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


Réka Albert*
Department of Physics
Pennsylvania State University
University Park, PA 16802
Email: ralbert@phys.psu.edu

Bhaskar DasGupta†
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: dasgupta@cs.uic.edu

Riccardo Dondi
Dipartimento di Scienze dei Linguaggi
della Comunicazione e degli Studi Culturali
Università degli Studi di Bergamo
Bergamo, Italy, 24129
Email: riccardo.dondi@unimib.it

Sema Kachalo‡
Department of Bioengineering
University of Illinois at Chicago
Chicago, IL 60607
Email: sema@uic.edu

Eduardo Sontag§
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903
Email: sontag@math.rutgers.edu

Alexander Zelikovsky
Department of Computer Science
Georgia State University
Atlanta, GA 30303
Email: alexz@cs.gsu.edu

Kelly Westbrooks
Department of Computer Science
Georgia State University
Atlanta, GA 30303
Email: kelly@cs.gsu.edu


January 22, 2007


**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:

- 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);
- we validate the biological usability of our approach by successfully applying it to a previously published signal transduction network by Li et al. [22] 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 (Section 6.1 and 6.2).



**Keywords:** Combinatorial Optimization, Signal Transduction Networks, Systems Biology.



*Partly supported by a Sloan Research Fellowship, NSF grants DMI-0537992, MCB-0618402 and USDA grant 2006-35100-17254.

†Partly supported by NSF grants IIS-0346973, IIS-0612044 and DBI-0543365.

‡Supported by NSF grant IIS-0346973.

§Partly supported by NSF grant DMS-0614371.

# 1 Introduction

Most biological characteristics of a cell arise from the complex interactions between its numerous constituents such as DNA, RNA, proteins and small molecules [3]. 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 [11, 12, 20, 21]; 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 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 exogeneous 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, for a review see [5]). Moreover, our method significantly expands the capability for incorporating indirect (pathway-level) information. Previous methods of synthesizing signal transduction networks [23] 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 [8, 22], we distinguish between positive and negative regulation, usually denoted by the verbs "promote" and "inhibit" and represented graphically as $\rightarrow$ and $\dashv$. 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-catalized 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 [3]. According to current observations the reality is not far: the average in/out degree of transcriptional regulatory networks[20, 26] and the mammalian signal transduction network [23] is close to 1. Philosophically, the approach of obtaining the sparest network can be called as a "parsimony" approach used in the construction of phylogenies and elsewhere.

**The Map.** 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. [22].
- 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 $u \stackrel{x}{\Rightarrow} v$ 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 $u \stackrel{x}{\rightarrow} v$ or $u \rightarrow v$.
- For a subset of edges $E' \subseteq E$, reachable$(E')$ is the set of all ordered triples $(u, v, x)$ such that $u \stackrel{x}{\Rightarrow} v$ is a path of the restricted subgraph $(V, E')$. We will sometimes simply say $u \stackrel{x}{\Rightarrow} v$ is contained in $E'$ to mean $u \stackrel{x}{\Rightarrow} v$ 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 reachable$(E')$ =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 $\text{in}(v) = \{(u, x) \mid u \stackrel{x}{\Rightarrow} v, x \in \{0, 1\}\} \backslash \{v\}$ and let $\text{out}(v) = \{(u, x) \mid v \stackrel{x}{\Rightarrow} u, x \in \{0, 1\}\} \setminus \{v\}$.
> - Collapsing two vertices $u$ and $v$ is permissible provided both are not "real" vertices and in$(u)$ =in$(v)$ and out$(u)$ =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 1 and the finally constructed network appears in Figure 2(b).

It is very easy to add new pseudo-vertices in Step 2.1 using a Floyd-Warshall type transitive closure algorithm [7]. 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 in($u$) =in($v$) and out($u$) =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. ❑

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 $1.617 + \varepsilon$ for *any* constant $\varepsilon > 0$ [17] and can be solved in polynomial time for directed acyclic graphs [1]. More recently, Vetta [27] has claimed a $\frac{3}{2}$-approximation

**1 [encoding single causal inferences]**

**1.1** Build a network for each causal inference of the type $A \xrightarrow{0} B$ or $A \xrightarrow{1} B$ noting each critical edge.

**1.2** Solve the BTR problem for this network.

**2 [encoding double causal inferences]**

**2.1** Consider each indirect causal relationship $A \xrightarrow{x} (B \xrightarrow{y} C)$ where $x, y \in \{0, 1\}$. We add new nodes and/or edges in the network based on the following cases:

- If $B \xrightarrow{y} C \in E_{\text{critical}}$, then we add $A \xrightarrow{x} (B \xrightarrow{y} C)$.
- If there is no subgraph of the form

$$\begin{array}{ccccc} & & A & & \\ & & \Downarrow x & & \\ B & \overset{a}{\Rightarrow} & D & \overset{b}{\Rightarrow} & C \end{array}$$

for some node D where $b = a + b = y \pmod 2$ then add the subgraph

$$\begin{array}{ccccc} & & A & & \\ & & \downarrow x & & \\ B & \overset{a}{\rightarrow} & \mathrm{P} & \overset{b}{\rightarrow} & C \end{array}$$

to the network where a new "pseudo-node" P is added and $b = a + b = y \pmod 2$.

**2.2** Solve the PVC problem for the resulting graph.

**3 [final reduction]** Solve the BTR problem for the network.

Figure 1: The overall network synthesis approach.

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 [10, 19]; this implies a 2-approximation for the BTR problem without the restriction $E_{\text{critical}} = \emptyset$. In a previous paper [4], 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.

## 4 Summary of 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, in [28] Wagner's 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. in [6] 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. in [22] 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 referred to the excellent surveys in [9, 16] for further general information on biological network inference and modelling.

In our previous publication [4], 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 [4] are purely theoretical in nature with no experimental or implementation results, moreover the network synthesis process described in Figure 1 does *not* appear in [4]. All the theoretical results reported in this paper are *disjoint* from the results reported in [4]. A copy of [4] can be obtained online by following the publications link in the webpage `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 below. 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 [18]; the algorithm of [18] reduce this special case to bipartite edge cover problem, which is known to be equivalent to maximum bipartite matching [25] and thus polynomial time solvable [14]. Due to the result in [4, Section 7.4], 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 [18] 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 [18] 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 [18] 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 ([18])** *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 [18] 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 [14].

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 [22], 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 $u \overset{x}{\Rightarrow} v$
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 (*e.g.*, see [1, 4]). 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 $G_{\text{GREEDY}}$ be the graph obtained from $G$ by GREEDY. The proof follows via the following steps.

**(a)** We first consider the case when $E_{\text{critical}} = \emptyset$ and show a 2-approximation for this case. The proof of 2-approximation proceeds via the following steps.

  **(i)** We first show that $G^0_{\text{GREEDY}}$ contains at most one superfluous edge.

  **(ii)** We then show that using **(i)** one can show that the number of edges in $G_{\text{GREEDY}}$ is at most $2 \cdot |V| + 1$.

**(b)** We then observe the constraint $E_{\text{critical}} \neq \emptyset$ 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 $G^0_{\text{GREEDY}}$ induces a cycle of odd parity in $G_{\text{GREEDY}}$. Let $i \rightarrow j$ be a superfluous edge in $G^0_{\text{GREEDY}}$. Since $i \rightarrow j$ is superfluous in $G^0_{\text{GREEDY}}$, it follows that there is a path $p_{i,j}$ in $G^0_{\text{GREEDY}}$ 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 $G_{\text{GREEDY}}$. Since the heuristics does not remove edge $i \rightarrow j$ from $G_{\text{GREEDY}}$, it implies that $w(i \rightarrow j) \neq w(p_{i,j})$. Since $G_{\text{GREEDY}}$ 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 $w(\Gamma_1) = w(i \rightarrow j) + w(q_{j,i})$ (mod 2); cycle $\Gamma_2$ consists of the path $p_{i,j}$ and the path $q_{j,i}$ of parity $w(\Gamma_2) = w(p_{i,j}) + w(q_{j,i})$ (mod 2). $w(\Gamma_1) - w(\Gamma_2) = w(i \rightarrow j) - w(p_{i,j})$ (mod 2) Since $w(i \rightarrow j) \neq w(p_{i,j})$, it follows that $w(\Gamma_1) - w(\Gamma_2) = w(i \rightarrow j) - w(p_{i,j}) \neq 0$ (mod 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 $G^0_{\text{GREEDY}}$. We will arrive at a contradiction by showing that GREEDY can delete all these superfluous edges except for one in $G_{\text{GREEDY}}$. Indeed suppose that we delete all the superfluous edges from $G^0_{\text{GREEDY}}$ by applying GREEDY to $G^0_{\text{GREEDY}}$. 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 $G_{\text{GREEDY}}$ 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 $G_{\text{GREEDY}}$.

Now we show **(a)-(ii)** that the number of edges in $G_{\text{GREEDY}}$ is at most $2 \cdot |V| + 1$. We show this by showing that the graph $H = (V, E_H)$ obtained by applying GREEDY to $G^0_{\text{GREEDY}}$ 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 [17] 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 $E_{\text{critical}}$ must also appear in any optimal solution, the constraint $E_{\text{critical}} \neq \emptyset$ 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 $C_1 = (V_{C_1}, E_{C_1}), C_2 = (V_{C_2}, E_{C_2}), \ldots, C_m = (V_{C_m}, E_{C_m})$ being the $m$ strongly connected components where the $i^{\text{th}}$ 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 [4].

**Definition 6** [4] *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 $u, v \in V_{C_i}$, $u \stackrel{x}{\Rightarrow} v$ exists in $C_i$ for every $x \in \{0, 1\}$ and a single parity component if for any two vertices $u, v \in V_{C_i}$, $u \stackrel{x}{\Rightarrow} v$ exists in $C_i$ for exactly one $x$ from $\{0, 1\}$.*

**Lemma 7** [4]
**(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 [7]. 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 [4][1]. 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 $u \stackrel{0,1}{\rightarrow} v$ 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:

- For $i = 1, 2, \ldots, m$ do the following:
  - The starting graph for the $i^{\text{th}}$ 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 $u \stackrel{x}{\rightarrow} v$ in $G^{i-1}$ from a vertex $u$ not in $C_i$ to a vertex $v$ in $C_i$ the replacement is an edge $u \stackrel{0,1}{\rightarrow} \gamma_i$.
      - For an outgoing edge $u \stackrel{x}{\rightarrow} v$ in $G^{i-1}$ from a vertex $u$ in $C_i$ to a vertex $v$ not in $C_i$ the replacement is an edge $\gamma_i \stackrel{0,1}{\rightarrow} v$.
    - If $C_i$ is a single parity component then we do the following.

[1] The reader may find the following correspondence between our descriptions and those in [4] useful. A "gadget" for a strongly connected component is simply a vertex in our context. A doubly-labeled edge $u \stackrel{0,1}{\rightarrow} v$ in our context is a set of two parallel edges in [4], one labeled 0 and one labeled 1, such that a solution can contain either both of them or none of them.

· 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 $\gamma_i \xrightarrow{y} u$ for some $y \in \{0,1\}$.

∗ The resultant graph at the end of the $i^{\text{th}}$ 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.

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 $i = m, m-1, \ldots, 1$ 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** [4]

**(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 transforms $G$ 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 [4] 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:

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

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

The edge replacement is as follows[2]:

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

**Lemma 9 (see Lemma 10 of [4])** *For any two vertices $u, u' \in V_C$ with $u \in [\mathbf{i}]$ and $u' \in [\mathbf{j}]$ the path $u \overset{j-i \pmod 2}{\Longrightarrow} u'$ exists in $C$ but the path $u \overset{j-i+1 \pmod 2}{\Longrightarrow} u'$ is not in $C$.*

To verify ($\star$), one must consider the following cases.

**(i)** $u \overset{x}{\Rightarrow} w$ 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 $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\rightarrow} w' \overset{x_3}{\Rightarrow} w$ 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 $\gamma_C \overset{x_2+s \pmod 2}{\Longrightarrow} w' \overset{x_3}{\Rightarrow} w$ since the edge $\gamma_C \overset{x_2+s \pmod 2}{\longrightarrow} w'$ exists in $G'$. Suppose that $T_{\text{gadget}-\text{to}-\text{cycle}}$ translated this path to a path $u'' \overset{x_2+s-t \pmod 2}{\Longrightarrow} w' \overset{x_3}{\Rightarrow} w$ for some $u'' \in [\mathbf{t}]$. Then the path $u \overset{x_1}{\Rightarrow} u' \overset{t-s \pmod 2}{\Longrightarrow} u'' \overset{x_2+s-t \pmod 2}{\Longrightarrow} w' \overset{x_3}{\Rightarrow} w$ is of parity $x$.

**(ii)** $w \overset{x}{\Rightarrow} u$ is in $G$ when $u \in V_C$ and $w \notin V_C$. Similar to **(i)**.

**(iii)** $u \overset{x}{\Rightarrow} w$ is in $G$ when $u, w \notin V_C$ but the path contains at least one vertex from $V_C$. Let $u \overset{x_1}{\Rightarrow} u' \overset{x_2}{\Rightarrow} v' \overset{x_3}{\Rightarrow} w$ where $u'$ and $v'$ are the first and the last vertices that belong to $V_C$. But, then $u \overset{x_1}{\Rightarrow} u'$ and $v' \overset{x_3}{\Rightarrow} w$ exist in $E_1$ by **(i)** and **(ii)**, respectively, and $u' \overset{x_2}{\Rightarrow} v'$ exist in $E_2$ because $T_{\text{gadget}-\text{to}-\text{cycle}}$ 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 ($\star\star$). To verify ($\star\star$) one needs to consider the following cases:

**(i)** $\gamma_C \overset{x}{\rightarrow} w$ exists in $G'$. Using Lemma 9 and the construction of $T_{\text{cycle}-\text{to}-\text{gadget}}$ it follows that $G_2$ contains $u \overset{x-j \pmod 2}{\Longrightarrow} w$ for some $u \in [j] \subseteq V_C$. Suppose that this path is of the form $u \overset{y_1}{\Rightarrow} w' \overset{y_2}{\Rightarrow} w'' \overset{x-j-y_1-y_2 \pmod 2}{\Longrightarrow} w$ 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 $w' \overset{y_2}{\Rightarrow} w'' \overset{x-j-y_1-y_2 \pmod 2}{\Longrightarrow} w$ in $G_2$ translates to the path $\gamma_C \overset{j+y_1+y_2 \pmod 2}{\Rightarrow} w'' \overset{x-j-y_1-y_2}{\Longrightarrow} w$ in $G_1$ which is a path of parity $x$ from $\gamma_C$ to $w$.

**(ii)** $w \overset{x}{\rightarrow} \gamma_C$ is in $G'$. Similar to **(i)**.

**(iii)** $w_1 \overset{x_1}{\rightarrow} \gamma_C \overset{x_2}{\rightarrow} w_2$ is in $G'$. $w_1 \overset{x_1}{\Rightarrow} \gamma_C$ and $\Gamma_C \overset{x_2}{\Rightarrow} w$ exist in $G_1$ by **(ii)** and **(i)**, respectively.

To complete a proof of 3-approximation for GREEDY, we need to show that $T_{\text{cycle}-\text{to}-\text{gadget}}$ and $T_{\text{gadget}-\text{to}-\text{cycle}}$ 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.

[2]This is different from the corresponding edge replacement procedure discussed in [4].

**Proposition 3** *Let $\vec{e} = (e_1, e_2, \ldots, e_t)$ 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 $u \overset{x}{\Rightarrow} v$ that does not involve $e_p$. Since both $u$ and $v$ belong to $V_{C_i}$, the path $u \overset{x}{\Rightarrow} v$ 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 $u' \overset{x}{\Rightarrow} v'$ that does not involved $e_q$. Since $C_i$ is a strongly connected component, the path $u' \overset{x}{\Rightarrow} v'$ is of the following form $u' \overset{x_1}{\Rightarrow} u'' \overset{x_2}{\Rightarrow} v'' \overset{x_3}{\Rightarrow} v'$ with $x_1 + x_2 + x_3 = x \pmod 2$ and the path $u'' \overset{x_2}{\Rightarrow} v''$ 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 $u'' \overset{x_2}{\Rightarrow} v''$. ❑

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:

$$\Big( \underbrace{\qquad}_{\substack{\text{edges connecting}\\ \text{strong components}\\ \text{(Group 0)}}}, \underbrace{\qquad}_{\substack{\text{edges in } C_m\\ \text{(Group } m\text{)}}}, \underbrace{\qquad}_{\substack{\text{edges in } C_{m-1}\\ \text{(Group } m-1\text{)}}}, \cdots\cdots \underbrace{\qquad}_{\substack{\text{edges in } C_1\\ \text{(Group 1)}}} \Big)$$

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 $(\star)$ and $(\star\star)$ 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 $x_1, \ldots, x_n$, for each $x_i$ we have an edge $x_i \to r$ and an edge $r \to x_i$, for each $i$ we have edges $x_i \to x_{i+1}$ and all edge labels are zeroes. GREEDY may remove the edges $x_i \to x_{i+1}$ for $i = 1, 2, \ldots, n-1$ thus providing a solution with $2n$ edges. But an optimal solution with $n+1$ edges contains the edge $r \to x_1$, the edges $x_i \to x_{i+1}$ for each $i$ and the edge $x_n \to r$.

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 $i^{\text{th}}$ 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, $OPT(G) \geq \mathcal{L}$.*

**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 [4]) $|E \cap E_{C_i}| = |E_i|$. 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 Implemention 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 $u \stackrel{x}{\Rightarrow} v$ exists in $G$ using ideas similar to that in the Floyd-Warshall transitive closure algorithm; reference [4] 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 $e = u \stackrel{x}{\rightarrow} v$ for removal, tentatively removing it from $G$ and checking if the resulting graph has a path $u \stackrel{x}{\Rightarrow} v$. 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 $u \stackrel{x}{\Rightarrow} v$ 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 neglible 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 [4]), 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 $i^{\text{th}}$ 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 [22]. Our starting point is the list of experimentally observed causal relationships in ABA-induced closure collected by Li et al. and published as Table S1 in [22]. 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 [22]; the complete list of causal relationships is given in Table 1 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 [22]) 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 [22] 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 $A \to (x = C)$ and $(x = C) \to B$ based on the following two cases:

- if $a + b = 0 \pmod 2$ then both $A \to (x = C)$ and $(x = C) \to B$ have zero parities.
- if $a + b = 1 \pmod 2$ then $A \to (x = C)$ has parity 1 and $(x = C) \to B$ 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 Figures 2 **(a)**–**(b)**. Here is a brief summary of an overall comparison of the two networks:

- [22] has 54 vertices and 92 edges; our network has 57 vertices (3 extra pseudo-vertices) but only 84 edges.
- Both [22] and our network has 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 [22] reconstruction are present in our network as well. Our network has the extra path ROP10$\stackrel{1}{\Rightarrow}$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 ABA$\stackrel{0}{\rightarrow}$PLC, PA$\stackrel{1}{\rightarrow}$ABI1 and ROS$\stackrel{0}{\rightarrow}$CaIM that would be explainable by feedback processes. Some of our edges such as NO$\stackrel{0}{\rightarrow}$AnionEM correspond to paths in Li et al.'s reconstruction. Our graph contains the full pseudo-vertex-using representation of the process AtPP2C$\stackrel{1}{\rightarrow}$(ABA$\stackrel{0}{\rightarrow}$Closure) that Li et al. simplifies to AtPP2C$\stackrel{1}{\rightarrow}$ABA. We have pHc$\stackrel{0}{\rightarrow}$ROS and ROS$\stackrel{0}{\Rightarrow}$Atrboh where [22] has pH$\stackrel{0}{\rightarrow}$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

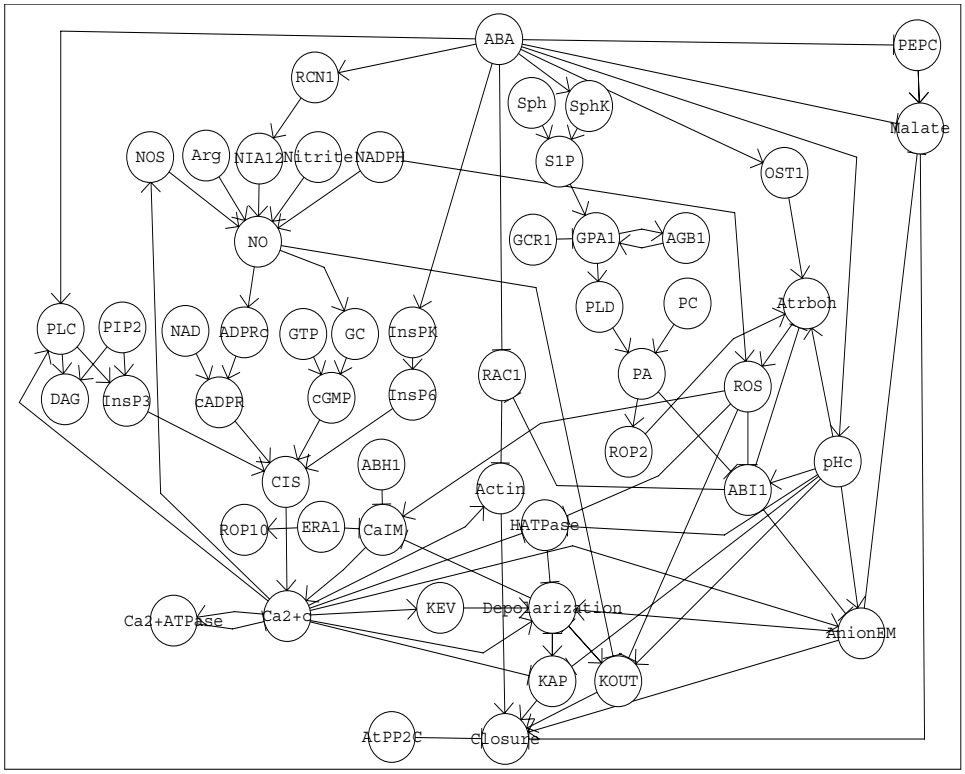
A variety of cellular interaction and regulatory networks have been mapped and graph theoretically characterized. One of the most frequently reported graph measures is the distribution of node degrees, *i.e.*, the distribution of the number of incoming or outgoing edges per node. A variety of networks, 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 [2, 11, 15, 21, 23]. Transcriptional regulatory networks exhibit a power-law out-degree distribution, while the in-degree distribution is more restricted [20, 26]. 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 [24]. We base the degree distributions on the yeast transcriptional regulatory network that has a maximum out-degree $\sim 150$ and maximum in-degree $\sim 15$ [20]. In our generated network the distribution of in-degree of the network is *exponential*, *i.e.*, Pr[in-degree$= x$]$= L\mathbf{e}^{-Lx}$ 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 $x \geq 1$ Pr[out-degree$= x$]$= cx^{-c}$ and for $x = 0$ Pr[out-degree$= 0$]$\geq c$ with $c$ between 2 and 3 and the maximum out-degree is 200. We varied the ratio of excitory to inhibitory edges between 2 and 4. Since there are no known biological estimates of critical edges[3] 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.

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

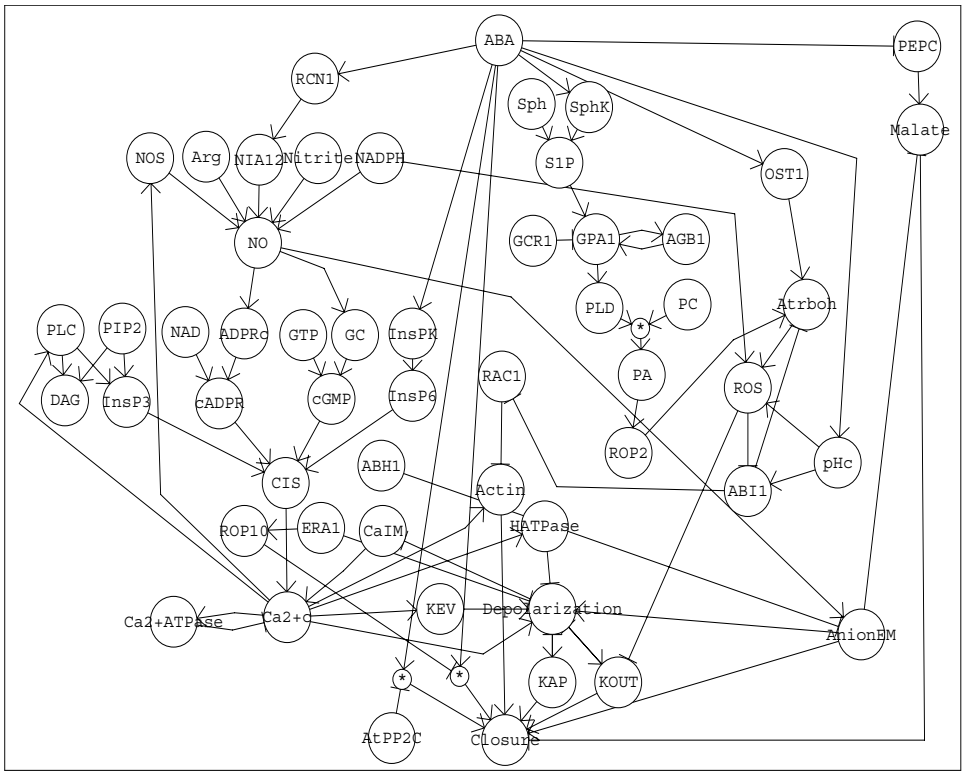
$$\text{OPT} = \max\{n + s - c, t, \mathcal{L}\} \tag{1}$$

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 $u\stackrel{x}{\rightarrow}v$ such that

[3]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. [23] 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 [11, 12, 21]. 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.




**(a)**




**(b)**

Figure 2: **(a)** The network manually synthesized by Li et al. [22] redrawn for easier visual comparison. **(b)** The network synthesized in this paper. A pseudo-vertex is displayed as ⊛.

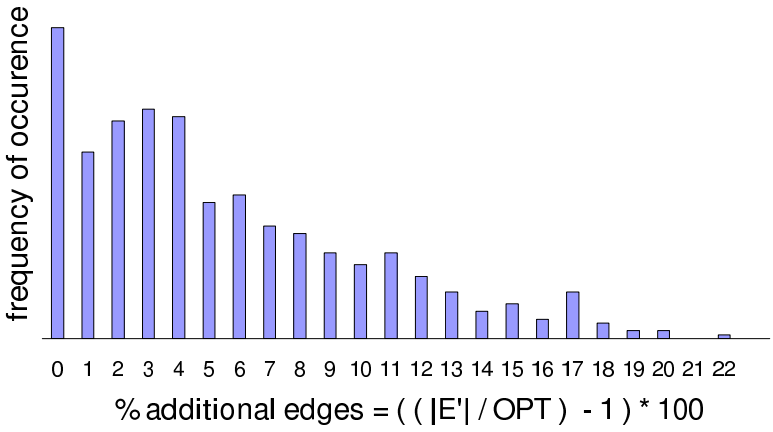



Figure 3: A plot of the empirical performance of our BTR algorithm on the 561 simulated interaction networks. $E'$ is our solution, OPT is the trivial lower bound on the minimum number of edges described in Equation (1) and $100 \times \left(\frac{|E'|}{\text{OPT}} - 1\right)$ is the percentage of additional 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).

| 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 |

Figure 4: Basic statistics of the simulated networks used in Figure 3.

either $u \xrightarrow{x} v \in E_{\text{critical}}$ or there is no alternate path $u \overset{x}{\Rightarrow} v$ 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 Figure 4. 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 `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.

## 7 Conclusions and Further Work

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 `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.

## References


[1] A. Aho, M. R. Garey and J. D. Ullman. *The transitive reduction of a directed graph*, SIAM Journal of Computing, 1 (2), pp. 131-137, 1972.

[2] R. Albert and A.-L. Barabási. *Statistical mechanics of complex networks*, Reviews of Modern Physics, 74 (1), 47-97, 2002.

[3] B. Alberts. Molecular biology of the cell, New York: Garland Pub., 1994.

[4] R. Albert, B. DasGupta, R. Dondi and E. Sontag. *Inferring (Biological) Signal Transduction Networks via Transitive Reductions of Directed Graphs*, to appear in Algorithmica.

[5] G. W. Carter. *Inferring network interactions within a cell*, Briefings in Bioinformatics, 6 (4), pp. 380-389, 2005.

[6] T. Chen, V. Filkov and S. Skiena. *Identifying Gene Regulatory Networks from Experimental Data*, Third Annual International Conference on Computational Moledular Biology (RECOMB), pp. 94-103, 1999.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*, The MIT Press, 2001.

[8] B. DasGupta, G. A. Enciso, E. D. Sontag and Y. Zhang. *Algorithmic and Complexity Results for Decompositions of Biological Networks into Monotone Subsystems*, to appear in Biosystems (conference version in WEA-2006, LNCS 4007, pp. 253-264, Springer-Verlag, 2006).

[9] V. Filkov. *Identifying Gene Regulatory Networks from Gene Expression Data*, in Handbook of Computational Molecular Biology (S. Aluru editor), Chapman & Hall/CRC Press, 2005.

[10] G. N. Frederickson and J. JàJà. *Approximation algorithms for several graph augmentation problems*, SIAM Journal of Computing, 10 (2), pp. 270-283, 1981.

[11] L. Giot, J. S. Bader et al. *A protein interaction map of Drosophila melanogaster*, Science 302, 1727-1736, 2003.

[12] J. D. Han, N. Bertin et al. *Evidence for dynamically organized modularity in the yeast protein-protein interaction network*, Nature 430, 88-93, 2004.

[13] R. Heinrich and S. Schuster. The regulation of cellular systems, New York: Chapman & Hall, 1996.

[14] J. E. Hopcroft and R. M. Karp. *An $n^{\frac{5}{2}}$ algorithm for maximum matching in bipartite graphs*, SIAM Journal of Computing, 2 , pp. 225-231, 1973.

[15] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai and A.-L. Barabási. *The large-scale organization of metabolic networks* Nature 407, pp. 651-654, 2000.

[16] H. D. Jong. *Modelling and Simulation of Genetic Regulatory Systems: A Literature Review*, Journal of Computational Biology, Volume 9, Number 1, pp. 67-103, 2002.

[17] S. Khuller, B. Raghavachari and N. Young. *Approximating the minimum equivalent digraph*, SIAM Journal of Computing, 24(4), pp. 859-872, 1995.

[18] S. Khuller, B. Raghavachari and N. Young. *On strongly connected digraphs with bounded cycle length*, Discrete Applied Mathematics, 69 (3), pp. 281-289, 1996.

[19] S. Khuller, B. Raghavachari and A. Zhu. *A uniform framework for approximating weighted connectivity problems*, 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 937-938, 1999.

[20] T. I. Lee, N. J. Rinaldi et al. *Transcriptional regulatory networks in Saccharomyces cerevisiae*, Science 298, 799-804, 2002.

[21] S. Li, C. M. Armstrong et al. *A map of the interactome network of the metazoan C. elegans*, Science 303, 540-543, 2004.

[22] S. Li, S. M. Assmann and R. Albert. *Predicting Essential Components of Signal Transduction Networks: A Dynamic Model of Guard Cell Abscisic Acid Signaling*, to appear in PLoS Biology, 4(10), October 2006.

[23] A. Ma'ayan, S. L. Jenkins, S. Neves, A. Hasseldine, E. Grace, B. Dubin-Thaler, N. J. Eungdamrong, G. Weng, P. T. Ram, J. J. Rice, A. Kershenbaum, G. A. Stolovitzky, R. D. Blitzer, R. Iyengar. *Formation of Regulatory Patterns During Signal Propagation in a Mammalian Cellular Network*, Science, Vol. 309. no. 5737, pp. 1078-1083, 12 August 2005.

[24] M. E. J. Newman, S. H. Strogatz and D. J. Watts. *Random graphs with arbitrary degree distributions and their applications*, Phys. Rev. E, 64 (2), pp. 026118-026134, July 2001.

[25] R. Z. Norman and M. O. Rabin. *An algorithm for a minimum cover of a graph*, Proceeding of the American Mathematical Society, 10, pp. 315-319, 1959.

[26] S. S. Shen-Orr, R. Milo, S. Mangan and U. Alon. *Network motifs in the transcriptional regulation network of Escherichia coli*, Nature Genetics 31, pp. 64-68, 2002.

[27] A. Vetta. *Approximating the minimum strongly connected subgraph via a matching lower bound*, 12th ACM-SIAM Symposium on Discrete Algorithms, pp. 417-426, 2001.

[28] A. Wagner. *Estimating Coarse Gene Network Structure from Large-Scale Gene Perturbation Data*, Genome Research, 12, pp. 309-315, 2002.

# APPENDIX 1

| interaction | critical | enzymatic | interaction | critical | enzymatic |
|---|---|---|---|---|---|
| ABA $\xrightarrow{0}$ SphK | No | No | ABA $\xrightarrow{0}$ OST1 | No | No |
| ABA $\xrightarrow{0}$ CaIM | No | No | ABA $\xrightarrow{0}$ InsP6 | No | No |
| ABA $\xrightarrow{0}$ Ca2+c | No | No | ABA $\xrightarrow{0}$ NO | No | No |
| ABA $\xrightarrow{0}$ InsP3 | No | No | ABA $\xrightarrow{0}$ AnionEM | No | No |
| ABA $\xrightarrow{1}$ PEPC | No | No | ABA $\xrightarrow{1}$ Malate | No | No |
| ABA $\xrightarrow{1}$ HATPase | No | No | ABA $\xrightarrow{1}$ RAC1 | No | No |
| ABA $\xrightarrow{0}$ PLD | No | No | ABA $\xrightarrow{0}$ ROS | No | No |
| Ca2+c $\xrightarrow{1}$ CaIM | No | No | Ca2+c $\xrightarrow{0}$ KEV | No | No |
| Ca2+c $\xrightarrow{0}$ AnionEM | No | No | InsP6 $\xrightarrow{0}$ Ca2+c | No | No |
| InsP6 $\xrightarrow{0}$ CIS | No | No | ROS $\xrightarrow{0}$ CaIM | No | No |
| ROS $\xrightarrow{0}$ Closure | No | No | ROS $\xrightarrow{1}$ ABI1 | No | No |
| ROS $\xrightarrow{1}$ KOUT | No | No | pHc $\xrightarrow{0}$ KOUT | No | No |
| pHc $\xrightarrow{0}$ ABI1 | No | No | pHc $\xrightarrow{0}$ ROS | No | No |
| pHc $\xrightarrow{0}$ HATPase | No | No | PA $\xrightarrow{1}$ ABI1 | No | No |
| PA $\xrightarrow{0}$ Closure | No | No | PA $\xrightarrow{0}$ ROS | No | No |
| NO $\xrightarrow{0}$ Closure | No | No | NO $\xrightarrow{0}$ AnionEM | No | No |
| 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) | - | - |

Table 1. Regulatory interactions between ABA signal transduction pathway components [22].

Table 1 (continued)

| interaction | critical | enzymatic | interaction | critical | enzymatic |
|---|---|---|---|---|---|
| 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) | - | - |
| ROS $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ CaIM) | - | - | NADPH $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ CaIM) | - | - |
| NAD $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ CaIM) | - | - | ERA1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ CaIM) | - | - |
| ERA1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - | RCN1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| RCN1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ AnionEM) | - | - | RCN1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Ca2+c) | - | - |
| OST1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - | OST1 $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ ROS) | - | - |
| PLC $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - | Ca2+c $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| AnionEM $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - | PLD $\xrightarrow{0}$ (PC $\xrightarrow{0}$ PA) | - | - |
| PLD $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - | PLC $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| ABA $\xrightarrow{0}$ (PLD $\xrightarrow{0}$ PA) | - | - | ABA $\xrightarrow{0}$ (PLD $\xrightarrow{0}$ PA) | - | - |
| ROP2 $\xrightarrow{0}$ (PA $\xrightarrow{0}$ ROS) | - | - | Actin $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| Ca2+c $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Actin) | - | - | RAC1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| ROP10 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - | ROS $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| GCR1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - | GCR1 $\xrightarrow{1}$ (S1P $\xrightarrow{0}$ Closure) | - | - |
| cADPR $\xrightarrow{0}$ (Ca2+c $\xrightarrow{0}$ CIS) | - | - | AnionEM $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ Closure) | - | - |
| CaIM $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ KOUT) | - | - | cADPR $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ KOUT) | - | - |
| PLC $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ KOUT) | - | - | ROS $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ CaIM) | - | - |
| Ca2+c $\xrightarrow{1}$ (Depolarization $\xrightarrow{0}$ KAP) | - | - | pHc $\xrightarrow{1}$ (Depolarization $\xrightarrow{0}$ KAP) | - | - |
| ABH1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - | ABH1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Ca2+c) | - | - |
| ROS $\xrightarrow{0}$ (ABA $\xrightarrow{1}$ HATPase) | - | - | ABI1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ AnionEM) | - | - |
| ABI1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ ROS) | - | - | ABI1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Ca2+c) | - | - |
| AtPP2C $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ Closure) | - | - | Ca2+c $\xrightarrow{0}$ (PLC $\xrightarrow{0}$ InsP3) | - | - |
| GPA1 $\xrightarrow{0}$ AGB1 | No | No | AGB1 $\xrightarrow{0}$ GPA1 | No | No |
| AtPP2C $\xrightarrow{1}$ Closure | No | No | NO $\xrightarrow{0}$ ADPRc | No | No |
| Ca2+c $\xrightarrow{0}$ HATPase | No | No | ABI1 $\xrightarrow{1}$ Atrboh | No | No |
| NO $\xrightarrow{0}$ GC | No | No | ABA $\xrightarrow{0}$ pHc | No | No |
| PA $\xrightarrow{0}$ ROP2 | No | No | PEPC $\xrightarrow{0}$ Malate | Yes | Yes |
| ABI1 $\xrightarrow{1}$ (ABA $\xrightarrow{0}$ ROS) | - | - | ABA $\xrightarrow{0}$ PLC | No | No |
| Depolarization $\xrightarrow{0}$ KOUT | Yes | No | Depolarization $\xrightarrow{0}$ KAP | Yes | No |
| Depolarization $\xrightarrow{1}$ CaIM | Yes | No | ABI1 $\xrightarrow{0}$ (ABA $\xrightarrow{1}$ RAC1) | - | - |
| InsPK $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ AnionEM) | - | - | InsPK $\xrightarrow{0}$ (ABA $\xrightarrow{0}$ InsP6) | - | - |
| S1P $\xrightarrow{0}$ GPA1 | No | No | | | |

Table 1. Regulatory interactions between ABA signal transduction pathway components [22].

# APPENDIX 2: Details of a Mixed ILP Formulation for BTR

Below 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 [7] 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, $f^{even}_{u,v,e}$ and $f^{odd}_{u,v,e}$ , 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 incoming($x$) and outgoing($x$) refer to the sets $\{u \,|\, u \to x \in E_1\}$ and $\{u \,|\, x \to u \in E_1\}$, respectively):

*minimize* $\sum_{e \in E_1} x_e$
*subject to:*

$$x_e = 1 \qquad \forall e \in E_1 \cap E_{\text{critical}}$$
$$\sum_{x \in \text{outgoing}(u)} f^{even}_{u,v,x} = 1 \qquad \forall e = u \to v \in E_1$$
$$\sum_{x \in \text{incoming}(v)} f^{even}_{u,v,x} - f^{odd}_{u,v,x} = -1 \qquad \forall e = u \to v \in E_1 \text{ and } w(e) = 0$$
$$\sum_{x \in \text{incoming}(v)} f^{even}_{u,v,x} - f^{odd}_{u,v,x} = 1 \qquad \forall e = u \to v \in E_1 \text{ and } w(e) = 1$$
$$\sum_{y \in \text{incoming}(x)} f^{even}_{u,v,y} - \sum_{y \in \text{outgoing}(x)} f^{odd}_{u,v,y} = 0 \qquad \forall e = u \to v \in E_1 \text{ and } \forall x \in V_1 - \{u, v\}$$
$$\sum_{y \in \text{incoming}(x)} f^{odd}_{u,v,y} - \sum_{y \in \text{outgoing}(x)} f^{even}_{u,v,y} = 0 \qquad \forall e = u \to v \in E_1 \text{ and } \forall x \in V_1 - \{u, v\}$$
$$\sum_{f^{even}_{u,v,e} + f^{odd}_{u,v,e}} \leq x_e \; \forall e \in E_1, u, v \in V$$

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.