

# A Novel Method for Signal Transduction Network Inference from Indirect Experimental Evidence



RÉKA ALBERT,<sup>1</sup> BHASKAR DASGUPTA,<sup>2</sup> RICCARDO DONDI,<sup>3</sup> SEMA KACHALO,<sup>4</sup>  
EDUARDO SONTAG,<sup>5</sup> ALEXANDER ZELIKOVSKY,<sup>6</sup> and KELLY WESTBROOKS<sup>6</sup>

## ABSTRACT

**In this paper, we introduce a new method of combined synthesis and inference of biological signal transduction networks. A main idea of our method lies in representing observed causal relationships as network paths and using techniques from combinatorial optimization to find the sparsest graph consistent with all experimental observations. Our contributions are twofold: (a) We formalize our approach, study its computational complexity and prove new results for exact and approximate solutions of the computationally hard transitive reduction substep of the approach (Sections 2 and 5). (b) We validate the biological usability of our approach by successfully applying it to a previously published signal transduction network by Li et al. (2006) and show that our algorithm for the transitive reduction substep performs well on graphs with a structure similar to those observed in transcriptional regulatory and signal transduction networks.**

**Key words:** combinatorial optimization, signal transduction networks, systems biology.

## 1. INTRODUCTION

**M**OST BIOLOGICAL CHARACTERISTICS of a cell arise from the complex interactions between its numerous constituents such as DNA, RNA, proteins, and small molecules (Alberts, 1994). Cells use signaling pathways and regulatory mechanisms to coordinate multiple functions, allowing them to respond to and acclimate to an ever-changing environment. Genome-wide experimental methods now identify interactions among thousands of proteins (Lee et al., 2002; Giot et al., 2003; Han et al., 2004; Li et al., 2004); however these experiments are rarely conducted in the specific cell type of interest and are not able to probe the directionality of the interactions (i.e., to distinguish between the regulatory source and target). Identification of every reaction and regulatory interaction participating even in a relatively simple

---

<sup>1</sup>Department of Physics, Pennsylvania State University, University Park, Pennsylvania.

<sup>2</sup>Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois.

<sup>3</sup>Dipartimento di Scienze dei Linguaggi della Comunicazione e degli Studi Culturali, Università degli Studi di Bergamo, Bergamo, Italy.

<sup>4</sup>Department of Bioengineering, University of Illinois at Chicago, Chicago, Illinois.

<sup>5</sup>Department of Mathematics, Rutgers University, New Brunswick, New Jersey.

<sup>6</sup>Department of Computer Science, Georgia State University, Atlanta, Georgia.

function of a single-celled organism requires a concerted and decades-long effort. Consequently, the state of the art understanding of many signaling processes is limited to the knowledge of key mediators and of their positive or negative effects on the whole process.

Experimental information about the involvement of a specific component in a given signal transduction network can be partitioned into three categories. First, biochemical evidence that provides information on enzymatic activity or protein-protein interactions. This first category is a *direct interaction*, e.g., binding of two proteins or a transcription factor activating the transcription of a gene or a chemical reaction with a single reactant and single product. Second, pharmacological evidence, in which a chemical is used either to mimic the elimination of a particular component, or to exogenously provide a certain component, leads to observed relationships that are not direct interactions but indirect causal effects most probably resulting from a chain of interactions and reactions. For example, binding of a chemical to a receptor protein starts a cascade of protein-protein interactions and chemical reactions that ultimately results in the transcription of a gene. Observing gene transcription after exogenous application of the chemical allows inferring a causal relationship between the chemical and the gene that however is not a direct interaction. Third, genetic evidence of differential responses to a stimulus in wild-type organisms versus a mutant organism implicates the product of the mutated gene in the signal transduction process. This category is a three-component inference that in a minority of cases could correspond to a single reaction (namely, when the stimulus is the reactant of the reaction, the mutated gene encodes the enzyme catalysing the reaction and the studied output is the product of the reaction), but more often it is indirect. As stated above, the last two types of inference do not give direct interactions but indirect causal relationships that correspond to reachability relationships in the unknown interaction network. Here we describe a method for synthesizing indirect (path-level) information into a consistent network by constructing the sparsest graph that maintains all reachability relationships.

This method's novelty over other network inference approaches is that it does not require expression information, as all reverse engineering approaches do (Carter, 2005). Moreover, our method significantly expands the capability for incorporating indirect (pathway-level) information. Previous methods of synthesizing signal transduction networks (Ma'ayan et al., 2005) only include direct biochemical interactions, and are therefore restricted by the incompleteness of the experimental knowledge on pairwise interactions. Our method is able to incorporate indirect causal effects as network paths with known starting and end vertices and (yet) unknown intermediary vertices.

The first step of our method is to distill experimental conclusions into qualitative regulatory relations between cellular components. Following DasGupta et al. (2006) and Li et al. (2006), we distinguish between positive and negative regulation, usually denoted by the verbs "promote" and "inhibit" and represented graphically as  $\rightarrow$  and  $\neg$ . Biochemical and pharmacological evidence is represented as component-to-component relationships, such as "A promotes B," and is incorporated as a directed arc from A to B. Arcs corresponding to direct interactions are marked as such. Genetic evidence leads to double causal inferences of the type "C promotes the process through which A promotes B." The only way this statement can correspond to a direct interaction is if C is an enzyme catalyzing a reaction in which A is transformed into B. We represent supported enzyme-catalyzed reactions as both A (the substrate) and C (the enzyme) activating B (the product). If the interaction between A and B is direct and C is not a catalyst of the A-B interaction, we assume that C activates A. In all other cases we assume that the three-node indirect inference corresponds to an intersection of two paths ( $A \Rightarrow B$  and  $C \Rightarrow B$ ) in the interaction network; in other words, we assume that C activates an unknown intermediary (pseudo)-vertex of the AB path. The main idea of our method is finding the minimal graph, both in terms of pseudo vertex numbers and non-critical edge numbers, that is consistent with all reachability relationships between real vertices. The algorithms involved are of two kinds: (i) transitive reduction of the resulting graph subject to the constraints that no edges flagged as direct are eliminated and (ii) pseudo-vertex collapse subject to the constraints that real vertices are not eliminated.

Note that we are not claiming that real signal transduction networks are the sparsest possible; our goal is to minimize false positive (spurious) inferences, even if risking false negatives. This means that we want to be as close as possible to a "tree topology" while supporting all experimental observations. The implicit assumption of chain-like or tree-like topologies permeates the traditional molecular biology literature: signal transduction and metabolic pathways were assumed to be close to linear chains, genes were assumed to be regulated by one or two transcription factors (Alberts, 1994). According to current

observations the reality is not far: the average in/out degree of transcriptional regulatory networks (Shen-Orr et al., 2002; Lee et al., 2002) and the mammalian signal transduction network (Ma'ayan et al., 2005) is close to 1. Philosophically, the approach of obtaining the sparsest network can be called as a “parsimony” approach used in the construction of phylogenies and elsewhere.

The rest of the paper is organized as follows:

- In Section 2, we formalize our approach for network synthesis and identify the computational complexities of various steps.
- In Section 5, we provide new algorithmic results on the computationally hard transitive reduction substep of our approach.
- In Section 6.1, we validate the biological usability of our approach by successfully applying it to a previously published signal transduction network by Li et al. (2006).
- In Section 6.2, we show that our algorithm for the transitive reduction substep performs well on graphs with a structure similar to those observed in transcriptional regulatory and signal transduction networks.

## 2. FORMAL DESCRIPTION OF THE NETWORK SYNTHESIS PROCEDURE

The goal of this section is to introduce a formal framework of the network synthesis procedure that is sufficiently general in nature, and amenable to algorithmic analysis and consequent automation. First, we need to describe a graph-theoretic problem which we refer to as the *binary transitive reduction* (BTR) problem. We are given a directed graph  $G = (V, E)$  with an edge labeling function  $w : E \mapsto \{0, 1\}$ . Biologically, edge labels 0 and 1 in edges  $u \xrightarrow{0} v$  and  $u \xrightarrow{1} v$  correspond to the “ $u$  promotes  $v$ ” and “ $u$  inhibits  $v$ ,” respectively.

The following definitions and notations are used throughout the paper:

- All paths are (possibly self-intersecting) directed paths unless otherwise stated. A non-self-intersecting path or cycle is called a *simple* path or cycle.
- If edge labels are removed or not mentioned, they are assumed to be 0 for the purpose of any problem that needs them.
- The *parity* of a path  $P$  from vertex  $u$  to vertex  $v$  is  $\sum_{e \in P} w(e) \pmod{2}$ . A path of parity 0 (resp., 1) is called a path of *even* (resp, *odd*) parity. The same notions carries over to cycles in an obvious manner.
- The notation  $u \xRightarrow{x} v$  denotes a path from  $u$  to  $v$  of parity  $x \in \{0, 1\}$ . If we do not care about the parity, we simply denote the path as  $u \Rightarrow v$ . An edge will simply be denoted by  $u \xrightarrow{x} v$  or  $u \rightarrow v$ .
- For a subset of edges  $E' \subseteq E$ ,  $\text{reachable}(E')$  is the set of all ordered triples  $(u, v, x)$  such that  $u \xRightarrow{x} v$  is a path of the restricted subgraph  $(V, E')$ . We will sometimes simply say  $u \xRightarrow{x} v$  is contained in  $E'$  to mean  $u \xRightarrow{x} v$  is a path of the restricted subgraph  $(V, E')$ .

The BTR problem is defined as follows:

**Instance:** A directed graph  $G = (V, E)$  with an edge labeling function  $w : E \mapsto \{0, 1\}$  and a set of critical edges  $E_{\text{critical}} \subseteq E$ .

**Valid solutions:** A subgraph  $G' = (V, E')$  where  $E_{\text{critical}} \subseteq E' \subseteq E$  and  $\text{reachable}(E') = \text{reachable}(E)$ .

**Objective:** *Minimize*  $|E'|$ .

Note that an exact or an approximate solution to the BTR problem is not unique; alternate solutions represent alternate interpretations of the same data. Intuitively, the BTR problem is useful for determining the sparsest graph consistent with a set of experimental observations. The set of “critical edges” represent edges which are known to be direct interactions with concrete evidence. By maximizing sparseness we do not simply mean to minimize the number of edges per se, but seek to minimize the number of spurious feed-forward loops (i.e., a node regulating another both directly and indirectly). This means that we want to be as close as possible to a “tree topology” while supporting the experimental observations.

We also need to define one more problem that will be used in the formal framework of the network synthesis approach. The pseudo-vertex collapse (PVC) problem is defined as follows:

**Problem name:** Pseudo-Vertex Collapse (PVC)

**Instance:** A directed graph  $G = (V, E)$  with an edge labeling function  $w : E \mapsto \{0, 1\}$  and a subset  $V' \subset V$  of vertices called pseudo-vertices. The vertices in  $V \setminus V'$  are called “real” vertices.

**Definition:**

- For any vertex  $v$ , let  $\text{in}(v) = \{(u, x) \mid u \xrightarrow{x} v, x \in \{0, 1\} \setminus \{v\}\}$  and let  $\text{out}(v) = \{(u, x) \mid v \xrightarrow{x} u, x \in \{0, 1\} \setminus \{v\}\}$ .
- Collapsing two vertices  $u$  and  $v$  is permissible provided both are not “real” vertices and  $\text{in}(u) = \text{in}(v)$  and  $\text{out}(u) = \text{out}(v)$ .
- If permissible, the collapse of two vertices  $u$  and  $v$  creates a new vertex  $w$ , makes every incoming (resp. outgoing) edges to (resp. from) either  $u$  or  $v$  an incoming (resp. outgoing) edge from  $w$ , removes any parallel edge that may result from the collapse operation and also removes both vertices  $u$  and  $v$ .

**Valid solutions:** A graph  $G'' = (V'', E'')$  obtained from  $G$  by a sequence of permissible collapse operations.

**Objective:** Minimize  $|V''|$ .

Intuitively, the PVC problem is useful for reducing the pseudo-vertex set to the the minimal set that maintains the graph consistent with all indirect experimental observations. As in the case of the BTR problem, our goal is to minimize false positive (spurious) inferences of additional components in the network.

A formal framework for the network synthesis procedure is presented in Figure 1. As described in Section 1, in the first step we incorporate biochemical interaction or causal evidence as labeled edges, noting the critical edges corresponding to direct interactions. Then we perform a binary transitive reduction to eliminate spurious inferred edges (i.e., edges that can be explained by paths of the same label). In step two we incorporate double causal relationships  $A \xrightarrow{x} (B \xrightarrow{y} C)$  by (i) adding a new edge  $A \xrightarrow{x} B$  if  $B \xrightarrow{y} C$  is a critical edge, (ii) doing nothing if existing paths in the network already explain the relationship, or (iii) adding a new pseudo-vertex and three new edges. To correctly incorporate the parity of the  $A \xrightarrow{x+y \pmod{2}} C$  relationship, positive  $B \xrightarrow{y} C$  paths will be broken into two positive edges, while negative paths will be broken into a positive edge ( $a = 0$ ) and a negative edge ( $b = 1$ ), summarized in a concise way by the equation  $b = a + b = y \pmod{2}$ . The unnecessary redundancy of the resulting graph is reduced by performing pseudo-vertex collapse, then a second round of binary transitive reduction. Intuitively speaking, the approach in Figure 1 first expands the network by the addition of the pseudo-vertices at the intersection of the two paths corresponding to three-node inferences, then uses the additional information available in the network to collapse these pseudo-vertices, i.e., to identify them with real vertices or with each other. The PVC is the heart of the algorithm, the final BTR is akin to a final cleanup step; thus it is important to perform PVC before BTR in Step 2.2 of Figure 1.

An example of a set of input interactions for a network synthesis approach such as shown in Figure 1 appears in Table 2 in Appendix 1 and the finally constructed network appears in Figure 2b.

It is very easy to add new pseudo-vertices in Step 2.1 using a Floyd-Warshall type transitive closure algorithm (Cormen et al., 2001). Thus the two remaining major steps in the synthesis procedure are in fact the BTR and the PVC problems. It is easy to design a polynomial-time algorithm for the PVC problem.

**Proposition 1.** *The PVC problem can be solved in polynomial time.*

**Proof.** Partition the vertices into equivalence classes such that two vertices are in the same partition provided  $\text{in}(u) = \text{in}(v)$  and  $\text{out}(u) = \text{out}(v)$ . It can be easily seen if two vertices  $u$  and  $v$  in the same partition are collapsed into a new vertex  $w$  then the resulting equivalence partition is same as before except that the two vertices  $u$  and  $v$  are replaced by a new vertex  $w$  in the same equivalence partition. Thus, an optimal solution would consist of collapsing all pseudo-nodes with one arbitrary real-node (if it exists) in each equivalence partition. ■

**F1**

**AU2**

**F2**

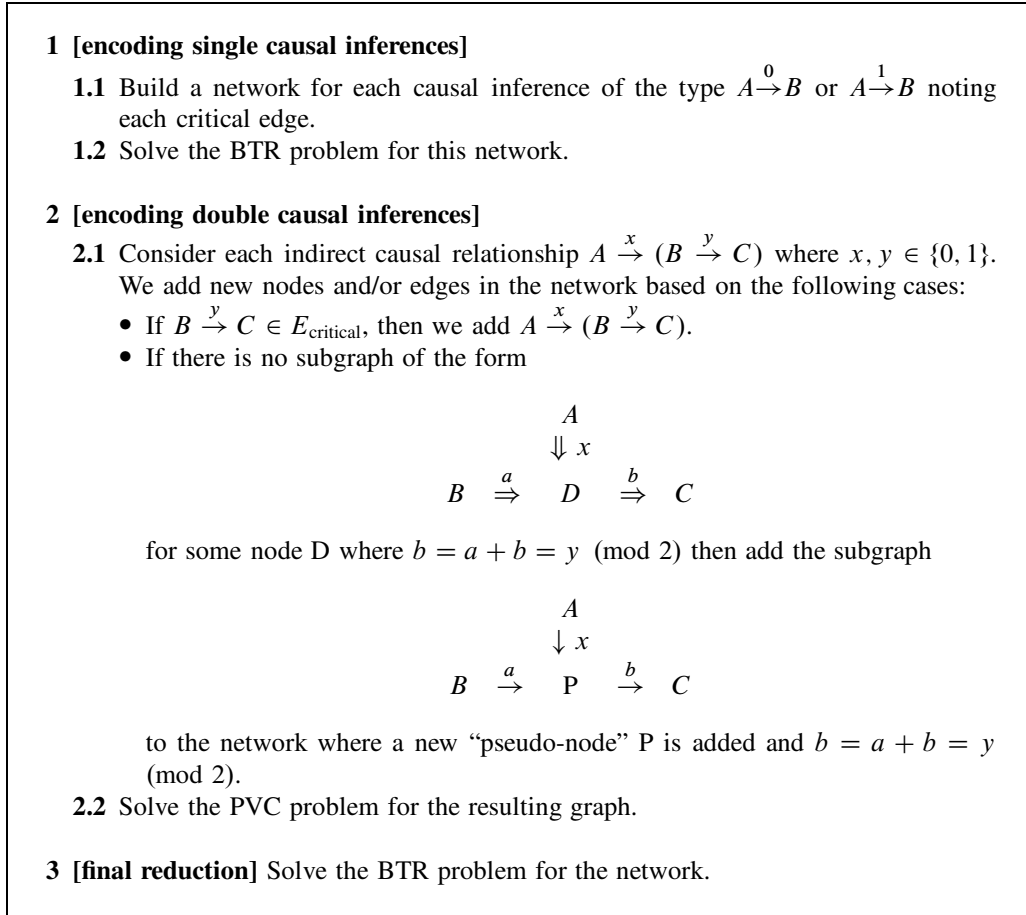


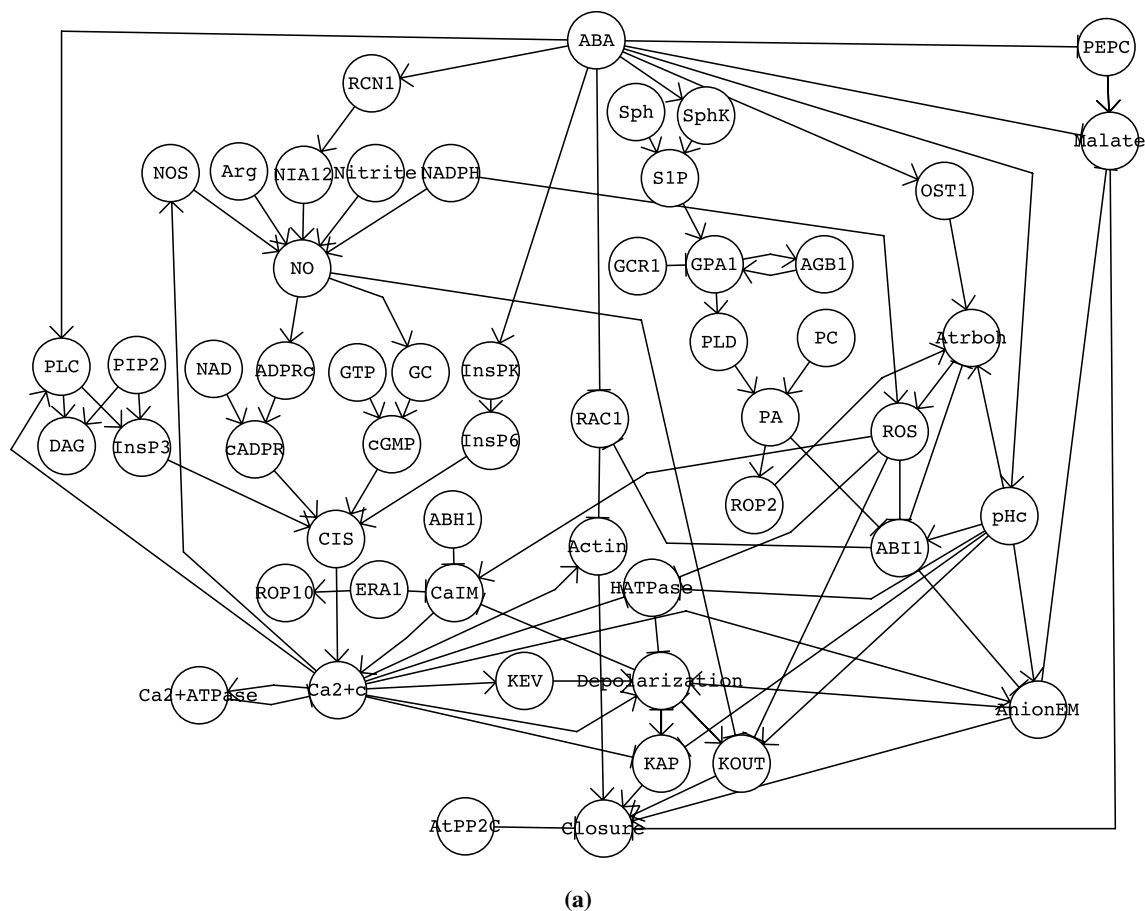
FIG. 1. The overall network synthesis approach.

Thus, we have proved the following proposition.

**Proposition 2.** *All the steps in the network synthesis procedure except the steps that involve BTR can be solved in polynomial time.*

### 3. PREVIOUS RESULTS ON THE BTR PROBLEM

Obviously, BTR is NP-complete since the special case with all-zero edge labels includes the problem of finding a directed Hamiltonian cycle in a graph. If  $E_{\text{critical}} = \emptyset$ , BTR with all-zero edge labels is known as the *minimum equivalent digraph* (MED) problem. MED is known to be MAX-SNP-hard, admits a polynomial time algorithm with an approximation ratio of  $1.617 + \epsilon$  for any constant  $\epsilon > 0$  (Khuller et al., 1995) and can be solved in polynomial time for directed acyclic graphs (Aho et al., 1972). More recently, Vetta (2001) has claimed a  $\frac{3}{2}$ -approximation for the MED problem. A weighted version of the MED problem, in which each edge has a non-negative real weight and the goal is to find a solution with a least value of the sum of weights of the edges in the solution, admits a 2-approximation (Frederickson and JàJà, 1981; Khuller et al., 1999); this implies a 2-approximation for the BTR problem without the restriction  $E_{\text{critical}} = \emptyset$ . In a previous paper (Albert et al., 2007), we have been able to design a  $2 + o(1)$ -approximation for the BTR problem, provided a 1.78-approximation for the BTR problem when all edge labels are zero but critical edges are allowed and observed that the BTR problem can be solved in polynomial time if the input graph is a DAG.



**FIG. 2.** (a) The network manually synthesized by Li et al. (2006) redrawn for easier visual comparison. (b) The network synthesized in this paper. A pseudo-vertex is displayed as  $\otimes$ . (continued)

#### 4. PERTINENT PREVIOUS WORKS ON NETWORK INFERENCE

The idea of transitive reduction, though in a more simplistic setting and/or integrated in an approach different from what appears in this paper, has been used by a few researchers before. For example, Wagner's (2002) goal is to find the network from the *reachability information*. He constructs uniformly random graphs and scale-free networks in a range of connectivities (average degrees), and matches their reachability information to the range of gene reachability information found from yeast perturbation studies. He concludes that the expected number of direct regulatory interactions per gene is around 1 (if the underlying graph is uniformly random) or less than 0.5 (if the underlying graph is scale free with a degree exponent of 2).

Chen et al. (1999) use time-dependent gene expression information to determine candidate activators and inhibitors of each gene, then prune the edges by assuming that no single gene functions both as activator and inhibitor. This assumption is too restrictive given that transcription factors can have both activation and inhibition domains, and the same protein-level interactions (for example phosphorylation by a kinase) can have positive or negative functional character depending on the target.

Li et al. (2006) manually synthesize a plant signal transduction network from indirect (single and double) inferences introducing a first version of pseudo-vertex collapse. They assume that if  $A \xrightarrow{0} B$ ,  $A \xrightarrow{0} C$  and  $C \xrightarrow{0} (A \xrightarrow{0} B)$ , the most parsimonious explanation is that  $A \xrightarrow{0} C \xrightarrow{0} B$ .

The reader is elsewhere (Filkov, 2005; Jong, 2002) for further general information on biological network inference and modeling.

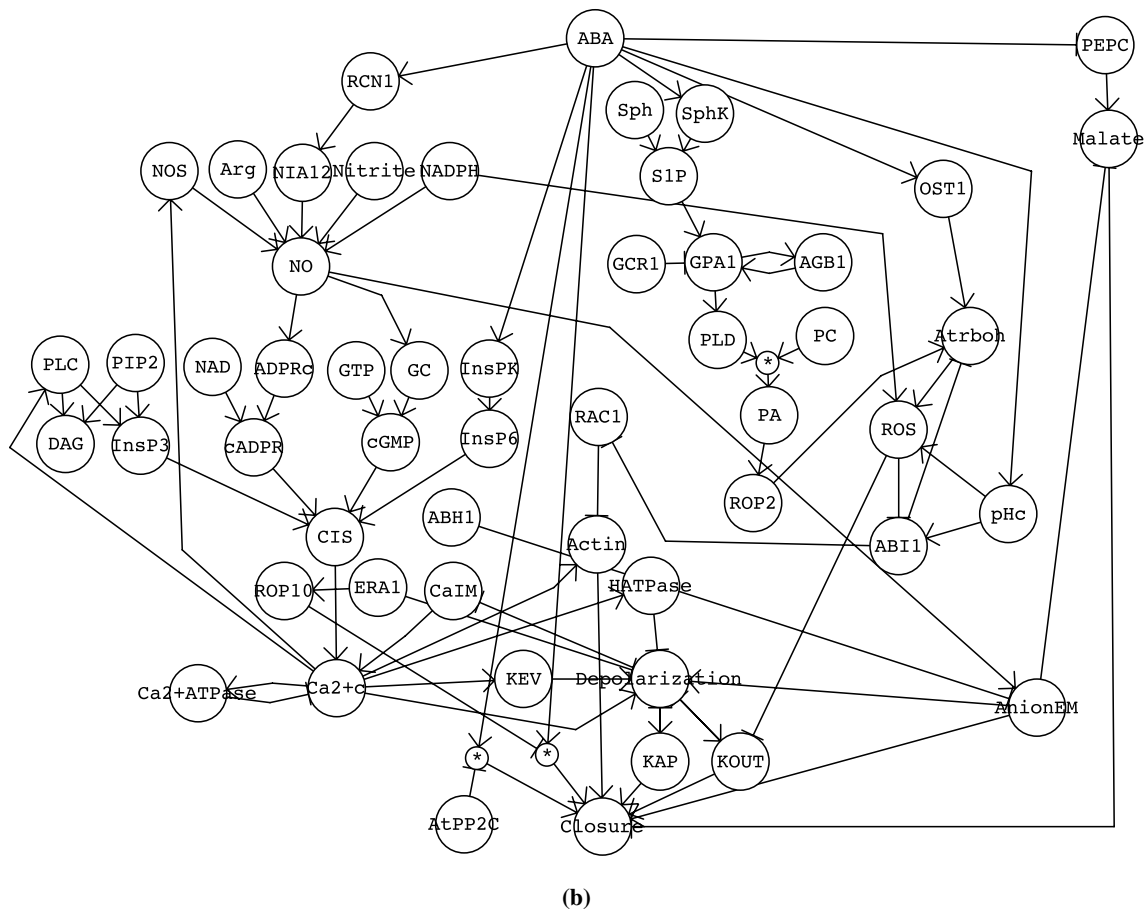


FIG. 2. (Continued)

In our previous publication (Albert et al., 2007), we considered the BTR problem, generalized it to a so-called *p-ary transitive reduction* problem and provided an approximation algorithm for this generalization. The results in Albert et al. (2007) are purely theoretical in nature with no experimental or implementation results, moreover the network synthesis process described in Figure 1 does *not* appear in Albert et al. (2007). All the theoretical results reported in this paper are *disjoint* from the results reported in Albert et al. (2007). A copy of Albert et al. (2007) can be obtained online by following the publications link in the webpage [www.cs.uic.edu/~dasgupta](http://www.cs.uic.edu/~dasgupta) of the second author.

### 5. NEW ALGORITHMIC RESULTS FOR BTR

Our new *theoretical results* on the computational complexity of the BTR problem appear in Sections 5.1–5.3. In Section 5.4, we explain the algorithmic approach that we implemented for BTR to test on real and simulated data.

#### 5.1. Polynomial time algorithm when maximum cycle length is 3

In this section we consider the restriction of the BTR problem where the maximum cycle length of the input graph  $G$  is 3. We denote such restriction by BTR(3). We show that BTR(3) is polynomial time solvable. Observe that BTR(3) with all zero edge labels and no critical edges is already known to be polynomial time solvable (Khuller et al., 1996); the algorithm of Khuller et al. (1996) reduces this special case to bipartite edge cover problem, which is known to be equivalent to maximum bipartite matching (Norman and Rabin, 1959) and thus polynomial time solvable (Hopcroft and Karp, 1973). Due to the

result in Section 7.4 of Albert et al. (2007), we may assume that the input graph  $G$  is a strong connected component since otherwise the problem can be decomposed in polynomial time to computing an optimum solution in each strongly connected component. One can prove the following result; critical steps in the algorithm involve observing that all the critical edges must belong to any solution and that in an optimum solution the edges that are needed to be added must be a minimum edge cover over a certain bipartite graph.

**Theorem 1.** *BTR can be solved in polynomial time if the graph has no cycles of length more than 3.*

Now, we discuss the proof of the above theorem. Let  $G = (V, E)$  be a strongly connected digraph instance of BTR(3). We assume that  $|V| > 3$  and none of the vertices of  $G$  are cut vertices. A *cut vertex* is a vertex whose removal disconnects the underlying undirected graph. By standard techniques the cut vertices can be found in polynomial time, the graph can be partitioned in 2-connected components and the problem can be solved separately in each 2-connected component.

First we will review the approach proposed in Khuller et al. (1996) for BTR(3) with all-zero edge labels and no critical edge. In order to apply this approach to our case, we just do not consider the labels of the edges and the fact that edges can be critical. Let  $G'$  be the graph obtained from  $G$  making the labels of all the edges as zeroes and assuming that all the edges are not critical.

An edge of  $G'$  is called *redundant* if deleting the edge from  $G'$  leaves a strongly connected graph, otherwise it is called *necessary*. Moreover an edge  $(u, v)$  is *unsatisfied* if there is no path from  $v$  to  $u$  consisting of necessary edges. A redundant edge  $e$  provides a cycle for an unsatisfied edge  $(u, v)$  if there is a path from  $v$  to  $u$  consisting of necessary edges and edge  $e$ .

The authors in Khuller et al. (1996) prove the following fact:

**Fact 1.** *Each redundant edge lies on exactly one cycle of  $G'$ .*

From Fact 1 it follows that each redundant edge provides a cycle for at most two unsatisfied edges. Moreover, another fundamental result from Khuller et al. (1996) is the following.

**Fact 2.** *Each cycle in  $G'$  contains at most one redundant edge.*

Hence in order to find a solution for BTR(3) over input graph  $G'$ , we have to find the minimum number of redundant edges that have to be added to necessary edges so that the obtained subgraph is a solution for  $G'$ . Observe that a solution for BTR(3) with all-zero edge labels and no critical edges over input graph  $G'$  consists of the set of necessary edges and a set of redundant edges  $E_r$  such that for each unsatisfied edge  $e$ , a redundant edge providing a cycle for  $e$  is contained in  $E_r$ .

Let  $G'' = (V'', E'')$  be an undirected graph, such that the nodes in  $V''$  are the unsatisfied edges and if a redundant edge provides a cycle for two unsatisfied edges, then we add an edge between the two corresponding nodes. Hence we can state the following lemma:

**Lemma 2 (Khuller et al., 1996).** *The optimum solution for BTR(3) with all-zero edge labels and no critical edges over an input graph  $G'$  consists of the set of necessary edges and of a minimum edge cover of  $G''$ .*

A second result from Khuller et al. (1996) states that  $G''$  is a bipartite graph. Now, since the graph  $G''$  is bipartite, an edge cover of  $G''$  can be computed in polynomial time (Hopcroft and Karp, 1973).

In what follows we will show that, starting from bipartite graph  $G''$ , we build a new bipartite graph  $G^c$ , such that the optimum solution for BTR(3) over input  $G$  can be computed from an edge cover of  $G^c$ .

First, we have to consider that some of the edges of  $G$  are critical. So let  $G_1$  be the graph obtained from  $G$  by ignoring the labels of the edges, i.e., making all the edge labels as zeroes. Observe that  $G_1$  can be obtained from  $G'$  by considering the fact that some of the edges are critical. We will construct an optimum solution  $S$  of  $G_1$  for the BTR(3) problem starting from a solution  $S'$  for  $G'$ . In particular we have to consider the fact the all critical edges of  $G_1$  must be in  $S$ . If a critical edge is necessary in  $G'$ , then surely it will be add to any solution of  $G'$ , thus also in  $S'$ . Hence we have to consider the case when



there exists a set  $E_x$  of critical edges of  $G$  such that each  $e \in E_x$  is classified as redundant. We add the set of edges  $E_x$  to  $S$  and we build the graph  $G^c$  deleting from  $G'$  the edges in  $E_x$  and all the nodes that are endpoints of an edge in  $E_x$ .

Observe that  $G^c$  is a bipartite graph, since it is a subgraph of  $G''$ , which is bipartite. Thus in order to compute an optimum solution  $S$  for the BTR(3) problem over  $G$ , we have to add to the set of redundant and critical edges the minimum set of redundant edges so that the solution obtained cover all the unsatisfied edges. Hence we have to compute a *Minimum Edge Cover* of  $G_c$ , which can be computed in polynomial time. This leads to a minimum solution of BTR(3).

Next we extend the optimum solution  $S$  to an optimum solution  $S_f$  of BTR(3) for  $G$ . Observe that the input to our problem is now the graph  $G$ , i.e., we have to consider the labels of edges. Observe that each optimum solution of BTR(3) for  $G$  must contain at least one cycle of odd parity. Now let  $E_z$  be the set of critical and necessary edges. If at least one cycle of odd parity is contained in the set of edges  $E_z$ , then the solution  $S$  of BTR(3) for  $G_1$  is also a minimum solution of BTR(3) for  $G$ . Thus assume that no cycle of odd parity is included in the solution. By Fact 2 each such cycle contains exactly one redundant edge. Let  $S'$  be an optimum solution of BTR(3) and let  $C$  be an odd cycle contained in  $S'$ . Thus  $C$  consists of two necessary edges and one redundant edge.

Now let  $E_o$  be the set of redundant edges such that, adding  $e \in E_g$  to the set of edges  $E_z$ , the solution contains an odd cycle. For each edge  $e \in E_o$ , we compute a feasible solution  $S_{f(e)}$  as follows. First we add  $E_z$  and  $e$  to  $S_{f(e)}$ . Then we remove edge  $e$  (and the endpoints of  $e$ ) from graph  $G^c$  and we add to  $S_{f(e)}$  a *Minimum Edge Cover* of the resulting graph. The algorithm outputs  $S_f$  the minimal of all the solutions  $S_{f(e)}$ .

**Lemma 3.** *Solution  $S_f$  is a minimum solution for BTR(3).*

**Proof.** Note that any optimum solution must contain one of the edges of  $E_o$ , otherwise no cycle of odd parity is included in the solution. Furthermore, let  $e \in E_o$ , observe that, since  $S_{f(e)}$  is obtained by computing an optimum solution of  $G^c$  after the removal of  $e$ ,  $S_{f(e)}$  is an optimal solution with respect to the solutions that contain edge  $e$ .

Now assume that  $S_f$  contains edge  $e \in E_o$ . Since  $S_f$  is an optimum solution with respect to the solutions that contain edge  $e$ , if there is an optimum solution that contains  $e$ , it follows that  $S_f$  is an optimum solution. Thus, assume that there is no optimum solution containing  $e$ . An optimum solution must contain another edge  $e' \in E_o$ , and since each solution  $S_{f(e')}$  is an optimum solution with respect to the solutions that contain edge  $e$ , the algorithm would have output  $S_{f(e')}$ . ■

This concludes the proof of Theorem 1.

### 5.2. Approximation guarantee of a greedy procedure

Recall that an approximation algorithm for a minimization problem of performance or approximation ratio  $\alpha$  (or simply an  $\alpha$ -approximation) is a polynomial-time algorithm that provides a solution with a value of the objective function that is at most  $\alpha$  times the optimal value of the objective function. In Li et al. (2006), the authors used the following ad-hoc greedy procedure for BTR within the network synthesis procedure to manually create the network for ABA-induced stomatal closure:

**Definition**  
 an edge  $u \xrightarrow{x} v$  is redundant if there is an alternate path  $u \xRightarrow{x} v$   
**GREEDY**  
**while** there exists a redundant edge  
     delete the redundant edge

It is not difficult to see that this greedy procedure for BTR is in fact optimal if the graph is a DAG (Aho et al., 1972; Albert et al., 2007). Below we prove that this simple approach in fact produces a 3-approximation for the BTR problem.

**Theorem 4.** *The GREEDY procedure is a 3-approximation for the BTR problem. Moreover, there are input instances of BTR for which GREEDY has an approximation ratio of at least 2.*

The rest of the section discusses the proof of the above theorem. First, we prove the following.

**Lemma 5.** *The GREEDY procedure is a 3-approximation if the input graph is strongly connected.*

**Proof.** Let  $G = (V, E)$  denote the given input graph and  $OPT(G)$  denotes the number of edges in an optimal solution of BTR for  $G$ . Note that  $OPT(G) \geq |V|$ . For a given graph  $H$ , let  $H^0$  be the graph obtained from  $H$  by setting all edge labels to zeroes and an edge  $e$  in  $H^0$  is called *superfluous* if it would be removed in  $H^0$  by GREEDY but not in  $H$ . Let  $G_{GREEDY}$  be the graph obtained from  $G$  by GREEDY. The proof follows via the following steps:

- (a) We first consider the case when  $E_{critical} = \emptyset$  and show a 2-approximation for this case. The proof of 2-approximation proceeds via the following steps.
  - (i) We first show that  $G_{GREEDY}^0$  contains at most one superfluous edge.
  - (ii) We then show that using (i) one can show that the number of edges in  $G_{GREEDY}$  is at most  $2 \cdot |V| + 1$ .
- (b) We then observe the constraint  $E_{critical} \neq \emptyset$  adds at most one to the approximation ratio.

We first start with the proof of (a)-(i). First we show that a superfluous edge in  $G_{GREEDY}^0$  induces a cycle of odd parity in  $G_{GREEDY}$ . Let  $i \rightarrow j$  be a superfluous edge in  $G_{GREEDY}^0$ . Since  $i \rightarrow j$  is superfluous in  $G_{GREEDY}^0$ , it follows that there is a path  $p_{i,j}$  in  $G_{GREEDY}^0$  from  $i$  to  $j$  that does not contain the edge  $i \rightarrow j$ . Now consider edge  $i \rightarrow j$  and path  $p_{i,j}$  in  $G_{GREEDY}$ . Since the heuristics does not remove edge  $i \rightarrow j$  from  $G_{GREEDY}$ , it implies that  $w(i \rightarrow j) \neq w(p_{i,j})$ . Since  $G_{GREEDY}$  is a strongly connected component, there must be also a path  $q_{j,i}$  from node  $j$  to node  $i$  of parity  $w(q_{j,i})$ . Consider the following two (not necessary simple) cycles: cycle  $\Gamma_1$  consists of the edge  $i \rightarrow j$  and the path  $q_{j,i}$  of parity  $w(\Gamma_1) = w(i \rightarrow j) + w(q_{j,i}) \pmod{2}$ ; cycle  $\Gamma_2$  consists of the path  $p_{i,j}$  and the path  $q_{j,i}$  of parity  $w(\Gamma_2) = w(p_{i,j}) + w(q_{j,i}) \pmod{2}$ .  $w(\Gamma_1) - w(\Gamma_2) = w(i \rightarrow j) - w(p_{i,j}) \pmod{2}$  Since  $w(i \rightarrow j) \neq w(p_{i,j})$ , it follows that  $w(\Gamma_1) - w(\Gamma_2) = w(i \rightarrow j) - w(p_{i,j}) \neq 0 \pmod{2}$  and thus at least one of the two cycles must be of odd parity. Assume without loss of generality that  $\Gamma_1$  is of odd parity.

Now suppose that there exist other superfluous edges in  $G_{GREEDY}^0$ . We will arrive at a contradiction by showing that GREEDY can delete all these superfluous edges except for one in  $G_{GREEDY}$ . Indeed suppose that we delete all the superfluous edges from  $G_{GREEDY}^0$  by applying GREEDY to  $G_{GREEDY}^0$ . Let  $G'$  be the resulting graph; observe that it is a strongly connected graph. Now, let  $G''$  be the strongly connected graph which consists of the edges in  $G'$  and one superfluous edge  $i \rightarrow j$ . This induces a cycle of odd parity in  $G_{GREEDY}$  which implies from every vertex to every other vertex there is a both an even parity path and an odd parity path. Thus, GREEDY will definitely remove all other superfluous edges in  $G_{GREEDY}$ .

Now we show (a)-(ii) that the number of edges in  $G_{GREEDY}$  is at most  $2 \cdot |V| + 1$ . We show this by showing that the graph  $H = (V, E_H)$  obtained by applying GREEDY to  $G_{GREEDY}^0$  has at most  $2 \cdot |V|$  edges. An edge  $u \rightarrow v$  is called a *chord* if  $u$  and  $v$  are two non-adjacent vertices in a path  $u \Rightarrow v$ . Note that  $H$  does not contain a chord. We use a counting method used in the cycle contraction approach in Khuller et al. (1995) to show that  $|E_H| \leq 2 \cdot |V|$ . Contraction of an edge  $u \rightarrow v$  is to merge  $u$  and  $v$  into a single vertex and delete any resulting self-loops or multi-edges. Contracting a cycle is equivalent to contracting the edges of a cycle. Consider the simple procedure of starting with  $H$ , contract an arbitrary cycle of the current graph, and continue in this manner until we have collapsed  $H$  into a single vertex. Note that a cycle contraction cannot produce self-loops or multi-edges since  $H$  has no chords. A contraction of a cycle of  $x$  edges reduces the number of edges by  $x$  and the number of vertices by  $x - 1$ . Thus,  $|E_H| \leq 2 \cdot |V|$ .

Now, we observe (b) by noting that since the edges in  $E_{critical}$  must also appear in any optimal solution, the constraint  $E_{critical} \neq \emptyset$  adds an additional one in the approximation ratio. ■

We now continue with the proof of Theorem 4 by extending the above result to the general case. Let  $G = (V, E)$  be the given graph with  $C_1 = (V_{C_1}, E_{C_1}), C_2 = (V_{C_2}, E_{C_2}), \dots, C_m = (V_{C_m}, E_{C_m})$  being the  $m$  strongly connected components where the  $i$ th component  $C_i$  contains  $n_i$  vertices; thus

- $\cup_{i=1}^m V_{C_i} = V$  and  $\sum_{i=1}^m n_i = |V|$ ;
- $V_{C_i} \cap V_{C_j} = \emptyset$  if  $i \neq j$ ;
- $\cup_{i=1}^m E_{C_i} \subseteq E$ .

First, we recall some pertinent definitions and results from Albert et al. (2007).

**Definition 6 (Albert et al., 2007).** Consider a strongly connected component  $C_i = (V_{C_i}, E_{C_i})$  of the given graph  $G$ .  $C_i$  is called a multiple parity component if for any two vertices  $u, v \in V_{C_i}$ ,  $u \xrightarrow{x} v$  exists in  $C_i$  for every  $x \in \{0, 1\}$  and a single parity component if for any two vertices  $u, v \in V_{C_i}$ ,  $u \xrightarrow{x} v$  exists in  $C_i$  for exactly one  $x$  from  $\{0, 1\}$ .

**Lemma 7 (Albert et al., 2007).**

- (a) Every strongly connected component of  $G$  is either single parity or multiple parity.
- (b) It is possible to design a straightforward dynamic programming approach to determine, given a strongly connected component  $C_i$ , if  $C_i$  is of single or multiple parity using ideas similar to that in the Floyd-Warshall transitive closure algorithm (Cormen et al., 2001). The running time of the algorithm is  $O(|V_{C_i}|^3)$ .

We now recall some results which follow directly from the results in Sections 6 and 7.4 of (Albert et al., 2007).<sup>1</sup> For notational convenience let  $G^0 = (V^0, E^0)$  be a graph identical to the given graph  $G$ ; thus  $V^0 = V$  and  $E^0 = E$ . A doubly-labeled edge  $u \xrightarrow{0,1} v$  is an edge such that traversing the edge gives a path from  $u$  to  $v$  of both parity 0 and parity 1. Let  $G'$  be a new graph obtained from  $G$  by a polynomial-time procedure  $T_{\text{cycle-to-gadget}}$  of the following nature:

**FN1**

- For  $i = 1, 2, \dots, m$  do the following:
  - The starting graph for the  $i$ th iteration is  $G^{i-1}$ .
  - The component  $C_i$  in  $G^{i-1}$  is replaced by a single vertex  $\gamma_i$ .
  - The edge replacement mapping is as follows:
    - \* An edge  $e$  in  $G^{i-1}$  with both end-points not in  $C_i$  stays the same, i.e., the replacement of the edge is the edge itself.
    - \* If  $C_i$  is a multiple parity component then we do the following.
      - For an incoming edge  $u \xrightarrow{x} v$  in  $G^{i-1}$  from a vertex  $u$  not in  $C_i$  to a vertex  $v$  in  $C_i$  the replacement is an edge  $u \xrightarrow{0,1} \gamma_i$ .
      - For an outgoing edge  $u \xrightarrow{x} v$  in  $G^{i-1}$  from a vertex  $u$  in  $C_i$  to a vertex  $v$  not in  $C_i$  the replacement is an edge  $\gamma_i \xrightarrow{0,1} v$ .
    - \* If  $C_i$  is a single parity component then we do the following.
      - For an incoming edge  $u \xrightarrow{x} v$  in  $G^{i-1}$  from a vertex  $u$  not in  $C_i$  to a vertex  $v$  in  $C_i$  the replacement is an edge  $u \xrightarrow{y} \gamma_i$  for some  $y \in \{0, 1\}$ .
      - For an outgoing edge  $u \xrightarrow{x} v$  in  $G^{i-1}$  from a vertex  $u$  in  $C_i$  to a vertex  $v$  not in  $C_i$  the replacement is an edge  $\gamma_i \xrightarrow{y} v$  for some  $y \in \{0, 1\}$ .
    - \* The resultant graph at the end of the  $i$ th iteration is denoted by is  $G^i = (V^i, E^i)$ .
  - Remove identical edges from  $G^m$ . If there are two edges  $u \xrightarrow{x} v$  in  $G^m$ , remove one of them. Let  $G'$  be the resulting graph.

<sup>1</sup>The reader may find the following correspondence between our descriptions and those in Albert et al. (2007) useful.

A “gadget” for a strongly connected component is simply a vertex in our context. A doubly labeled edge  $u \xrightarrow{0,1} v$  in our context is a set of two parallel edges in Albert et al. (2007), one labeled 0 and one labeled 1, such that a solution can contain either both of them or none of them.

Given an optimal solution  $E(m) \subseteq E^m$  of the BTR problem on the DAG  $G' = G^m = (V^m, E^m)$  with  $|E(m)| = OPT'$ , we associate it with a subgraph  $E(0) \subseteq E^0$  of  $G = G^0 = (V^0, E^0)$  via a procedure  $T_{\text{gadget-to-cycle}}$  in the following manner:

- For  $i = m, m-1, \dots, 1$  do the following:
  - Replace the vertex  $\gamma_i$  by the vertices and edges in an 3-approximate solution of  $C_i$  produced by GREEDY on  $C_i$ .
  - Replace an edge  $u \xrightarrow{y} \gamma_i$  incoming to the vertex  $\gamma_i$  by its “corresponding edge”  $u \xrightarrow{x} v$  in  $G^{i-1}$ .
  - Replace an edge  $\gamma_i \xrightarrow{y} v$  outgoing from vertex  $\gamma_i$  by its “corresponding edge”  $u \xrightarrow{x} v$  in  $G^{i-1}$ .
  - The replacement of any other edge is the edge itself.

**Lemma 8 (Albert et al., 2007).**

- (a)  $G'$  is a DAG.
- (b) GREEDY on  $G'$  produces an optimal solution for the BTR problem on  $G'$ .
- (c) To prove that procedure  $T_{\text{cycle-to-gadget}}$  followed by the procedure  $T_{\text{gadget-to-cycle}}$  produces a 3-approximation for  $G$  it suffices to just show the following with the stated assumptions:
  - (i)  $G$  is assumed to contain at least one strongly connected component of either single or multiple parity.
  - (ii)  $T_{\text{cycle-to-gadget}}$  replaces just one arbitrarily strongly connected component, say  $C = (V_C, E_C)$ , of  $G$  to transform  $G$  to  $G' = (V', E')$ . Suppose that  $T_{\text{gadget-to-cycle}}$  transforms an optimal  $E_1 \subseteq E'$  solution of  $G'$  to a solution  $E_2 \subseteq E$  of  $G$ . This procedure satisfy the following invariants:
    - (★) If  $E_1$  is an optimal solution for  $G'$  then  $E_2$  is a valid solution for  $G$ .
    - (★★) A subgraph  $G_2 = (V, E_2)$  that is an optimal solution  $E_2$  for  $G$ , after application of the procedure  $T_{\text{cycle-to-gadget}}$  on the connected component  $C$ , is transformed to a subgraph  $G_1 = (V', E_1)$  that is a valid solution for  $G'$ .

Section 7.4.1 of Albert et al. (2007) show that our edge replacement procedures for a multiple parity component satisfies (★) and (★★).

We now provide exact details of the procedure  $T_{\text{cycle-to-gadget}}$  for a single parity component. Let  $v_C \in V_C$  be any vertex in the single parity component  $C = (V_C, E_C)$ . Let  $\gamma_C$  be the vertex that replaces the component  $C$ . Define the following two notations:

$$[0] = \{v' \in V_C \mid v_C \xrightarrow{0} v' \text{ exists in } C\}$$

$$[1] = \{v' \in V_C \mid v_C \xrightarrow{1} v' \text{ exists in } C\}$$

The edge replacement is as follows:<sup>2</sup>

**FN2**

- For an incoming edge  $u \xrightarrow{x} u'$  in  $G$  from a vertex  $u \notin V_C$  to a vertex  $u' \in [j]$  of  $C$  the replacement is an edge  $u \xrightarrow{x+j \pmod{2}} \gamma_C$ .
- For an outgoing edge  $u' \xrightarrow{x} u$  in  $G$  from a vertex  $u' \in [j]$  of  $C$  to a vertex  $u \notin V_C$  the replacement is an edge  $\gamma_C \xrightarrow{x+j \pmod{2}} u$ .

**Lemma 9 (see Lemma 10 of Albert et al. [2007]).** For any two vertices  $u, u' \in V_C$  with  $u \in [i]$  and  $u' \in [j]$  the path  $u \xrightarrow{j-i \pmod{2}} u'$  exists in  $C$  but the path  $u \xrightarrow{j-i+1 \pmod{2}} u'$  is not in  $C$ .

To verify (★), one must consider the following cases.

<sup>2</sup>This is different from the corresponding edge replacement procedure discussed in Albert et al. (2007).

- (i)  $u \xrightarrow{x} w$  is in  $G$  when  $u \in V_C$  and  $w \notin V_C$ . Suppose that  $u'$  is the last vertex on this path that belongs to  $V_C$ . Thus the path is of the form  $u \xrightarrow{x_1} u' \xrightarrow{x_2} w' \xrightarrow{x_3} w$  with  $x = x_1 + x_2 + x_3 \pmod{2}$ . Suppose that  $u \in [r]$  and  $u' \in [s]$ ; then, by Lemma 9,  $s - r = x_1 \pmod{2}$ . The set of edges  $E_1$  contains the path  $\gamma_C \xrightarrow{x_2+s \pmod{2}} w' \xrightarrow{x_3} w$  since the edge  $\gamma_C \xrightarrow{x_2+s \pmod{2}} w'$  exists in  $G'$ . Suppose that  $T_{\text{gadget-to-cycle}}$  translated this path to a path  $u'' \xrightarrow{x_2+s-t \pmod{2}} w' \xrightarrow{x_3} w$  for some  $u'' \in [t]$ . Then the path  $u \xrightarrow{x_1} u' \xrightarrow{t-s \pmod{2}} u'' \xrightarrow{x_2+s-t \pmod{2}} w' \xrightarrow{x_3} w$  is of parity  $x$ .
- (ii)  $w \xrightarrow{x} u$  is in  $G$  when  $u \in V_C$  and  $w \notin V_C$ . Similar to (i).
- (iii)  $u \xrightarrow{x} w$  is in  $G$  when  $u, w \notin V_C$  but the path contains at least one vertex from  $V_C$ . Let  $u \xrightarrow{x_1} u' \xrightarrow{x_2} v' \xrightarrow{x_3} w$  where  $u'$  and  $v'$  are the first and the last vertices that belong to  $V_C$ . But, then  $u \xrightarrow{x_1} u'$  and  $v' \xrightarrow{x_3} w$  exist in  $E_1$  by (i) and (ii), respectively, and  $u' \xrightarrow{x_2} v'$  exist in  $E_2$  because  $T_{\text{gadget-to-cycle}}$  replaced the vertex  $\gamma_C$  by the vertices and edges in an 3-approximate solution of  $C$ .

Now we turn our attention to the verification of (★★). To verify (★★) one needs to consider the following cases:

- (i)  $\gamma_C \xrightarrow{x} w$  exists in  $G'$ . Using Lemma 9 and the construction of  $T_{\text{cycle-to-gadget}}$  it follows that  $G_2$  contains  $u \xrightarrow{x-j \pmod{2}} w$  for some  $u \in [j] \subseteq V_C$ . Suppose that this path is of the form  $u \xrightarrow{y_1} w' \xrightarrow{y_2} w'' \xrightarrow{x-j-y_1-y_2 \pmod{2}} w$  where  $w'$  is the last vertex on the path that belongs to  $V_C$ . By Lemma 9  $w' \in [j + y_1 \pmod{2}]$ . Thus, the path  $w' \xrightarrow{y_2} w'' \xrightarrow{x-j-y_1-y_2 \pmod{2}} w$  in  $G_2$  translates to the path  $\gamma_C \xrightarrow{j+y_1+y_2 \pmod{2}} w'' \xrightarrow{x-j-y_1-y_2} w$  in  $G_1$  which is a path of parity  $x$  from  $\gamma_C$  to  $w$ .
- (ii)  $w \xrightarrow{x} \gamma_C$  is in  $G'$ . Similar to (i).
- (iii)  $w_1 \xrightarrow{x_1} \gamma_C \xrightarrow{x_2} w_2$  is in  $G'$ .  $w_1 \xrightarrow{x_1} \gamma_C$  and  $\gamma_C \xrightarrow{x_2} w_2$  exist in  $G_1$  by (ii) and (i), respectively.

To complete a proof of 3-approximation for GREEDY, we need to show that  $T_{\text{cycle-to-gadget}}$  and  $T_{\text{gadget-to-cycle}}$  can indeed produce the same sequence of edges for removal as produced by GREEDY.

First, we show that it is sufficient to consider a “canonical” version of GREEDY that considers those edges that belong to the same strongly connected component for removal “consecutively.” By a *valid sequence* of edges for removal for GREEDY we mean a sequence of edges that can be considered by GREEDY in that order for removal.

**Proposition 3.** *Let  $\vec{e} = (e_1, e_2, \dots, e_t)$  be the set of  $t$  edges removed by GREEDY on  $G$ . Let  $e_p$  and  $e_q$  be two edges are in the sequence such that:*

- both end-points of  $e_p$  belong to the same strongly connected component, say  $C_i$ ;
- at most one end-point of  $e_q$  belongs to  $C_i$ .

*Then, exchanging  $e_p$  and  $e_q$  in  $\vec{e}$  produces a a valid sequence of edges for removal for GREEDY.*

**Proof.** Assume that  $p < q$ . Then, we need to show that the removal of edge  $e_q$  has no effect on a subsequent removal of  $e_p$ . The edge  $e_p = u \xrightarrow{x} v$  can be removed by GREEDY because of the existence of an alternate path  $u \xrightarrow{x} v$  that does not involve  $e_p$ . Since both  $u$  and  $v$  belong to  $V_{C_i}$ , the path  $u \xrightarrow{x} v$  does not include a vertex  $w$  not in  $C_i$  as an intermediate vertex. Thus, removal of edge  $e_q$  has no effect on this path.

Otherwise, assume that  $p > q$ . Then, we need to show that the removal of edge  $e_p$  has no effect on a subsequent removal of  $e_q$ . The edge  $e_q = u' \xrightarrow{x} v'$  can be removed by GREEDY because of the existence of an alternate path  $u' \xrightarrow{x} v'$  that does not involved  $e_q$ . Since  $C_i$  is a strongly connected component, the path  $u' \xrightarrow{x} v'$  is of the following form  $u' \xrightarrow{x_1} u'' \xrightarrow{x_2} v'' \xrightarrow{x_3} v'$  with  $x_1 + x_2 + x_3 = x \pmod{2}$  and the path

$u'' \xrightarrow{x_2} v''$  involve vertices from  $V_{C_i}$  only. But, since  $e_p$  was removed because an alternate path of same parity existed, removal of  $e_p$  does not affect the path  $u'' \xrightarrow{x_2} v''$ . ■

Now consider a sequence of edges  $\vec{e}$  that GREEDY selected for removal. By Proposition 3, we may assume that  $\vec{e}$  is of the following form:

$$\left( \underbrace{\hspace{10em}}_{\substack{\text{edges connecting} \\ \text{strong components} \\ \text{(Group 0)}}}, \underbrace{\hspace{10em}}_{\substack{\text{edges in } C_m \\ \text{(Group } m\text{)}}}, \underbrace{\hspace{10em}}_{\substack{\text{edges in } C_{m-1} \\ \text{(Group } m-1\text{)}}}, \dots, \underbrace{\hspace{10em}}_{\substack{\text{edges in } C_1 \\ \text{(Group 1)}}} \right)$$

Every edge  $e$  in Group 0 can first be removed by removing the corresponding edge  $e'$  in  $G'$  that  $T_{\text{cycle-to-gadget}}$  mapped  $e$  to. Because of (★) and (★★) an alternate path in  $G'$  not involving  $e'$  exists. Then, while  $T_{\text{gadget-to-cycle}}$  gradually replaces components by their 3-approximate solutions, replacing component  $C_i$  can be done by removing the edges of Group  $i$  from  $C_i$ .

For the following example input instance GREEDY has an approximation ratio 2. Let the graph  $G$  have a “root” vertex  $r$  and vertices  $x_1, \dots, x_n$ , for each  $x_i$  we have an edge  $x_i \rightarrow r$  and an edge  $r \rightarrow x_i$ , for each  $i$  we have edges  $x_i \rightarrow x_{i+1}$  and all edge labels are zeroes. GREEDY may remove the edges  $x_i \rightarrow x_{i+1}$  for  $i = 1, 2, \dots, n - 1$  thus providing a solution with  $2n$  edges. But an optimal solution with  $n + 1$  edges contains the edge  $r \rightarrow x_1$ , the edges  $x_i \rightarrow x_{i+1}$  for each  $i$  and the edge  $x_n \rightarrow r$ .

This concludes the proof of Theorem 4. The following corollary follows directly from the above proof and will be useful in experimental evaluation of the performance of our implemented algorithms for the BTR problem.

**Corollary 10.** *Let  $G = (V, E)$  be the given graph with  $m$  strongly connected components where the  $i$ th component  $C_i = (V_{C_i}, E_{C_i})$  contains  $n_i = |V_{C_i}|$  vertices. Let  $q_i$  be defined as*

$$q_i = \begin{cases} \max\{n_i, |E_{\text{critical}} \cap E_{C_i}|\} & \text{if } n_i > 1 \\ 0 & \text{otherwise} \end{cases}.$$

*Suppose that GREEDY removed all but  $d$  edges when it was run on  $G'$ . Let  $\mathcal{L} = d + \sum_{i=1}^m q_i$ . Then,  $\text{OPT}(G) \geq \mathcal{L}$ .*

**Proof.** Let  $E$  be an optimal solution of BTR on  $G$  and  $E_i$  be an optimal solution of BTR on  $C_i$ . It is easy to see that (see, for example, Proposition 6 of Albert et al. (2007))  $|E \cap E_{C_i}| = |E_i|$ . If  $n_i > 1$ , then trivially  $|E_i| \geq n_i$  since a directed Hamiltonian cycle is the best possible solution. The optimality of GREEDY on a DAG (and, thus, in particular on the DAG  $G'$ ) ensures that an optimal solution must select at least  $d$  edges that do not belong to  $\cup_{i=1}^m E_{C_i}$ . ■

### 5.3. A mixed ILP formulation for BTR

In theory, the BTR problem can be formulated as a mixed integer programming problem. Details are provided in Appendix 2. Obviously, this approach is not scalable for larger graphs.

### 5.4. Our implementation for the BTR problem

Given an instance graph  $G = (V, E)$  of the BTR problem, it is easy to design a straightforward dynamic programming approach to determine, for every  $u, v \in V$  and every  $x \in \{0, 1\}$ , if  $u \xrightarrow{x} v$  exists in  $G$  using ideas similar to that in the Floyd-Warshall transitive closure algorithm; Albert et al. (2007) provides the details for the sake of completeness. The worst-case running time of the algorithm is  $O(|V|^3)$ . To solve the BTR problem within a acceptable time complexity while ensuring a good accuracy, we have implemented the following two major approaches:

**Approach 1 (applicable for smaller graphs).** If the number of nodes in the graph is at most a threshold  $N$ , we implemented the GREEDY heuristic of Section 5.2 on the *entire graph*. The heuristic is implemented by iteratively selecting a new non-critical edge  $e = u \overset{x}{\rightarrow} v$  for removal, tentatively removing it from  $G$  and checking if the resulting graph has a path  $u \overset{x}{\Rightarrow} v$ . If so, we remove the edge; otherwise, we keep it and mark it so that we never select it again. We stop when we have no more edges to select for deletion.

**Approach 2 (applicable for larger graphs).** If the number of nodes in the graph is above the threshold  $N$ , we first use Approach 1 for every strongly connected component of  $G$ . Then we use the procedures  $T_{\text{cycle-to-gadget}}$  and  $T_{\text{gadget-to-cycle}}$  to identify the remaining edges that can be deleted.

To speed up our implementations and/or to improve accuracy, we also use some rather obvious algorithmic engineering approaches, such as:

- Stop the Floyd-Warshall iteration in Approach 1 as soon as a path  $u \overset{x}{\Rightarrow} v$  is discovered.
- Randomize the selection of the next edge for removal.
- In Approach 2, if the strongly connected component has very few vertices, calculate an exact solution of BTR on this component exhaustively.

Both Approach 1 and Approach 2 are guaranteed to be a 3-approximate solution by Theorem 4. However, in Approach 1 there is no bias towards a particular candidate edge for removal among all candidate edges; in contrast, in Approach 2 a bias is introduced via removal of duplicate edges in the gadget replacement procedure. Thus, the two approaches may return slightly different solutions in practice. Choosing  $N$  to be 150, *our implementation takes mostly negligible time* to run on networks with up to thousands of nodes, taking time of the order of seconds for the manually curated network that is described in Section 6.1 to about a minute for the 1000 node random biological networks described in Section 6.2 on which we tested the performance of our implementations. Theoretical worst-case estimates of the running times of the two approaches are as follows. Approach 1 runs in  $O(d \cdot |V|^3)$  time where  $d$  is the number of non-critical edges. By using a linear-time solution of the BTR problem on a DAG (see the algorithm described in Lemma 2 of Albert et al. (2007)), Approach 2 runs in  $O(m^2 + |E| + \sum_{i=1}^m d_i \cdot n_i^3)$  time where the given graph has  $m$  strongly connected components and  $d_i$  and  $n_i$  are the number of non-critical edges and vertices in the  $i$ th strongly connected component, respectively.

## 6. VERIFICATION OF THE METHODS

### 6.1. Synthesizing a network for ABA-induced stomatal closure

Here we discuss our computational results on synthesizing experimental results into a consistent guard cell signal transduction network for ABA-induced stomatal closure using our detailed procedure described in Section 2 and compare it with the manually curated network obtained in Li et al. (2006). Our starting point is the list of experimentally observed causal relationships in ABA-induced closure collected and published as Table S1 in Li et al. (2006). This table contains around 140 interactions and causal inferences, both of type “A promotes B” and “C promotes process(A promotes B).” We augment this list with critical edges drawn from biophysical/biochemical knowledge on enzymatic reactions and ion flows and with simplifying hypotheses made by Li et al., both described in Text S1 of Li et al. (2006); the complete list of causal relationships is given in our Table 2 in Appendix 1.

The synthesis of the network is carried out using the formal method described in Section 2. We also formalize an additional rule specific to the context of this network (and implicitly assumed by Li et al. [2006]) regarding enzyme-catalyzed reactions. We follow Li et al. in representing each of these reactions by two directed critical edges, one from the reaction substrate to the product and one from the enzyme to the product. As the reactants (substrates) of the reactions in Li et al. (2006) are abundant, the only way to regulate the product is by regulating the enzyme. The enzyme, being a catalyst, is always promoting the product’s synthesis, thus positive indirect regulation of a product will be interpreted as positive regulation of the enzyme, and negative indirect regulation of the product will be interpreted as negative regulation of the

enzyme. In graph-theoretic terms, this leads to the following rule. We have a subset  $E_{\text{enzymatic}} \subseteq E_{\text{critical}}$  of edges that are all labeled 0. Suppose that we have a path  $A \xrightarrow{a} x \xrightarrow{b} B$ , an edge  $C \xrightarrow{0} B \in E_{\text{enzymatic}}$ . Then, we identify the node  $C$  with  $x$  by collapsing them together and set the parities of the edges  $A \rightarrow (x = C)$  and  $(x = C) \rightarrow B$  based on the following two cases:

- if  $a + b = 0 \pmod{2}$  then both  $A \rightarrow (x = C)$  and  $(x = C) \rightarrow B$  have zero parities.
- if  $a + b = 1 \pmod{2}$  then  $A \rightarrow (x = C)$  has parity 1 and  $(x = C) \rightarrow B$  has parity 0.

The manually synthesized network of Li et al. includes a pseudo-vertex for each non-critical edge, indicating the existence of unknown biological mediators. For the ease of comparison we omit these degree two pseudo-vertices. The two networks are shown in Figure 2a,b. Here is a brief summary of an overall comparison of the two networks:

- Li et al.'s (2006) network has 54 vertices and 92 edges; our network has 57 vertices (3 extra pseudo-vertices) but only 84 edges.
- Both Li et al.'s (2006) and our network have identical strongly connected component (SCC) of vertices. There is one SCC of size 18 (KOUT Depolarization KAP CaIM Ca2+c Ca2+ATPase HATPase KEV PLC InsP3 NOS NO GC cGMP ADPRc cADPR CIS AnionEM), one SCC of size 3 (Atrboh ROS ABI1), one SCC of size 2 (GPA1 AGB1) and the rest of the SCCs are of size 1 each.
- All the paths present in the Li et al. (2006) reconstruction are present in our network as well. Our network has the extra path  $\text{ROP10} \xrightarrow{1} \text{Closure}$  that Li et al. cited in their Table S1 but did not include in their network due to weak supporting evidence.
- The two networks have 71 common edges.

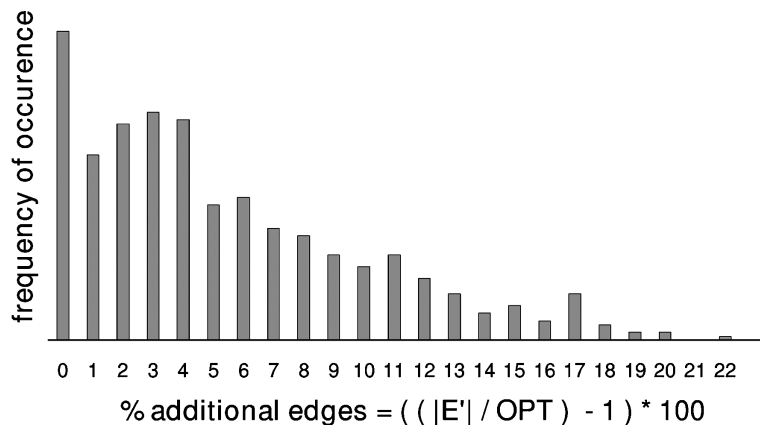
Thus, the two networks are highly similar but diverge on a number of edges. Li et al. keep a few graph-theoretically redundant edges such as  $\text{ABA} \xrightarrow{0} \text{PLC}$ ,  $\text{PA} \xrightarrow{1} \text{ABI1}$  and  $\text{ROS} \xrightarrow{0} \text{CaIM}$  that would be explainable by feedback processes. Some of our edges such as  $\text{NO} \xrightarrow{0} \text{AnionEM}$  correspond to paths in Li et al.'s reconstruction. Our graph contains the full pseudo-vertex-using representation of the process  $\text{AtPP2C} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$  that Li et al. simplifies to  $\text{AtPP2C} \xrightarrow{1} \text{ABA}$ . We have  $\text{pHc} \xrightarrow{0} \text{ROS}$  and  $\text{ROS} \xrightarrow{0} \text{Atrboh}$  where Li et al. (2006) has  $\text{pH} \xrightarrow{0} \text{Atrboh}$  and a positive feedback loop on Atrboh. All these discrepancies are due not to algorithmic deficiencies but to human decisions.

Finally, the entire network synthesis process was done within a few seconds by our implemented algorithms.

## 6.2. Performance of our solutions for BTR on simulated networks

A variety of cellular interaction and regulatory networks have been mapped and graph theoretically characterized. One of the most frequently reported graph measures is the distribution of node degrees, i.e., the distribution of the number of incoming or outgoing edges per node. A variety of networks, including many cellular interaction networks, are heterogeneous (diverse) in terms of node degrees and exhibit a degree distribution that is close to a power-law or a mixture of a power law and an exponential distribution (Jeong et al., 2000; Albert and Barabási, 2002; Giot et al., 2003; Li et al., 2004; Ma'ayan et al., 2005). Transcriptional regulatory networks exhibit a power-law out-degree distribution, while the in-degree distribution is more restricted (Shen-Orr et al., 2002; Lee et al., 2002). To test our algorithm on a network similar to the observed features, we generate random networks with a prescribed degree distribution using the methods in Newman et al. (2001). We base the degree distributions on the yeast transcriptional regulatory network that has a maximum out-degree of  $\sim 150$  and maximum in-degree of  $\sim 15$  (Lee et al., 2002). In our generated network the distribution of in-degree of the network is *exponential*, i.e.,  $\text{Pr}[\text{in-degree} = x] = L e^{-Lx}$  with  $L$  between  $1/2$  and  $1/3$  and the maximum in-degree is 12. The distribution of out-degree of the network is governed by a power-law, i.e., for  $x \geq 1$   $\text{Pr}[\text{out-degree} = x] = cx^{-c}$  and for  $x = 0$   $\text{Pr}[\text{out-degree} = 0] \geq c$  with  $c$  between 2 and 3 and the maximum out-degree is 200. We varied the ratio of excitatory to inhibitory edges between 2 and 4. Since there are no known biological estimates





**FIG. 3.** A plot of the empirical performance of our BTR algorithm on the 561 simulated interaction networks.  $E'$  is our solution,  $OPT$  is the trivial lower bound on the minimum number of edges described in Equation (1) and  $100 \times \left( \frac{|E'|}{OPT} - 1 \right)$  is the percentage of additional edges that our algorithm keeps. On an average, we use about 5.5% more edges than the optimum (with about 4.8% as the standard deviation).

of critical edges<sup>3</sup> we tried a few small and large values, such as 1%, 2%, and 50%, for the percentage of edges that are critical to catch qualitatively all regions of dynamics of the network that are of interest. FN3

To empirically test the performance of our algorithm, we used the following (rather loose) lower bound  $OPT$  for the optimal solution

$$OPT = \max\{n + s - c, t, \mathcal{L}\} \quad (1)$$

where  $n$  is the number of vertices,  $s$  is the number of strongly connected components,  $c$  is the number of connected components of the underlying undirected graph,  $t$  is the number of those edges  $u \xrightarrow{x} v$  such that either  $u \xrightarrow{x} v \in E_{\text{critical}}$  or there is no alternate path  $u \xrightarrow{x} v$  in the graph and  $\mathcal{L}$  is the lower bound that was mentioned in Corollary 10.

We tested the performance of our BTR algorithm on 561 randomly generated networks varying the number of vertices between roughly 100 and 900. A summary of the performance is shown in Figure 3, indicating that our transitive reduction procedure returns solutions close to optimal in many cases even with such a simple lower bound of  $OPT$ . The running time of BTR on an individual network is negligible (from about one second for a 100 node networks to about no more than a minute for a 1000 node network). A summary of the various statistics of these 561 networks is given in Table 1. More meticulous details about the performance of our algorithm for BTR together with the characteristics of the random networks (spanning over 24 pages and thus not suitable for a direct inclusion as an appendix) are available as a table from the website [www.cs.uic.edu/~dasgupta/network-synthesis/](http://www.cs.uic.edu/~dasgupta/network-synthesis/). To verify the performance of our BTR algorithm, we perturb most of these networks with increasing amounts of additional random edges chosen such they do not change the optimal solution of the original graph. The subcolumn in the table in the above-mentioned website under each random addition of edges shows that average number of edges after reduction over 100 runs. In many cases, our algorithm returns a solution that is very close to the original network on which additional edges are added. F3 T1

---

<sup>3</sup>By “estimates of critical edges,” we mean an accurate estimate of the percentage of total edges that are critical on an average in a biological network. Depending on the experimental or inference methods, different network reconstructions have *widely varying* expected fractions of critical edges. For example, the curated network of Ma’ayan et al. (2005) is expected to have close to 100% critical edges as they specifically focused on collecting direct interactions only. Protein interaction networks are expected to be mostly critical (Giot et al., 2003; Han et al., 2004; Li et al., 2004). The so-called genetic interactions (e.g., synthetic lethal interactions) represent compensatory relationships, and only a minority of them are direct interactions. Network inference (reverse engineering) approaches lead to networks whose interactions are close to 0% critical.

TABLE 1. BASIC STATISTICS OF THE SIMULATED NETWORKS USED IN FIGURE 3

Number of nodes (range)	Average number of edges			
	Total	Excitory	Inhibitory	Critical
98–100	206	147	59	31
250–282	690	552	138	33
882–907	2489	1991	498	118

## 7. DISCUSSION

The comparison of our method with previous work enables us to conclude that our methodology serves as a very important first step in formalizing the logical substrate of an inferred signal transduction network. We foresee its optimal application in conjunction with human expertise, as part of an interactive and iterative process. The user of the algorithm would give the experimentally known information as input, then use the resulting network to augment the input information with additional facts or hypotheses through several rounds of iterations. This will allow biologists to simultaneously synthesize their knowledge and formalize their hypotheses regarding a signal transduction network. On the theoretical side, we conjecture that the GREEDY procedure produces a  $2 + o(1)$ -approximation but have been unable to prove it.

A preliminary version of implementations of the network synthesis procedure is available from [www.cs.uic.edu/~dasgupta/network-synthesis/](http://www.cs.uic.edu/~dasgupta/network-synthesis/). We eventually plan to refine the algorithms further, include more help files in the webpage on how to use the software, and make the source codes available as well.

## 8. APPENDIX 1

TABLE 2. REGULATORY INTERACTIONS BETWEEN ABA SIGNAL TRANSDUCTION PATHWAY COMPONENTS

Interaction	Critical	Enzymatic	Interaction	Critical	Enzymatic
$\text{ABA} \xrightarrow{0} \text{SphK}$	No	No	$\text{ABA} \xrightarrow{0} \text{OST1}$	No	No
$\text{ABA} \xrightarrow{0} \text{CaIM}$	No	No	$\text{ABA} \xrightarrow{0} \text{InsP6}$	No	No
$\text{ABA} \xrightarrow{0} \text{Ca2+c}$	No	No	$\text{ABA} \xrightarrow{0} \text{NO}$	No	No
$\text{ABA} \xrightarrow{0} \text{InsP3}$	No	No	$\text{ABA} \xrightarrow{0} \text{AnionEM}$	No	No
$\text{ABA} \xrightarrow{1} \text{PEPC}$	No	No	$\text{ABA} \xrightarrow{1} \text{Malate}$	No	No
$\text{ABA} \xrightarrow{1} \text{HATPase}$	No	No	$\text{ABA} \xrightarrow{1} \text{RAC1}$	No	No
$\text{ABA} \xrightarrow{0} \text{PLD}$	No	No	$\text{ABA} \xrightarrow{0} \text{ROS}$	No	No
$\text{Ca2+c} \xrightarrow{1} \text{CaIM}$	No	No	$\text{Ca2+c} \xrightarrow{0} \text{KEV}$	No	No
$\text{Ca2+c} \xrightarrow{0} \text{AnionEM}$	No	No	$\text{InsP6} \xrightarrow{0} \text{Ca2+c}$	No	No
$\text{InsP6} \xrightarrow{0} \text{CIS}$	No	No	$\text{ROS} \xrightarrow{0} \text{CaIM}$	No	No
$\text{ROS} \xrightarrow{0} \text{Closure}$	No	No	$\text{ROS} \xrightarrow{1} \text{ABI1}$	No	No
$\text{ROS} \xrightarrow{1} \text{KOUT}$	No	No	$\text{pHc} \xrightarrow{0} \text{KOUT}$	No	No
$\text{pHc} \xrightarrow{0} \text{ABI1}$	No	No	$\text{pHc} \xrightarrow{0} \text{ROS}$	No	No
$\text{pHc} \xrightarrow{0} \text{HATPase}$	No	No	$\text{PA} \xrightarrow{1} \text{ABI1}$	No	No
$\text{PA} \xrightarrow{0} \text{Closure}$	No	No	$\text{PA} \xrightarrow{0} \text{ROS}$	No	No
$\text{NO} \xrightarrow{0} \text{Closure}$	No	No	$\text{NO} \xrightarrow{0} \text{AnionEM}$	No	No

(continued)

TABLE 2. (Continued)

<i>Interaction</i>	<i>Critical</i>	<i>Enzymatic</i>	<i>Interaction</i>	<i>Critical</i>	<i>Enzymatic</i>
NO $\xrightarrow{1}$ KOUT	No	No	RAC1 $\xrightarrow{1}$ Actin	No	No
RAC1 $\xrightarrow{1}$ Closure	No	No	ABH1 $\xrightarrow{1}$ AnionEM	No	No
AnionEM $\xrightarrow{1}$ Malate	No	No	ERA1 $\xrightarrow{0}$ ROP10	No	No
Depolarization $\xrightarrow{1}$ Ca2+c	No	No	GPA1 $\xrightarrow{0}$ PLD	Yes	No
Sph $\xrightarrow{0}$ S1P	Yes	No	InsPK $\xrightarrow{0}$ InsP6	Yes	Yes
PLC $\xrightarrow{0}$ DAG	Yes	Yes	PIP2 $\xrightarrow{0}$ DAG	Yes	No
PLC $\xrightarrow{0}$ InsP3	Yes	Yes	PIP2 $\xrightarrow{0}$ InsP3	Yes	No
GC $\xrightarrow{0}$ cGMP	Yes	Yes	GTP $\xrightarrow{0}$ cGMP	Yes	No
ADPRc $\xrightarrow{0}$ cADPR	Yes	Yes	NAD $\xrightarrow{0}$ cADPR	Yes	No
NADPH $\xrightarrow{0}$ NO	Yes	No	Nitrite $\xrightarrow{0}$ NO	Yes	No
Arg $\xrightarrow{0}$ NO	Yes	No	NOS $\xrightarrow{0}$ NO	Yes	Yes
NIA12 $\xrightarrow{0}$ NO	Yes	Yes	NADPH $\xrightarrow{0}$ ROS	Yes	No
Atrboh $\xrightarrow{0}$ ROS	Yes	Yes	Ca2+ATPase $\xrightarrow{1}$ Ca2+c	Yes	No
Ca2+c $\xrightarrow{0}$ Ca2+ATPase	Yes	No	HATPase $\xrightarrow{1}$ Depolarization	Yes	No
KOUT $\xrightarrow{1}$ Depolarization	Yes	No	KAP $\xrightarrow{1}$ Depolarization	Yes	No
AnionEM $\xrightarrow{0}$ Depolarization	Yes	No	Ca2+c $\xrightarrow{0}$ Depolarization	Yes	No
KEV $\xrightarrow{0}$ Depolarization	Yes	No	RCN1 $\xrightarrow{0}$ NIA12	No	No
CIS $\xrightarrow{0}$ Ca2+c	Yes	No	CaIM $\xrightarrow{0}$ Ca2+c	Yes	No
Malate $\xrightarrow{1}$ Closure	Yes	No	GCR1 $\xrightarrow{1}$ GPA1	Yes	No
ABA $\xrightarrow{0}$ RCN1	No	No	AnionEM $\xrightarrow{0}$ Closure	Yes	No
KAP $\xrightarrow{0}$ Closure	Yes	No	KOUT $\xrightarrow{0}$ Closure	Yes	No
ERA1 $\xrightarrow{1}$ CaIM	No	No	ABH1 $\xrightarrow{1}$ CaIM	No	No
cGMP $\xrightarrow{0}$ CIS	No	No	cADPR $\xrightarrow{0}$ CIS	No	No
InsP3 $\xrightarrow{0}$ CIS	No	No	Ca2+c $\xrightarrow{0}$ NOS	No	No
ROS $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	AnionEM $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—
PLC $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	SphK $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—
SphK $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ AnionEM)	—	—	SphK $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ S1P)	—	—
S1P $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	GPA1 $\xrightarrow{0}$ (S1P $\xrightarrow{0}$ AnionEM)	—	—
GPA1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ ROS)	—	—	GCR1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure)	—	—
PLC $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Ca2+c)	—	—	cADPR $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Ca2+c)	—	—
NOS $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	NO $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—
NO $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	NO $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ AnionEM)	—	—
Ca2+c $\xrightarrow{0}$ (NO $\xrightarrow{0}$ AnionEM)	—	—	NO $\xrightarrow{0}$ (Ca2+c $\xrightarrow{0}$ CIS)	—	—
ADPRc $\xrightarrow{0}$ (NO $\xrightarrow{0}$ Ca2+c)	—	—	GC $\xrightarrow{0}$ (NO $\xrightarrow{0}$ Ca2+c)	—	—
KOUT $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	GPA1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ AnionEM)	—	—
pHc $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	ERA1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ AnionEM)	—	—
ERA1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure)	—	—	ERA1 $\xrightarrow{1}$ (Depolarization $\xrightarrow{0}$ KOUT)	—	—
Atrboh $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure)	—	—	Atrboh $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ ROS)	—	—
Atrboh $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Ca2+c)	—	—	Atrboh $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ CaIM)	—	—

(continued)

TABLE 2. (Continued)

<i>Interaction</i>	<i>Critical</i>	<i>Enzymatic</i>	<i>Interaction</i>	<i>Critical</i>	<i>Enzymatic</i>
$\text{ROS} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{CaIM})$	—	—	$\text{NADPH} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{CaIM})$	—	—
$\text{NAD} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{CaIM})$	—	—	$\text{ERA1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{CaIM})$	—	—
$\text{ERA1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{RCN1} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{RCN1} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{AnionEM})$	—	—	$\text{RCN1} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Ca2+c})$	—	—
$\text{OST1} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{OST1} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{ROS})$	—	—
$\text{PLC} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{Ca2+c} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{AnionEM} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{PLD} \xrightarrow{0} (\text{PC} \xrightarrow{0} \text{PA})$	—	—
$\text{PLD} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{PLC} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{ABA} \xrightarrow{0} (\text{PLD} \xrightarrow{0} \text{PA})$	—	—	$\text{ABA} \xrightarrow{0} (\text{PLD} \xrightarrow{0} \text{PA})$	—	—
$\text{ROP2} \xrightarrow{0} (\text{PA} \xrightarrow{0} \text{ROS})$	—	—	$\text{Actin} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{Ca2+c} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Actin})$	—	—	$\text{RAC1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{ROP10} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{ROS} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{GCR1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{GCR1} \xrightarrow{1} (\text{SIP} \xrightarrow{0} \text{Closure})$	—	—
$\text{cADPR} \xrightarrow{0} (\text{Ca2+c} \xrightarrow{0} \text{CIS})$	—	—	$\text{AnionEM} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—
$\text{CaIM} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{KOUT})$	—	—	$\text{cADPR} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{KOUT})$	—	—
$\text{PLC} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{KOUT})$	—	—	$\text{ROS} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{CaIM})$	—	—
$\text{Ca2+c} \xrightarrow{1} (\text{Depolarization} \xrightarrow{0} \text{KAP})$	—	—	$\text{pHc} \xrightarrow{1} (\text{Depolarization} \xrightarrow{0} \text{KAP})$	—	—
$\text{ABH1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{ABH1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Ca2+c})$	—	—
$\text{ROS} \xrightarrow{0} (\text{ABA} \xrightarrow{1} \text{HATPase})$	—	—	$\text{ABI1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{AnionEM})$	—	—
$\text{ABI1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{ROS})$	—	—	$\text{ABI1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Ca2+c})$	—	—
$\text{AtPP2C} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{Closure})$	—	—	$\text{Ca2+c} \xrightarrow{0} (\text{PLC} \xrightarrow{0} \text{InsP3})$	—	—
$\text{GPA1} \xrightarrow{0} \text{AGB1}$	No	No	$\text{AGB1} \xrightarrow{0} \text{GPA1}$	No	No
$\text{AtPP2C} \xrightarrow{1} \text{Closure}$	No	No	$\text{NO} \xrightarrow{0} \text{ADPRc}$	No	No
$\text{Ca2+c} \xrightarrow{0} \text{HATPase}$	No	No	$\text{ABI1} \xrightarrow{1} \text{Atrboh}$	No	No
$\text{NO} \xrightarrow{0} \text{GC}$	No	No	$\text{ABA} \xrightarrow{0} \text{pHc}$	No	No
$\text{PA} \xrightarrow{0} \text{ROP2}$	No	No	$\text{PEPC} \xrightarrow{0} \text{Malate}$	Yes	Yes
$\text{ABI1} \xrightarrow{1} (\text{ABA} \xrightarrow{0} \text{ROS})$	—	—	$\text{ABA} \xrightarrow{0} \text{PLC}$	No	No
$\text{Depolarization} \xrightarrow{0} \text{KOUT}$	Yes	No	$\text{Depolarization} \xrightarrow{0} \text{KAP}$	Yes	No
$\text{Depolarization} \xrightarrow{1} \text{CaIM}$	Yes	No	$\text{ABI1} \xrightarrow{0} (\text{ABA} \xrightarrow{1} \text{RAC1})$	—	—
$\text{InsPK} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{AnionEM})$	—	—	$\text{InsPK} \xrightarrow{0} (\text{ABA} \xrightarrow{0} \text{InsP6})$	—	—
$\text{SIP} \xrightarrow{0} \text{GPA1}$	No	No			

Data from Li et al. (2006).

## 9. APPENDIX 2

### *Details of a mixed ILP formulation for BTR*

We describe one way to encode the BTR problem as a mixed integer program that makes use of the idea of a flow network between vertices in the problem instance. The reader is referred to a standard textbook such as Cormen et al. (2001) for basic concepts of network flows.

First, we use the following procedure to construct a new graph  $G_1 = (V_1, E_1)$  from the original graph  $G = (V, E)$  which preserves the reachability relationships in the original graph  $G$  while simultaneously eliminating the need for edge labels. For each edge  $e = u \xrightarrow{1} v \in E$ , add  $e$  to  $E_1$  and if  $e \in E_{\text{critical}}$ , then mark  $e$  as a critical edge in  $G_1$ . For each edge  $e = u \xrightarrow{0} v \in E$ , add a new vertex  $w$  to  $G_1$ , add the edges  $e_1 = u \xrightarrow{1} w$  and  $e_2 = w \xrightarrow{1} v$  to  $E_1$  and if  $e \in E_{\text{critical}}$ , then mark both  $e_1$  and  $e_2$  as critical edges in  $E_1$ . Every edge in  $G_1$  has the same label and thus we may disregard the edge labels in  $G_1$ . To find a binary transitive reduction of  $G$ , we will compute the binary transitive reduction of  $G_1$  and map the results back onto  $G$ . *Abusing notations slightly, we use  $E_{\text{critical}}$  to refer to the set of critical edges in  $G_1$ .*

We below describe flow-based mixed ILP for the BTR problem on  $G_1$ . It uses the following variables:

- For every  $e \in E_1$  we introduce the edge variable  $x_e \in \{0, 1\}$  where  $x_e = 0$  (resp.  $x_e = 1$ ) indicates that edge  $e$  is a not member (resp. is a member) of the transitive reduction of  $G_1$ .
- For every  $u, v \in V, e \in E_1$  we introduce flow variables,  $f_{u,v,e}^{\text{even}}$  and  $f_{u,v,e}^{\text{odd}}$ , both taking values in the nonnegative real numbers, called the *even* and *odd* flow variables, respectively.

Note that, for the problem instance  $G = (V, E)$ , the solution space has  $|V|^2 \cdot |E_1| + |E_1|$  dimensions, of which  $|E_1|$  dimensions are discrete, taking values in  $\{0, 1\}$ , while the remaining  $|V|^2 \cdot |E_1|$  dimensions are continuous, taking any nonnegative real value. The mixed integer program which correctly solves the binary transitive reduction program is given below (the notation  $\text{incoming}(x)$  and  $\text{outgoing}(x)$  refer to the sets  $\{u \mid u \rightarrow x \in E_1\}$  and  $\{u \mid x \rightarrow u \in E_1\}$ , respectively):

$$\text{minimize} \quad \sum_{e \in E_1} x_e$$

subject to:

$$x_e = 1 \quad \forall e \in E_1 \cap E_{\text{critical}}$$

$$\sum_{x \in \text{outgoing}(u)} f_{u,v,x}^{\text{even}} = 1 \quad \forall e = u \rightarrow v \in E_1$$

$$\sum_{x \in \text{incoming}(v)} f_{u,v,x}^{\text{even}} - f_{u,v,x}^{\text{odd}} = -1 \quad \forall e = u \rightarrow v \in E_1 \text{ and } w(e) = 0$$

$$\sum_{x \in \text{incoming}(v)} f_{u,v,x}^{\text{even}} - f_{u,v,x}^{\text{odd}} = 1 \quad \forall e = u \rightarrow v \in E_1 \text{ and } w(e) = 1$$

$$\sum_{y \in \text{incoming}(x)} f_{u,v,y}^{\text{even}} - \sum_{y \in \text{outgoing}(x)} f_{u,v,y}^{\text{odd}} = 0 \quad \forall e = u \rightarrow v \in E_1 \text{ and } \forall x \in V_1 - \{u, v\}$$

$$\sum_{y \in \text{incoming}(x)} f_{u,v,y}^{\text{odd}} - \sum_{y \in \text{outgoing}(x)} f_{u,v,y}^{\text{even}} = 0 \quad \forall e = u \rightarrow v \in E_1 \text{ and } \forall x \in V_1 - \{u, v\}$$

$$\sum_{f_{u,v,e}^{\text{even}} + f_{u,v,e}^{\text{odd}}} \leq x_e \quad \forall e \in E_1, u, v \in V$$

The objective ensures that the solution will be the minimum subgraph that satisfies the constraints, and the first constraint in this mixed integer program ensures that the solution will contain every critical edge. The second set of constraints are used to ensure that the solution has the same reachability properties as the original graph.

In the BTR problem, there are two different types of path parities: even and odd. For every edge in the problem instance, the flow constraints in the mixed integer program assume the existence of a flow network in the solution (an even or odd flow network, depending upon if the edge in the original problem instance was labeled with 0 or 1). Each flow variable represents a specific edge relative to some flow and its parity in the original problem instance. The first of the flow constraints states that there is some positive flow coming from the source in the flow network. The second constraint, states that the flow is consumed at the sink in the flow network. The final flow constraint is the flow conservation constraint. The last constraint ensures that when an edge is used by flow it should chosen.

## ACKNOWLEDGMENTS

This reserach was supported by a Sloan Research Fellowship, NSF grants DMI-0537992, MCB-0618402, and USDA grant 2006-35100-17254 (to R.A.); NSF grants IIS-0346973, IIS-0612044, and DBI-0543365 (to B.D.); NSF grant IIS-0346973 (to S.K.); and NSF grant DMS-0614371 (to E.S.).

## REFERENCES

- Aho, A., Garey, M.R., and Ullman, J.D. 1972. The transitive reduction of a directed graph. *SIAM J. Comput.* 1, 131–137.
- Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Rev. Modern Phys.* 74, 47–97.
- Albert, R., DasGupta, B., Dondi, R., et al. 2007. Inferring (biological) signal transduction networks via transitive reductions of directed graphs. *Algorithmica* (in press). **AU3**
- Alberts, B. 1994. *Molecular Biology of the Cell*. Garland, New York.
- Carter, G.W. 2005. Inferring network interactions within a cell. *Briefings Bioinform.* 6, 380–389.
- Chen, T., Filkov, V., and Skiena, S. 1999. Identifying gene regulatory networks from experimental data. *RECOMB 1999*, 94–103.
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., et al. 2001. *Introduction to Algorithms*. MIT Press, Cambridge, MA.
- DasGupta, B., Enciso, G.A., Sontag, E.D., et al. 2006. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Lect. Notes Comput. Sci.* 4007, 253–264.
- Filkov, V. 2005. Identifying gene regulatory networks from gene expression data. In Aluru, S., ed., *Handbook of Computational Molecular Biology*. Chapman & Hall/CRC Press.
- Frederickson, G.N., and JàJà, J. 1981. Approximation algorithms for several graph augmentation problems. *SIAM J. Computing* 10, 270–283.
- Giot, L., Bader, J.S., et al. 2003. A protein interaction map of *Drosophila melanogaster*. *Science* 302, 1727–1736. **AU4**
- Han, J.D., Bertin, N., et al. 2004. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature* 430, 88–93. **AU5**
- Heinrich, R., and Schuster, S. 1996. *The Regulation of Cellular Systems*. Chapman & Hall, New York. **AU6**
- Hopcroft, J.E., and Karp, R.M. 1973. An  $n^2$  algorithm for maximum matching in bipartite graphs. *SIAM J. Comput.* 2, 225–231.
- Jeong, H., Tombor, B., Albert, R., et al. 2000. The large-scale organization of metabolic networks. *Nature* 407, 651–654.
- Jong, H.D. 2002. Modelling and simulation of genetic regulatory systems: a literature review. *J. Comp. Biol.* 9, 67–103.
- Khuller, S., Raghavachari, B., and Young, N. 1995. Approximating the minimum equivalent digraph. *SIAM J. Comput.* 24, 859–872.
- Khuller, S., Raghavachari, B., and Young, N. 1996. On strongly connected digraphs with bounded cycle length. *Discrete Appl. Math.* 69, 281–289.
- Khuller, S., Raghavachari, B., and Zhu, A. 1999. A uniform framework for approximating weighted connectivity problems. *Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms*, 937–938.
- Lee, T.I., Rinaldi, N.J., et al. 2002. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science* 298, 799–804. **AU7**
- Li, S., Armstrong, C.M., et al. 2004. A map of the interactome network of the metazoan *C. elegans*. *Science* 303, 540–543. **AU8**
- Li, S., Assmann, S.M., and Albert, R. 2006. Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling. *PLoS Biol.* 4 (in press). **AU9**
- Ma’ayan, A., Jenkins, S.L., Neves, S., et al. 2005. Formation of regulatory patterns during signal propagation in a mammalian cellular network. *Science* 309, 1078–1083.

- Newman, M.E.J., Strogatz, S.H., and Watts, D.J. 2001. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E* 64, 026118–026134.
- Norman, R.Z., and Rabin, M.O. 1959. An algorithm for a minimum cover of a graph. *Proc. Am. Math. Soc.* 10, 315–319.
- Shen-Orr, S.S., Milo, R., Mangan, S., et al. 2002. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics* 31, 64–68.
- Vetta, A. 2001. Approximating the minimum strongly connected subgraph via a matching lower bound. *Proc. 12th ACM-SIAM Symp. Discrete Algorithms*, 417–426.
- Wagner, A. 2002. Estimating coarse gene network structure from large-scale gene perturbation data. *Genome Res.* 12, 309–315.

Address reprint requests to:  
*Dr. Bhaskar DasGupta*  
*Department of Computer Science*  
*University of Illinois at Chicago*  
*1120 SEO*  
*851 South Morgan St.*  
*Chicago, IL 60607-7053*

*E-mail: dasgupta@cs.uic.edu*

#### **AU1**

**Please review your article as a whole for correctness. Your paper was generated using an electronic file only – no hardcopy manuscript was provided. Please ensure all *artwork, symbols, and special characters* are intact/correct as shown on these proofs and that there were no software interpretation problems. Thank you!**

**Also, please provide short title for running head.**

#### **AU2**

**Tables were renumbered to be in numerical order, is this okay?**

#### **AU3**

**Please provide update if possible.**

#### **AU4**

**Please provide third author's name and initial(s).**

#### **AU5**

**Please provide third author's name and initial(s).**

#### **AU6**

**Please cite in text or delete.**

#### **AU7**

**Please provide third author's name and initial(s).**

#### **AU8**

**Please provide third author's name and initial(s).**

#### **AU9**

**Please provide update if possible.**