the definition of transitive dissimilarity of Eq.(16) is a stronger form of transitivity (dissimilarity does not increase as hops increase).

Here we show that by modifying the definition of the transitive dissimilarity (to a weaker form), the resulting minimal transitive dissimilarity satisfies the triangle inequality Eq.(14) and the original Floyd-Warshall algorithm. The minimal transitive dissimilarity is in fact, the usual shortest path distance.

The main purpose of this section is show that transitive dissimilarity of §3 is an natural extension of the usual shortest path problem. Another purpose is to motivate the development of §4.2 on the transitive affinity which satisfy some kind of triangle inequality, a softer version of the transitive affinity introduced in §2.

4.1 Transitive closure of dissimilarity graphs w.r.t. the triangle inequality

Instead of the definition of Eq.(16), we redefine the "transitive dissimilarity" on path P_{ij} as

$$f(P_{ij}) = d_{i,k_1} + d_{k_1,k_2} + \cdots + d_{k_{m-1},k_m} + d_{k_m,j}. \tag{20}$$

This is a weaker form of transitivity dissimilarity, which increase as hops increase. This definition satisfies the associativity. The *minimal transitive dissimilarity* $\{g_{ij}\}$ is defined to be the transitive dissimilarity between (i, j) that gives the minimal value, in similar fashion as in Eq.(17). One can see that this is just the standard *shortest path distance*.

It is known that the shortest path distance satisfies

Theorem 5A. For any dissimilarity graph, the minimal transitive dissimilarity defined through Eq.(20) and Eq.(17) satisfies the triangle inequality

$$g_{ij} \leq g_{ik} + g_{kj}. \tag{21}$$

Although this is obvious, we give a proof as follows (see proof of Theorem 1). P_{ij} is a path starting at i and ending at j. (P_{ik}, P_{kj}) is a path starting at i, going through k, and ending at j. Clearly $\{(P_{ik}, P_{kj})\}$ is a subset of $\{P_{ij}\}$. Thus we have

$$\begin{aligned} g_{ij} &= \min_{P_{ij}} f(P_{ij}) \\ &\leq \min_{(P_{ik},P_{kj})} f(P_{ik}, P_{kj}) \\ &= \min_{(P_{ik},P_{kj})} [f(P_{ik}) + f(P_{kj})] \\ &= \min_{P_{ik}} f(P_{ik}) + \min_{P_{kj}} f(P_{kj}) \\ &= g_{ik} + g_{kj}. \end{aligned}$$

Following Definition 2, we can define

Definition 3. The transitive closure of a dissimilarity graph w.r.t. the triangle inequality is formed by the minimal transitive dissimilarity defined through Eq.(20) and Eq.(17).

We can show that the minimal transitive dissimilarity $\{g_{ij}\}$ is smaller than the original dissimilarities, the inequality of Eq.(18). Following Theorem 6, we can prove

Theorem 6A. The transitive closure of a dissimilarity graph w.r.t. the triangle inequality can be equivalently computed by iteratively reducing edge dissimilarities when tightening the sub-optimal condition, i.e., to satisfy the triangle inequality.

This transitive closure can be computed using the original Floyd-Warshall algorithm, i.e., the algorithm in §2.6, with Line 5 replaced by $g_{ij} = \min(g_{ij}, g_{ik} + g_{kj})$.

Therefore, given a dissimilarity graph G, we can compute the minimal transitive dissimilarity defined through Eq.(20) and Eq.(17). This forms the transitive closure of G w.r.t. the triangle inequality.

The benefit of transitive closure w.r.t. the triangle inequality is that the edge weights are modified (reduced) relatively small (in comparison to the transitive closure w.r.t. ultra-metric inequality).

4.2 Transitive closure of affinity graphs w.r.t. the harmonic triangle inequality

The concept of transitive closure of a dissimilarity graph w.r.t. to triangle inequality, discussed in §4.1, can be extended to affinity graphs.

Although the ultra-metric inequality of distance metric has a natural counterpart in affinity measures as shown in Theorem 2 in §2.3, the triangle inequality of distance metric has no clear counterpart in affinity measures.

The simplest and perhaps most natural counterpart is obtained by assuming that affinity is inversely proportional to distance: $s_{ij} = const/d_{ij}$. From this, we obtain

$$\frac{1}{s_{ij}} \leq \frac{1}{s_{ik}} + \frac{1}{s_{kj}} \tag{22}$$

We call this the "harmonic triangle inequality" since it resembles the harmonic average: $\frac{1}{<x>} = \frac{1}{2}(\frac{1}{x_1} + \frac{1}{x_2})$. With this, we can define a new transitive affinity, similar to the transitive dissimilarity of Eq.(20),

$$\mathbb{T}(P_{ij}) = t(P_{ij}), \;\; \frac{1}{t(P_{ij})} = \frac{1}{w_{i,k_1}} + \frac{1}{w_{k_1,k_2}} + \cdots + \frac{1}{w_{k_{m-1},k_m}} + \frac{1}{w_{k_m,j}}. \tag{23}$$

in contrast to Eq.(5). Clearly, this transitive affinity satisfies associativity. From this, we can define the *maximal transitive affinity* through Eq.(6). Similar to Theorem 1, we have

Theorem 1A. For any affinity graph, the maximal transitive affinity defined through Eq.(23) and Eq.(6) satisfies the harmonic triangle inequality Eq.(22).

The proof is the same as for Eq.(21)). This maximal transitive affinity forms the transitive closure of an affinity graph:

Definition 4. The transitive closure of an affinity graph w.r.t. the harmonic triangle inequality is defined by the maximal transitive affinity defined through Eq.(23) and Eq.(6).

This transitive closure can be computed using the same Floyd-Warshall algorithm in §2.6, with Line 5 replaced by $h_{ij} = \max[h_{ij}, h_{ik}h_{kj}/(h_{ik} + h_{kj})]$.

The benefit of this transitive closure w.r.t. the harmonic triangle inequality is that edge weights are modified (increased) relatively small (in comparison to the transitive closure w.r.t. ultra-metric inequality in §2.4).

5 Applications of transitive closure

Here we discuss practical applications of transitive closure of weigted graphs.

First, in transitive closure, the originally not-so-densely connected subgraph structures become more dense, therefore this facilitates the discovery of these dense subgraph structures, i.e., communities in the network. In this direction, we will discuss an application to protein interaction in §7-8.

Second, in standard web link structure analysis, the web is represented by an unweighted directed graph. Since our approach emphasizes weighted graphs, we can incorporate webpage content into the link structure by weighting the link with the cosine similarity between the two webpage word contents. In this way, the transitive affinity along a path decreases with the number of edges because its strength is the weakest link between i, j. This is more natural than unweighted link analysis approaches.

Third, transitive closure provides a way to update affinity weights in a changing environment. Suppose we are given a social network. After a while two persons get married. This marriage will lead to a series of changes in the social network. Transitive closure provides a quantitative way to update the network weights. First the edge weight between the two persons in the graph is suddenly increased to maximum.

Second, the affinity weights between their in-laws are increased such that the transitive inequality holds for every node triple, etc.

6 Cliques on a weighted graph

A clique is the tightest structure or the densest subgraph in an unweighted graph, because every nodes in a clique connect to every other nodes. We generalize the concept of clique to weighted graphs and introduce an algorithm to compute them. The generalization is based on the theorem due to Motzkin and Straus (Motzkin and Straus, 1965) which relates maximal cliques of an unweighted undirected graph to the optimization of a quadratic function.

Let $G = (V, E)$ be an unweighted undirected graph of $n = |V|$ vertices and $|E|$ edges with adjacency matrix A. Define a vector $\mathbf{x} = (x_1, \cdots, x_n)^T$ on the vertices, i.e., $\mathbf{x} \in \mathbb{R}^n$. Consider the optimization problem:

$$\max_{\mathbf{x} \in S_n} J(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} \tag{24}$$

where $\mathbf{x}$ is restricted to the unit simplex S_n defined as

$$S_n : \quad x_1 + \cdots + x_n = 1, \; x_i \geq 0, \; \forall i. \tag{25}$$

The nonzero elements in the solution vector $\mathbf{x}$ play an important role. In particular, we define a characteristic vector of a subset C of vertices as $\mathbf{x}^C = (x_1^c, \cdots, x_n^c)^T$, where $x_i^c = 1/|C|$ if $i \in C$, $x_i^c = 0$ otherwise. The following theorem make the connection between vertices corresponding to nonzero elements to maximal cliques.

Theorem 7[Motzkin and Straus]. (1) Subset C is the largest maximal clique if and only if its characteristic vector is a global optimal solution of the optimization problem; (2) Subset C is a maximal clique if and only if its characteristic vector is a local optimal solution of the above optimization problem.

The Motzkin-Straus Theorem provides a convenient formalism to generalize the concept of cliques to weighted graphs.

Definition 5. *Generalized cliques* in a weighted graph with adjacency matrix A. The subset of vertices corresponding to the nonzero elements in the optimal solution $\mathbf{x}^*$ form a clique in A.

The key to this generalization is the recognition of the L_1-type constraint of Eq.(25) in the quadratic programming problem of Eq.(24) (The L_p-norm of a vector $\mathbf{x}$ in n-dimensional space is defined as $||\mathbf{x}||_p = (\sum_{j=1}^n |x_j|^p)^{1/p}, \; 1 \leq p \leq \infty$.) It is well-known (Tibshirani, 1996; Hastie et al., 2001) that this L_1-type constraint leads to sparse solutions, i.e., many if not most of entries in the final optimal solution $\mathbf{x}^*$ are zero. This sparsity property of the solution is the theoretical basis for our generalization of the Motzkin-Straus formalism to define cliques in weighted graphs. On the other hand, if the L_1-type constraint of Eq.(25) is replaced by a L_2 constraint: $\sum_i x_i^2 = 1$ and $x_i \geq 0$, the solution will not be sparse — almost every node will be in the clique. [1]

The quadratic programming problem of Eq.(24) can be solved by a simple method developed in the biological evolution field (Pelillo et al., 1999). The method is an iterative algorithm updating a current

[1] Using L_2 constraint, the solution is the principal eigenvector of A. By Perron's Theorem, all entries of the principal eigenvector are nonnegative.

solution vector using:

$$x_i \leftarrow x_i \frac{(A\mathbf{x})_i}{\mathbf{x}^T A\mathbf{x}}, \ \forall i. \tag{26}$$

We can show the following: (1) Feasibility of the solution: if the initial $\mathbf{x} \in S_n$ [defined in Eq.(25)], it will remain in S_n, since $\sum_i x_i = \sum_i x_i (A\mathbf{x})_i / (\mathbf{x}^T A\mathbf{x}) = 1$. (2) Correctness of the solution: we can prove that at convergence, the solution satisifies the KKT condition of the optimazation theory. (3) Convergence of the algorithm, by showing that the Lagrangian function for constraint optimization

$$L(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} - \lambda(\sum_i x_i - 1) \tag{27}$$

is monotonically increasing (or non-decreasing) under the above update rule: $L(\mathbf{x}^{(1)}) \leq J(\mathbf{x}^{(2)}) \leq \cdots$. The Lagrangian multiplier λ is to enforce the constraint $\sum_i x_i = 1$. Since $L(\mathbf{x})$ is bounded above, the updating algorithm converges.

If adjacency matrix A is positive definite, the objective J is a convex function and the optimal maxima is also the global maxima. Unfortunately, for many applications, A is in generally indefinite. There are a large number of local maximas, each representing a densely connected subgraph.

We use the above updating algorithm to solve for cliques. After one local optimal solution is obtained, the clique corresponding to non-zero entries in the solution vector is extracted; these nodes are eliminated from the graph. We solve for another local optimal solution and its corresponding clique, etc.

7 Application to the protein interaction network

In §7 - §8, we present application of transitive affinity and transitive closure to protein interaction networks.

Proteins carry out cellular functions and processes in a modular fashion, involving multiple interacting proteins. Identification of protein functional modules thus becomes an urgent research topic. There is a large body of genome-wide comprehensive experiments on protein interaction networks. The two-hybrid genetic screen (Ito et al., 2001; Uetz. et al., 2000) yields binary interaction data. High throughput mass spectrometry methods (Gavin et al., 2002; Ho et al., 2002) combine tagged "bait" proteins and protein-complex purification schemes with mass spectrometric measurements to yield physiologically relevant data on intact multi-protein complexes.

However, these high-throughput experiments are often associated with large false-positives. Protein interaction data obtained in two independent experiments (Ito et al., 2001) and (Uetz. et al., 2000) overlap by less than 4%. Therefore, using computational methods to predict protein interaction modules from these data is an important way to extract additional information.

An effective approach for predicting protein modules is to detect the densely connected subgraphs in protein interaction networks. The most intuitive definition of a densely connected subgraph is clique. One difficulty with this approach is that protein interactions are typically very sparse. Thus the cliques detected are very small. One way to overcome this problem is to use k-core (Ho et al., 2002; Bader and Hogue, 2002), where each protein only interacts with a fraction of other proteins in the subgraph. This relaxes the strict definition of clique, but it introduces the issue of how to choose the parameter k. A different approach using hyperclique has also been developed (Xiong et al., 2005).

We resolve the sparsity issue by using the transitive closure of §2. The idea is that on transitive closure, missing connection/interactions will be added and the graph becomes dense; therefore the detected cliques will be larger, thus more close to the true protein interaction modules.

Note that the simplest protein interaction network is unweighted, i.e., the interaction strength is either 1 or 0. This is a crude approximation. More refined descriptions use a weighted graph, assigning a probability or level of certainty that two proteins interact. Thus, the definition of cliques needs to be generalized to weighted graphs. This has been done in §6.

8 Experiments on Yeast multi-protein complex data

Two datasets of high-throughput mass spectrometry analysis of multi-protein complexes are available for the yeast *S. Cerevisiae*(Gavin et al., 2002; Ho et al., 2002). Studies have showen that the tandem affinity purification using mass spectrometry (TAP-MS) dataset by Gavin, *et al.* (Gavin et al., 2002) has the highest accuracy for predicting protein functions. Hence we use this data.

A protein complex is an assembly of a small number of proteins in permanent contact and usually perform a clear and specific biological function. A multi-protein complex can be represented as a bipartite graph which in turn gives several important quantities(Ding et al., 2004). The bipartite graph has two type of nodes: p-nodes denote proteins and are represented by rows in the bipartite graph adjacency matrix B. The c-nodes denote protein complexes, and are represented by columns in B. The entries of bipartite graph adjacency matrix $B = (b_{ij})$ is

$$b_{ij} = \begin{cases} 1 & \text{if protein } p_i \text{ is in protein complex } c_j \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

From the bipartite graph, we can naturally obtain the following two weighted interaction networks:

Protein - Protein Interactions. The interaction strength between two proteins p_i, p_j is given by $(BB^T)_{ij}$ which is the number of protein complexes containing both proteins p_i, p_j. $(BB^T)_{ii}$ = the number of protein complexes that protein p_i is involved.

Complex - Complex Cross Talk Associations. The interaction strength between two protein complexes c_i, c_j is given by $(B^TB)_{ij}$, which is the number of proteins shared by protein complexes c_i, c_j. Note $(B^TB)_{jj}$ = the number of proteins contained in the protein complex c_j.

To see clearly the net effects of transitive closure, we run the clique finding algorithm on two network weights: the original network $W = BB^T$ and its transitive closure W_{TC}, which is obtained using the modified Floyd-Warshall algorithm of §2.6. Figure 1 shows the original sparse protein interaction network (the color-coded adjacency matrix) and the dense protein interaction network produced by the transitive closure. The graph has 1,440 distinct proteins. We obtain two sets of cliques C and C_{TC} as the results, corresponding to W and W_{TC} respectively. These cliques are also shown in Figure 1. The obtained cliques are summarized below:

	# of cliques	Average-Size	Internal-Weight	External-Weight
C	296	4	0.59	0.014
C_{TC}	82	15	0.68	0.010

On the original network, the resulting C contains 296 cliques. On the transitive closure, the resulting C_{TC} contains 82 cliques. The original network is very sparse, thus we get a larger number of cliques

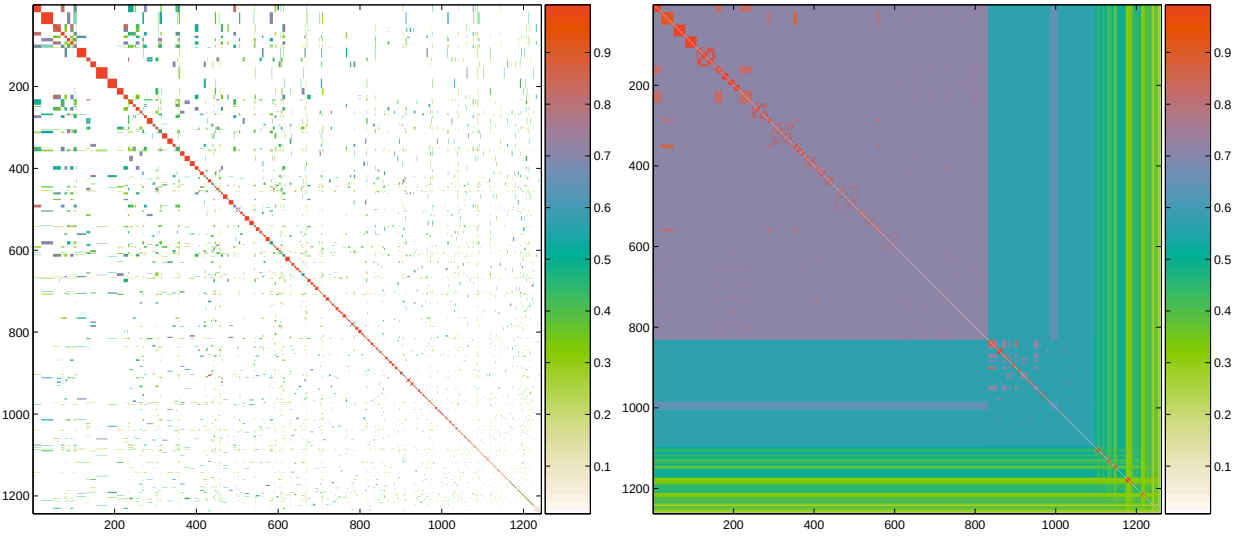

Figure 1: Weights and cliques in the weighted protein-protein interaction network. Left: original graph. Right: transitive closure. Cliques are shown as are small blocks on the diagonal.

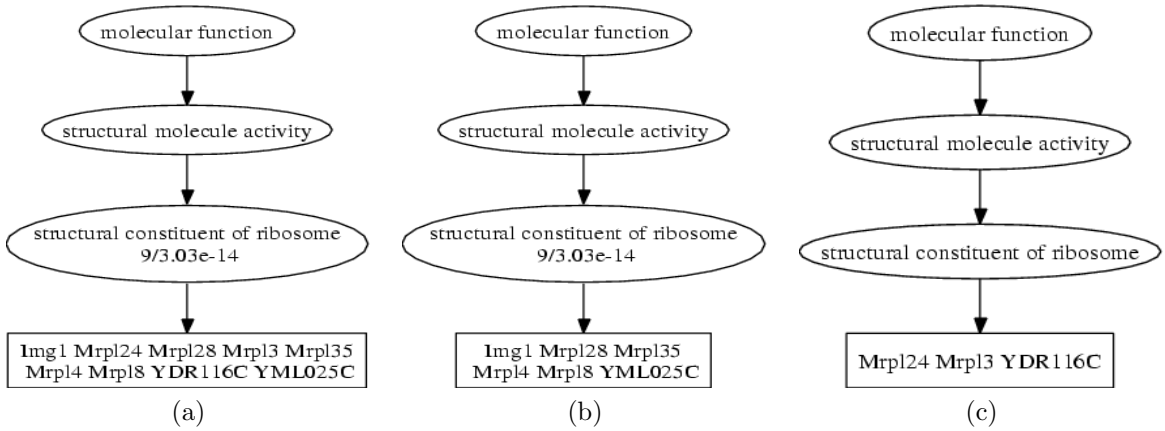

Figure 2: Gene ontology annotations of 3 cliques. (a) The clique obtained from the transitive closure, which is formed by merging the smaller cliques shown in (b) and (c) obtained on the original network.

with rather small sizes. The transitive closure is much more densely connected, thus we get a smaller number of cliques with larger sizes. These cliques also have higher average edge weight within the clique (measured in the original weight W) and lower average strength between different cliques. (Note that weights are normalized to $0 \leq w_{ij} \leq 1$.) These indicate that proteins in the cliques detected based on the transitive closure are more densely connected with each other, meanwhile the average connectivities between different cliques are sparser. Both of these features are desirable for protein module discovery from protein interaction networks.

We list the 10 largest cliques in Table 1. They also have large average weights in the original interaction strength W, thus representing dense regions in the interaction network. Their biological significances have been verified by Gene Ontology analysis, similar to Figure 1.

In Fig. 2, we show an example of the usefulness of transitive closure. Two smaller cliques [shown in Fig. 2 (b) and (c)] in the original interaction network W are merged into one big clique in the transitive

Clique	GO Annotation
Emg1 Imp3 Imp4 Kre31 **Mpp10** Nop14 Sof1 YMR093W YPR144C	sRNA transcript processing
Cus1 Msl1 Prp3 Prp9 Sme1 Smx2 Smx3 Yhc1 **YJR084W**	nuclear mRNA splicing
Fyv4 Mrp1 Mrp10 Mrp13 Mrp17 Mrp21 Mrp4 Mrp51 Mrps9 Nam9 Pet123 Rsm10 Rsm19 Rsm22 Rsm23 Rsm24 Rsm25 Rsm26 Rsm27 Trf4 Ubp10 YDR036C **YGR150C** YMR158W YMR188C YNL306W **YOR205C** YPL013C	protein biosynthesis & metabolism
Atp11 Caf130 Caf40 Ccr4 Cdc36 Cdc39 Fas2 Not3 Not5 Pop2 Sig1 YDR214W	regulation of transcription from RNA polymerase II promoter
Apc1 Apc2 Cdc16 Cdc23 Cdc27 Doc1	cyclin catabolism & ubiquitin-dependent catabolism protein
Sec65 Srp14 Srp21 Srp54 Srp68 Srp72	signal recognition complex
Csl4 Mtr3 Rrp42 Rrp43 Rrp45 Rrp6 Ski6 Ski7	mRNA catabolism
Cft2 Fip1 Pap1 Pfs2 **Pta1** Ref2 Rna14 YGR156W Ysh1	mRNA cleavage and polyadenylylation
Lsm1 Lsm2 Lsm5 Lsm6 Lsm7 Pat1 Prp24 Prp38 Snu23	mRNA metabolism
Apl1 Apl3 Apl5 Apl6 Apm3 Apm4 Aps2 Aps3	vesicle-mediated transport

Table 1: Ten cliques identified by our algorithm. Boldfaced proteins are uncharacterized in GO.

closure W_{TC}. GO annotations show that cliques (b) and (c) have the same function as the merged clique: *structural constituent of ribosome*. This example shows why we obtain fewer cliques on the transitive closure W_{TC}, but the sizes of these cliques are much larger. Since the larger cliques have the same function as the smaller merged ones, the cliques on the transitive closure are biologically more relevant (complete) protein modules.

Note that there are a number of uncharacterized proteins in GO (boldface protein names in Table 1). Since the GO annotation gives a clear biological function for most other proteins in the clique, we infer that these uncharacterized proteins should have similar functions. Thus our approach has the additional benefit of protein annotation.

9 Summary

We study transitive properties of affinity graphs. We show that maximal transitive affinity satisifies the transitive inequality which in turn is proved to be equivalent to the ultra-metric inequality of a distance metric. The maximal transitive affinity defines the transitive closure of the weighted affinity graphs. These concepts can be equivalently extended to dissimilarity graphs. We prove that a modifed Floyd-Warshall algorithm is optimally efficient to compute transitive closures. Furthermore, transitive closure of weighted affinity and dissimilarity graphs are generalized to transitive closure w.r.t. triangle inequalities. They change the edge weights relatively little and thus can apply to wider applications. These new metrics/concepts are parameterless, revealing intrinsic properties of the networks.

We apply transitive closure to protein interaction networks to overcome the sparsity problem. The Motzkin-Straus formalism for identifying cliques is generalized to weighted interaction networks; densely connected protein modules are detected as cliques on the weighted graph and their biological significance are verified by Gene Ontology analysis.

Acknowledgement. Work supported in part by the MAGGIE project, funded by U.S. Department of Energy, Office of Science, under contract DE-AC03-76SF00098.

References

Bader, G. D. and Hogue, C. (2002). Analyzing yeast protein-protein interaction data obtained from different sources. *Nature Biotechnology*, 20:991–997.

Cormen, T., Leiserson, C., Rives, R., and Stein, C. (1998). *Introduction to Algorithms, 2nd Ed.* MIT Press.

Ding, C., He, X., Meraz, R., and Holbrook, S. (2004). A unified representation for multi-protein complex data for modeling protein interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57:99–108.

Ding, C., He, X., and Peng, H. (2005). Finding cliques in protein interaction networks via transitive closure of a weighted graph. *Proc. ACM Int'l Conf Knowledge Disc. Data Mining (KDD) Workshop on Data Mining in Bioinformatics*, pages 70–76.

Gavin, A.-C., Bosche, M., Krause, R., et al. (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *Elements of Statistical Learning*. Springer Verlag.

Ho, Y., Gruhler, A., Heilbut, A., et al. (2002). Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–193.

Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., and Sakaki, Y. (2001). A comprehensive two hybrid analysis to explore the yeast protein interactome. *Proc. Natl. Acad. Sci.*, 98(8):4569–4574.

Motzkin, T. and Straus, E. (1965). Maxima for graphs and a new proof of a theorem of turan. *Canad. J. Math.*, 17:533–540.

Pelillo, M., Siddiqi, K., and Zucker, S. (1999). Matching hierarchical structures using association graphs. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 21:1105 – 1120.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc B.*, 58:267–288.

Uetz., P., Cagney, L., Mansfield, G., et al. (2000). A comprehensive analysis of protein-protein interactions in *saccharomyces cerevisiae*. *Nature*, 403(6770):623–627.

Wasserman, S. and Faust, K. (1994). *Social Network Analysis*. Cambridge University Press.

Watts, D. and Strogatz, S. (1998). Collective dynamics of small world networks. *Nature*, 393:440–442.

Xiong, H., He, X., Ding, C., Zhang, Y., Kumar, V., and Holbrook, S. (2005). Identification of functional modules in protein complexes via hyperclique pattern discovery. *Proc. of Pacific Symposium on Biocomputing*.